Universität Stuttgart
Institut für Baustatik
und Baudynamik

52 (2009)

Effiziente Algorithmen für die Fluid–Struktur–Wechselwirkung

Malte von Scheven

Malte von Scheven

Effiziente Algorithmen für die Fluid-Struktur-Wechselwirkung

von

Malte von Scheven

Bericht Nr. 52
Institut für Baustatik und Baudynamik der Universität Stuttgart
Professor Dr.-Ing. habil. M. Bischoff
2009

© Malte von Scheven

Berichte können bezogen werden über:
Institut für Baustatik und Baudynamik
Universität Stuttgart
Pfaffenwaldring 7
70550 Stuttgart

Tel.: 0711 - 685 66123
Fax: 0711 - 685 66130
E-Mail: sekretariat@ibb.uni-stuttgart.de
http://www.ibb.uni-stuttgart.de/

Alle Rechte, insbesondere das der Übersetzung in andere Sprachen, vorbehalten. Ohne Genehmigung des Autors ist es nicht gestattet, diesen Bericht ganz oder teilweise auf photomechanischem, elektronischem oder sonstigem Wege zu kommerziellen Zwecken zu vervielfältigen.

D93 - Dissertation an der Universität Stuttgart
ISBN 978-3-00-027330-8

Effiziente Algorithmen für die Fluid-Struktur-Wechselwirkung

Von der Fakultät Bau- und Umweltingenieurwissenschaften
der Universität Stuttgart zur Erlangung der Würde eines
Doktors der Ingenieurwissenschaften (Dr.-Ing.) genehmigte Abhandlung

vorgelegt von

Malte von Scheven

aus Bochum

Hauptberichter:	Prof. Dr.-Ing. Dr.-Ing. E.h. Dr. h.c. Ekkehard Ramm, Stuttgart
Mitberichter:	Prof. Dr. rer.nat. Ernst Rank, München
Mitberichter:	Prof. Dr.-Ing. habil. Manfred Bischoff, Stuttgart
Tag der mündlichen Prüfung:	19. Februar 2009

Institut für Baustatik und Baudynamik der Universität Stuttgart

2009

Kurzfassung

Gekoppelte Problemstellungen, zu denen auch die Fluid-Struktur-Wechselwirkung zählt, treten in vielen Bereichen des Ingenieurwesens auf. Sie zeichnen sich bei der Lösung häufig durch eine hohe Komplexität aus. Besonders bei dreidimensionalen Fragestellungen resultiert dies in sehr langen Rechenzeiten. In dieser Arbeit wird die partitionierte Lösung von Wechselwirkungs-Problemen zwischen dünnen Strukturen und inkompressiblen Fluiden betrachtet. Die Struktur wird dabei mit den Gleichungen der nichtlinearen Elastodynamik und das Fluid mit den inkompressiblen Navier-Stokes-Gleichungen beschrieben. Für die partitionierte Lösung werden die beiden Felder einzeln mithilfe der Finite-Element-Methode im Raum und mittels Differenzenverfahren in der Zeit diskretisiert. Der Schwerpunkt dieser Arbeit liegt in der Entwicklung effizienter Verfahren zur Lösung komplexer, dreidimensionaler Fragestellungen. Dabei werden zunächst die Einzelfelder betrachtet und im Anschluss daran das gekoppelte Problem.

Zur Vernetzung von großen Gebieten wird ein zweistufiges Verfahren vorgestellt, bei dem unter Berücksichtigung der exakten Geometrie ein grobes Netz aus dem Präprozessor auf dem Hochleistungsrechner verfeinert wird. Bei der Berechnung der Elementmatrizen eines Finite-Element-Programms für unstrukturierte Netze können auf einem Vektorrechner durch eine Gruppierung der Elemente und Umstrukturierung des Programmcodes die Vorteile des Prozessors genutzt und die Rechenzeit erheblich verkürzt werden. Auch bei der Lösung der linearen Gleichungssysteme kann durch die Wahl des Iterationsverfahrens und Vorkonditionierers viel Rechenzeit eingespart werden. Der Einfluss verschiedener Parameter auf die Effizienz des Lösers wird für eine reine Fluid-Simulation untersucht.

Bei der partitionierten Behandlung der Wechselwirkung von inkompressiblen Fluiden mit dünnen Strukturen sind in der Regel implizite, d.h. iterativ gestaffelte Kopplungsverfahren erforderlich. Insbesondere bei einer starken Kopplung der beiden Felder führt dies aufgrund der zusätzlichen Iteration zu sehr langen Rechenzeiten. In dieser Arbeit werden zunächst die gebräuchlichsten iterativ gestaffelten Kopplungsverfahren in einer einheitlichen Darstellung vorgestellt. Durch Verwendung der Lösung des gekoppelten Problems auf einer gröberen Diskretisierung wird die Kopplungsiteration beschleunigt. Abschließend werden diese Zwei-Level-Verfahren für verschieden stark gekoppelte Beispiele mit anderen iterativ gestaffelten Methoden verglichen.

Abschließend werden mit numerischen Beispielen der Kopplung von dünnen Schalenstrukturen mit dreidimensionalen Strömungen Anwendungsmöglichkeiten der vorgestellten effizienten Algorithmen aufgezeigt.

Abstract

Coupled problems, like fluid-structure interaction, arise in various areas of engineering. Often they are characterized by high complexity. Especially for three-dimensional problems this results in very long simulation times. In the present work partitioned solution schemes are used for fluid-structure interaction of thin-walled structures and incompressible fluids. The structural domain is governed by the nonlinear elastodynamic equations while the fluid dynamics is described by the incompressible Navier-Stokes equations. Both fields are discretized by finite elements in space and finite difference methods in time. The focus of this work is the development of efficient algorithms for the solution of complex, three-dimensional fluid-structure interaction problems. In the first part of this work the efficiency of the single fields is examined while the second part focuses is on the coupling.

For mesh generation in large domains a two level procedure is introduced. A coarse mesh provided by the preprocessor is refined on the high performance computer taking into account the exact geometry. The calculation of element matrices of a finite element code for unstructured meshes can be accelerated on a vector computer by grouping the elements and restructuring the code making use of the advantages of vector processors. In addition, for the solution of the linear system of equations the necessary cpu time can be reduced by the correct choice of iteration scheme and preconditioner. The influence of different parameters on the efficiency of the solver is studied for a fluid simulation.

The simulation of interaction between incompressible flows and thin-walled structures often requires implicit i.e. iterative staggered coupling algorithms. Especially for strongly coupled problems the additional iteration leads to very long simulation times. In the present work commonly used iterative staggered coupling schemes are presented in a consistent notation. Using the solution of the coupled problem on a coarse discretization as a predictor the coupling iteration can be accelerated. Finally, convergence rate and calculation time of these new two-level coupling algorithms are compared with other iterative staggered schemes.

Numerical examples for the coupling of thin-walled shell structures and three-dimensional flows indicate possible applications of the presented efficient algorithms.

Vorwort

Mein besonderer Dank gilt Herrn Professor Ekkehard Ramm, unter dessen Leitung ich diese Arbeit in den Jahren 2002 bis 2009 angefertigt habe und der mich in meiner wissenschaftlichen Arbeit unter anderem durch anregende inhaltliche Diskussionen und die Möglichkeit der Teilnahme an zahlreichen Workshops und Konferenzen vielfältig unterstützt hat.

Für das Interesse an meiner Arbeit, deren intensive Durchsicht und Besprechung möchte ich mich bei Herrn Professor Manfred Bischoff bedanken. Vielen Dank an ihn und Herrn Professor Ernst Rank vom Lehrstuhl für Computation in Engineering der Technischen Universität München für die Übernahme der Mitberichte.

Durch die Unterstützung des Höchstleistungsrechenzentrums der Universität Stuttgart ist diese Arbeit erst möglich geworden. Insbesondere bedanke ich mich bei Herrn Uwe Küster für seine wertvollen Hinweise zur effizienten Programmierung auf Vektorrechnern und bei Herrn Sunil Reddy Tiyyagura für die Entwicklung des Löserpakets BLIS und seine tatkräftige Unterstützung in vielen praktischen Fragen des Hochleistungsrechnens.

Meinen Kolleginnen und Kollegen am Institut für Baustatik und Baudynamik der Universität Stuttgart gilt mein Dank für das freundschaftliche und produktive Arbeitsklima und viele fachliche Gespräche.

Stuttgart, im März 2009 Malte von Scheven

Inhaltsverzeichnis

Abbildungsverzeichnis **ix**

Tabellenverzeichnis **xiii**

Abkürzungen und Bezeichnungen **xv**

1 Einleitung **1**
1.1 Motivation und Zielsetzung 1
1.2 Gliederung der Arbeit 3

2 Grundlagen **5**
2.1 Strukturdynamik 6
2.1.1 Kontinuumsmechanische Grundgleichungen 6
2.1.2 Anfangs-Randwert-Problem 9
2.1.3 Diskretisierung im Raum 12
2.1.4 Diskretisierung in der Zeit 13
2.1.5 Iteratives Lösungsverfahren 15
2.1.6 Reduktion auf den Kopplungsrand 16
2.2 Fluiddynamik 16
2.2.1 Kontinuumsmechanische Grundgleichungen 18
2.2.2 Anfangs-Randwert-Problem 20
2.2.3 Diskretisierung im Raum 24
2.2.4 Diskretisierung in der Zeit 29
2.2.5 Iteratives Lösungsverfahren 31
2.2.6 Reduktion auf den Kopplungsrand 31

2.3 Netzdynamik 32

3 Strömungssimulation durch Hochleistungsrechnen 35
3.1 Grundlagen des Hochleistungsrechnens 37
3.1.1 Rechengeschwindigkeit messen 37
3.1.2 Vektorprozessoren 38
3.2 Netzerzeugung in zwei Stufen 40
3.2.1 Problemstellung 40
3.2.2 Vorteile 41
3.2.3 Einschränkungen 42
3.2.4 Umsetzung 43
3.2.5 Berücksichtigung der exakten Geometrie 46
3.3 Vektorisierung 53
3.3.1 Regeln für die Vektorisierung 53
3.3.2 Vektorisierungskonzept für die Berechnung von Elementmatrizen . 56
3.3.3 Einflüsse auf die Effizienz 59

4 Iterative Methoden zur Lösung linearer Gleichungssysteme 67
4.1 Einführung in iterative Löser 67
4.2 Klassifizierung von Iterationsverfahren 68
4.3 Lineare Iterationsverfahren 69
4.3.1 Jacobi-Verfahren 70
4.3.2 Gauß-Seidel-Verfahren 71
4.3.3 Symmetrisches Gauß-Seidel-Verfahren 72
4.3.4 Gauß-Seidel-Relaxationsverfahren 72
4.4 Projektionsmethoden und Krylow-Unterraum-Verfahren 73
4.4.1 CG-Verfahren 74
4.4.2 BiCGSTAB-Verfahren 77
4.4.3 GMRES-Verfahren 78
4.5 Vorkonditionierung 80
4.5.1 Splitting-assoziierte Vorkonditionierer 81
4.5.2 Unvollständige LU-Zerlegung 82
4.5.3 Block-Vorkonditionierer 83
4.6 Vergleich der Effizienz 84
4.6.1 Problemstellung 85
4.6.2 „Block-based Linear Iterative Solver“ (BLIS) 86
4.6.3 Einfluss der Löserparameter 87
4.6.4 Einfluss der Diskretisierungs-Parameter 94
4.6.5 Einfluss der Hardware-Architektur 97

5 Kopplung von Fluid und Struktur 101
5.1 Einleitung . . . 101
5.1.1 Klassifizierung von gekoppelten Problemen . . . 102
5.1.2 Lösungsstrategien für gekoppelte Probleme . . . 102
5.2 Das gekoppelte Problem . . . 104
5.2.1 Kopplungsbedingungen . . . 105
5.2.2 Schreibweisen für das gekoppelte Fluid-Struktur-Problem . . . 106
5.3 Überblick über Lösungsverfahren für die Fluid-Struktur-Kopplung . . . 107
5.3.1 Monolithische Lösungsverfahren . . . 108
5.3.2 Partitionierte Lösungsverfahren . . . 108
5.4 Demonstrationsbeispiel . . . 110
5.5 Einfach gestaffelte Lösungsverfahren . . . 110
5.5.1 Parallel gestaffeltes Lösungsverfahren . . . 111
5.5.2 Sequenziell gestaffeltes Lösungsverfahren . . . 113
5.5.3 Strukturprädiktor . . . 114
5.5.4 „Artificial Added Mass“-Effekt . . . 115
5.6 Iterativ gestaffelte Lösungsverfahren . . . 116
5.6.1 Block-Iterations-Verfahren . . . 117
5.6.2 Zwei-Level-Verfahren . . . 126
5.6.3 Newton-Krylow-Verfahren . . . 135
5.6.4 Weitere Kopplungsverfahren . . . 141
5.7 Vergleich der Kopplungsverfahren . . . 142
5.7.1 Sequenziell gestaffeltes Verfahren . . . 142
5.7.2 Block-Gauß-Seidel-Verfahren . . . 143
5.7.3 Grob-Gitter-Prädiktor . . . 143
5.7.4 Newton-Krylow-Verfahren mit Finiter Differenz . . . 146

6 Numerische Beispiele 147
6.1 Gebäude mit starrem Dach . . . 147
6.1.1 Diskretisierung . . . 148
6.1.2 Randbedingungen . . . 149
6.1.3 Ergebnisse . . . 150
6.2 Gebäude mit flexiblem Dach . . . 152
6.2.1 Fluid-Diskretisierung . . . 153
6.2.2 Struktur-Diskretisierung . . . 153
6.2.3 Randbedingungen . . . 154
6.2.4 Kopplung . . . 154
6.2.5 Ergebnisse . . . 155
6.3 Flattern einer „Fahne“ im Wind . . . 158
6.3.1 Fluid-Diskretisierung . . . 159

6.3.2 Struktur-Diskretisierung 160
6.3.3 Randbedingungen 160
6.3.4 Kopplung 161
6.3.5 Ergebnisse 161

7 Zusammenfassung und Ausblick **165**
7.1 Zusammenfassung 165
7.2 Ausblick 166

Literaturverzeichnis **169**

Index **193**

Abbildungsverzeichnis

2.1 Referenz- und Momentankonfiguration in der Strukturdynamik. 7
2.2 Tonti-Diagramm der starken Form des Anfangs-Randwert-Problems der Strukturdynamik. 10
2.3 Kontinuumsmechanische Gebiete für die ALE-Formulierung der Fluiddynamik. 17
2.4 Tonti-Diagramm der starken Form des Anfangs-Randwert-Problems für instationäre, inkompressible Strömungen. 22

3.1 Segmentierung der Subtraktion von zwei Gleitkommazahlen. 39
3.2 Pipelining der Subtraktion von zwei Gleitkommazahlen. 40
3.3 Element-Unterteilung: Sukzessive Erzeugung der Slave-Knoten. 43
3.4 Master- und Slave-Diskretisierung nach der Element-Unterteilung. 44
3.5 Element-Unterteilung: Veränderung der Elementtopologie und des Grads der Ansatzfunktionen. 45
3.6 Kubische Bernstein-Polynome. 48
3.7 Parametrische Beschreibung einer Kurve $C(s)$ mit den erforderlichen Kontrollpunkten, Knoten und dem Kontrollpunkt-Polygon. 49
3.8 Quadratische B-Spline-Basisfunktionen zur Beschreibung eines Viertelkreises. 51
3.9 Element-Unterteilung eines linearen Linienelements auf einem Viertelkreis. 52
3.10 Struktogramm zur typischen Berechnung der Elementmatrizen. 57
3.11 Struktogramm zur vektorisierten Berechnung der Elementmatrizen. . . . 58
3.12 Pseudo-Code für eine Funktion zur Berechnung der Jacobi-Matrix: nicht vektorisierende Version. 59

3.13 Pseudo-Code für eine Funktion zur Berechnung der Jacobi-Matrix: vektorisierende Version. 59
3.14 Zeit für die Berechnung repräsentativer Matrixanteile für unterschiedliche Größen der Elementgruppen. 62
3.15 Anteile an der Rechenzeit für die Simulation von 32 Zeitschritten der Beltrami-Strömung auf dem Vektorprozessor SX-6+. 64

4.1 Geometrie und Randbedingungen eines umströmten Zylinders mit quadratischer Grundfläche. 85
4.2 BiCGSTAB und SGS ($\omega = 0{,}9$): Einfluss der Anzahl der Vorkonditionierer-Iterationen auf Rechenzeit und Iterationszahl. 88
4.3 GMRES(120) und Jacobi ($\omega = 0{,}4$): Einfluss der Anzahl der Vorkonditionierer-Iterationen auf Rechenzeit und Iterationszahl. 88
4.4 BiCGSTAB und SGS(2): Einfluss des Relaxationsparameters ω auf Rechenzeit und Iterationszahl. 89
4.5 BiCGSTAB und Jacobi(2): Einfluss des Relaxationsparameters ω auf Rechenzeit und Iterationszahl. 90
4.6 GMRES und BILU: Einfluss der Größe des Krylow-Unterraums auf Rechenzeit und Iterationszahl. 91
4.7 GMRES und BILU: Einfluss der Größe des Krylow-Unterraums auf den Speicherbedarf. 92
4.8 Vergleich der Rechenzeiten und benötigten Krylow-Iterationen für die „optimalen“ Löserparameter. 93
4.9 Einfluss der Diskretisierungsparameter auf Konvergenzgeschwindigkeit und Rechenzeit des linearen Lösers. 95
4.10 Einfluss der Hardware-Architektur auf Rechenzeit und Iterationszahl des linearen Lösers. 98

5.1 Geometrie und Materialparameter des 2-D-Membrandachs. 110
5.2 Ablaufdiagramm zum parallel gestaffelten Lösungsverfahren mit Prädiktor. 112
5.3 Ablaufdiagramm zum sequenziell gestaffelten Lösungsverfahren mit Prädiktor. 113
5.4 Ablaufdiagramm zum Block-Gauß-Seidel-Verfahren mit Prädiktor und Relaxation. 118
5.5 Struktogramm für einen Zeitschritt des Block-Gauß-Seidel-Verfahrens mit Prädiktor und Relaxation. 121
5.6 Ablaufdiagramm zum Block-Jacobi-Verfahren mit Prädiktor. 122
5.7 Struktogramm zur Berechnung des Relaxationsparameters $\omega^{(k)}$ mit dem Aitken-Verfahren. 124

5.8 Struktogramm zur Berechnung des Relaxationsparameters $\omega^{(k)}$ mit dem Gradienten-Verfahren. 125
5.9 Ablaufdiagramm zum Block-Gauß-Seidel-Verfahren mit Grob-Gitter-Prädiktor. 128
5.10 Struktogramm für den Grob-Gitter-Prädiktor. 130
5.11 Verschiebungen am Kopplungsrand auf dem groben und feinen Gitter jeweils zu Beginn und am Ende der Iteration. 131
5.12 Korrektur der Verschiebungen am Kopplungsrand durch die Grob- und Fein-Gitter-Iteration. 132
5.13 Struktogramm zur Berechnung des Relaxationsparameters ω_i mit dem Gradienten-Verfahren auf dem Grob-Gitter. 133
5.14 Ablaufdiagramm zum Newton-Krylow-Verfahren. 137
5.15 Vergleich der benötigten Iterationen und aufsummierten Rechenzeit in den ersten 20 Zeitschritten für das Block-Gauß-Seidel-Verfahren mit und ohne Grob-Gitter-Prädiktor. 145

6.1 Gebäude mit starrem Dach: Geometrie und Randbedingungen. 148
6.2 Gebäude mit starrem Dach: Grobes Netz in zwei Ansichten. 149
6.3 Gebäude mit starrem Dach: räumliche (links) und zeitliche (rechts) Veränderung der Einström-Randbedingung. 150
6.4 Gebäude mit starrem Dach: zeitliche Veränderung des Fluid-Drucks in Dachmitte ($\mathbf{x} = [50{,}0;\ 50{,}0;\ 5{,}0]^{\mathrm{T}}$ m). 151
6.5 Gebäude mit starrem Dach: Momentaufnahmen des Fluid-Drucks in der Schnittebene $y = 50{,}0$ m (Ausschnitt 41,0 m $\leq x \leq$ 76,0 m und 0,0 m $\leq z \leq$ 20,0 m). 152
6.6 Gebäude mit flexiblem Dach: zeitliche Veränderung des Fluiddrucks p in Dachmitte ($\mathbf{x} = [50{,}0;\ 50{,}0;\ 5{,}0]^{\mathrm{T}}$ m). 155
6.7 Gebäude mit flexiblem Dach: zeitliche Veränderung der vertikalen Verschiebungskomponente d_z in Dachmitte ($\mathbf{x} = [50{,}0;\ 50{,}0;\ 5{,}0]^{\mathrm{T}}$ m). 156
6.8 Gebäude mit flexiblem Dach: Momentaufnahmen des Verformungszustands des Membrandachs mit Farbkontur der vertikalen Veschiebungskomponente d_z (Verformung 5-fach überhöht). 157
6.9 Fahne im Wind: Geometrie und Randbedingungen. 158
6.10 Fahne im Wind: zeitliche Veränderung der Einström-Randbedingung. . . 161
6.11 Fahne im Wind: Verschiebungen d_y in y-Richtung an drei Punkten an der hinteren Kante der Struktur (BiCGSTAB, Jacobi(4), 0,3). 162
6.12 Fahne im Wind: Momentaufnahmen des Verformungszustands des Struktur mit Farbkontur der Veschiebungkomponente d_y (Verformung 5-fach überhöht). 163

6.13 Fahne im Wind: Verschiebungen d_y in y-Richtung an drei Punkten an der hinteren Kante der Struktur (GMRES(120), BILU). 164

Tabellenverzeichnis

3.1 Eigenschaften der sechs Implementierungen und deren relative Zeit für die Berechnung repräsentativer Matrixanteile auf drei verschiedenen Hardware-Architekturen. 61
3.2 Effizienz der ursprünglichen (Original) und der vektorisierten Implementierung (Vektor 5) in Prozent der theoretisch erreichbaren Maximalleistung. 64

4.1 Splitting-assoziierte Vorkonditionierer. 82
4.2 „Optimale“ Löserparameter für die sechs untersuchten Kombinationen von Krylow-Verfahren und Vorkonditionierer. 93
4.3 Einfluss der Diskretisierungsparameter auf Rechenzeit und Konvergenzgeschwindigkeit des linearen Lösers (Rechenzeit [s] und Krylow-Iterationen). 95
4.4 Einfluss der Hardware-Architektur auf Rechenzeit und Iterationszahl des linearen Lösers (Rechenzeit [s] und Krylow-Iterationen). 97

5.1 Diskretisierungen des Demonstrationsbeispiels. 111
5.2 Vergleich der benötigten Iterationen und Rechenzeiten für verschiedene Relaxationsverfahren. 134
5.3 Durchschnittlich benötige Anzahl an Iterationen und Rechenzeit für einen Zeitschritt mit dem Block-Gauß-Seidel-Verfahren und Aitken-Relaxation. 143
5.4 Durchschnittlich benötige Anzahl an Iterationen und Rechenzeit für einen Zeitschritt mit dem Block-Gauß-Seidel-Verfahren, Aitken-Relaxation und Grob-Gitter-Prädiktor. 144
5.5 Durchschnittlich benötige Anzahl an Iterationen und Rechenzeit für einen Zeitschritt mit dem Newton-Krylow-Verfahren mit Finiter Differenz. . . . 146

6.1 Gebäude mit starrem Dach: Diskretisierungen. 149
6.2 Gebäude mit flexiblem Dach: Diskretisierungen. 153
6.3 Fahne im Wind: Diskretisierungen. 159

Abkürzungen und Bezeichnungen

Abkürzungen

ALE	Arbitrary Lagrangean Eulerian
BiCG	Bi-Conjugate Gradient
BiCGSTAB ..	Bi-Conjugate Gradient stabilized
BLIS	Block-based Linear Iterative Solver
CG	Conjugate Gradients
CGP	Coarse Grid Predictor
CGS	Conjugate Gradient Squared
FEM	Finite-Element-Methode
FLOPS	Floating Point Operations per Second
GCL	Geometric Conservation Laws
GLS	Galerkin/Least-Squares
GMRES	Generalized Minimal Residual Method
HLRS	Höchstleistungsrechenzentrum Stuttgart
HPC	High Performance Computing
ILU	Incomplete LU-Decomposition
LBB	Ladyschenskaja-Babuška-Brezzi
NURBS	Non-Uniform Rational B-Spline
PK1	Erster Piola-Kirchhoff-Spannungstensor
PK2	Zweiter Piola-Kirchhoff-Spannungstensor
SGS	symmetrisches Gauß-Seidel-Verfahren
SIMD	Single Instruction, Multiple Data
SOR	Successive Overrelaxation

SUPG/PSPG	Streamline Upwind Petrow-Galerkin/Pressure Stabilized Petrow-Galerkin
USFEM	Unusual Stabilized Finite Element Method

Mathematische Notationen

$\bigcup$	Assemblierungs-Operator
$(\bullet)_{\mathrm{C}}$	Größe bezogen auf die Kontinuitätsgleichung des Fluids
$\dot{(\bullet)}$	Zeitableitung einer Größe
$\hat{(\bullet)}$	vorgeschriebene Größe
$(\bullet)_{\mathrm{e}}$	Größe bezogen auf ein Element
$(\bullet)_{\mathrm{F}}$	Größe bezogen auf das Fluid
$(\bullet)_{\Gamma}$	Größe bezogen auf den Kopplungsrand
$(\bullet)^{\mathrm{h}}$	diskretisierte Größe
$(\bullet)_{\mathrm{I}}$	Größe bezogen auf ein Gebiet ohne Kopplungsrand
$(\bullet)^{(k)}$	Größe im Iterationsschritt k
$(\bullet)_{\mathrm{M}}$	Größe bezogen auf die Impulsbilanz des Fluids
$(\bullet)_n$	Größe zum Zeitpunkt n
$(\bullet)^*$	Fixpunkt eines Iterationsverfahrens
$(\bullet)_{\mathrm{S}}$	Größe bezogen auf die Struktur
$(\bullet)_{\mathrm{x}}$	kontinuierliche Größe
$(\bullet)_{\mathrm{P}}$	Prädiktorwert einer Größe
$(\bullet)'$	Größe bezogen auf die grobe Diskretisierung
$\Delta(\bullet)$	Inkrement einer Größe
$\delta(\bullet)$	virtuelle Größe
$\nabla_X \cdot$	materieller Divergenz-Operator
$diag(\bullet)$	Diagonalmatrix von $(\bullet)$
∇_X	materieller Gradienten-Operator
∇_x	räumlicher Gradienten-Operator
$\mathrm{tr}(\bullet)$	Spur einer Matrix
$(\bullet,\bullet)_{\Omega_{\mathrm{F}}}$	L^2-inneres Produkt ausgewertet über dem Gebiet Ω_{F}
$(\bullet)\,\vert_x$	ausgewertet bei festgehaltener räumlicher Koordinate $\mathbf{x}$
$(\bullet)' \xrightarrow{\mathrm{P}} (\bullet)$	Prolongation
$(\bullet) \xrightarrow{\mathrm{R}} (\bullet)'$	Restriktion

Lateinische Buchstaben

$\mathbf{A}$	Systemmatrix eines linearen Gleichungssystems
$\mathbf{b}$	rechte Seite eines linearen Gleichungssystems
$\hat{\mathbf{b}}$	Volumenkraftvektor

B_{gal}	Bilinearform der Galerkin-Formulierung
$B_{i,n}$	Bernstein-Polynome
$\mathbb{C}$	Materialtensor
$\mathbf{C}$	Korrektormatrix eines linearen Iterationsverfahrens
$\mathbf{c}_{\text{x}}$	konvektive Geschwindigkeit
$C(s)$	parametrische Beschreibung einer Bézierkurve/eines B-Splines
$\mathbf{d}$	Vektor der Knotenverschiebungen
$\mathbf{D}_{\text{A}}$	Diagonalmatrix zur Matrix $\mathbf{A}$
$\mathbf{d}_{\text{x}}$	kontinuierliches Verschiebungsfeld
$\mathrm{d}\mathbf{X}$	differenzielles Linienelement der Referenzkonfiguration
$\mathrm{d}\mathbf{x}$	differenzielles Linienelement der Momentankonfiguration
d	Dicke einer Struktur
$\mathbf{E}$	Green-Lagrange-Verzerrungstensor
$\mathbf{E}_{\text{A}}$	strikte untere Dreiecksmatrix zur Matrix $\mathbf{A}$
$\mathbf{e}_{1-3}$	orthonormierte, kartesische Ortsvektoren
E_{S}	Elastizitätsmodul der Struktur
$\mathbf{f}$	diskrete rechte Seite der Impulsbilanz des Fluids
$\mathbf{F}$	Deformationsgradient
$\mathbf{f}^{\text{ext}}$	Vektor der äußeren Kräfte
$\mathbf{f}^{\text{int}}$	Vektor der inneren Kräfte
$\mathbf{F}_{\text{A}}$	Fehlermatrix der unvollständigen LU-Zerlegung der Matrix $\mathbf{A}$
$\mathbf{F}_{\text{A}}$	strikte obere Dreiecksmatrix zur Matrix $\mathbf{A}$
$\mathcal{F}$	Fluid-Operator
F	Bilinearform der rechten Seite des Fluids
$\mathbf{G}_{\text{M}}$	semidiskreter stabilisierter Divergenz-Operator
$\mathbb{H}^1(\Omega)$	Sobolew-Raum von quadratintegrierbaren Funktionen mit quadratintegrierbaren ersten Ableitungen
$\hat{\mathbf{h}}$	Randlasten des Fluids
h_e	charakteristische Elementlänge
$\mathbf{I}$	Einheitstensor zweiter Stufe
$\mathbf{K}_{\text{C}}$	semidiskreter stabilisierter Gradienten-Operator
$\mathbf{K}_{\text{M}}$	Fluid-Koeffizientenmatrix der viskosen Anteile
$\mathbf{K}_{\text{N}}$	Steifigkeitsmatrix der Pseudo-Struktur
$\mathbf{K}_t(\mathbf{d}_{n+1}^{(k)})$	tangentiale Steifigkeitsmatrix
κ	spektrale Konditionszahl
K_{k}	k-dimensionaler Unterraum des $\mathbb{R}^n$
$\mathbb{L}^2(\Omega)$	Sobolew-Raum von quadratintegrierbaren Funktionen
$\mathbf{L}_{\text{A}}$	linke untere Dreiecksmatrix zur Matrix Γ
$\mathcal{L}$	Interface-Kompatibilitäts-Operator
$\mathcal{L}'(\mathbf{d}_{\Gamma}^{(k)})$	Tangente des Interface-Kompatibilitäts-Operators

$\mathcal{L}^{\text{stab}}$	Stabilisierungsoperator
L_{k}	k-dimensionaler Unterraum des $\mathbb{R}^n$
$\mathbf{M}$	Iterationsmatrix eines linearen Iterationsverfahrens
$\mathbf{M}_{\text{C}}$	Massenmatrix der Massenbilanz des Fluids
$\mathbf{M}_{\text{M}}$	Massenmatrix der Impulsbilanz des Fluids
$\mathbf{M}_{\text{S}}$	Massenmatrix der Struktur
m	Größe des Krylow-Unterraums beim GMRES
$m_{\max}$	maximale Größe des Krylow-Unterraums beim GMRES mit Neustart
$\mathbf{N}$	Ansatzfunktion
$\mathbf{N}_{\text{M}}$	Fluid-Koeffizientenmatrix der konvektiven Anteile
$\mathbf{n}_X$	Normalenvektor in der Referenzkonfiguration
$\mathcal{N}$	Netz-Operator
n_{ele}	Anzahl der Elemente
n_{t}	Anzahl der Zeitschritte
n_{eq}	Anzahl der Freiheitsgrade/Gleichungen
$N_{i,n}$	B-Spline-Basisfunktionen
nnz	Anzahl der Matrixeinträge ungleich null
$\bar{p}_{\text{x}}$	hydrostatischer Druck im Fluid
$\mathbb{P}$	Raum der Lösungs- und Testfunktionen des Fluid-Drucks
$\mathbf{P}$	Erster Piola-Kirchhoff-Spannungstensor
$\mathbf{p}^{(k)}$	Suchrichtung in der Iteration k des CG-Verfahrens
$\mathbf{P}_{\text{L}}$	linker Vorkonditionierer
$\mathbf{P}_{\text{R}}$	rechter Vorkonditionierer
$\mathcal{P}$	Prädiktor-Operator
p	Knotenwerte des Drucks
p_{x}	kinematischer Druck
P_i	Kontrollpunkte für Bézierkurven und B-Splines
q_{x}	Testfunktionen für die Massenbilanz des Fluids
$\mathbf{r}$	Residuenvektor
$\mathbf{r}_{\text{C}}$	semidiskrete rechte Seite der Massenbilanz des Fluids
$\mathbf{r}_{\text{M,b}}$,$\mathbf{r}_{\text{M,h}}$	semidiskrete rechte Seite der Impulsbilanz des Fluids
$\mathcal{R}$	Residuum
$R_{i,n}$	rationale B-Spline-Basisfunktionen
$\mathbf{S}$	Zweiter Piola-Kirchhoff-Spannungstensor
$\mathbf{S}_{\text{B}}$	Knotenvektor für B-Splines
$\mathbf{S}_{\text{S}}, \mathbf{S}_{\text{F}}$	Schurkomplement-Matrizen für Struktur und Fluid
$\mathcal{S}$	Struktur-Operator
s	Parameter für Bézierkurven und B-Splines
$\mathbf{t}$	Spannungsvektor
$\mathcal{T}$	Fluid-Struktur-Kopplungs-Abbildung

Δt	Zeitschrittweite
$\hat{\mathbf{t}}$	vorgegebener Spannungsvektor auf einen Neumann Rand Γ_t
T	Endzeitpunkt
t	Zeit
t_0	Ausgangszeitpunkt
$\mathbf{u}$	Knotenwerte der Geschwindigkeiten
$\mathbf{U}_\mathrm{A}$	rechte obere Dreiecksmatrix zur Matrix Γ
$\mathbf{u}_\mathrm{x}$	kontinuierliches Geschwindigkeitsfeld
$\mathbf{u}_\mathrm{x}^\mathrm{G}$	kontinuierliche Netzgeschwindigkeit
$\mathbb{V}$	Raum der Lösungsfunktionen der Fluid-Geschwindigkeit
$\mathbb{V}_0$	Raum der Testfunktionen für die Fluid-Geschwindigkeit
$\mathbf{v}_\mathrm{x}$	Testfunktionen für die Impulsbilanz des Fluids
W_int	Verzerrungs- oder Formänderungsarbeit
w_i	Gewichte der Kontrollpunkte eines NURBS
$\mathbf{X}$	materieller Ortsvektor
$\mathbf{x}$	räumlicher Ortsvektor

Griechische Buchstaben

α	Parameter zur Wahl verschiedener Stabilisierungsverfahren
$\alpha^{(k)}$	Schrittweite in der Iteration k beim CG-Verfahren
α_m, α_f	Shift-Parameter beim Generalisierten-α-Verfahren
α_j	Koeffizienten zur Bestimmung der orthogonalen Suchrichtungen beim CG-Verfahren
β, γ	Newmark-Parameter
χ	Ortsvektor im ALE-Referenzgebiet
δ,δ_θ,δ_BDF2	Parameter aus dem Zeitintegrationsverfahren
$\boldsymbol{\varepsilon}$	Deformationsgeschwindigkeit
Γ	Kopplungsrand zwischen Fluid und Struktur
Γ_d	Dirichlet-Rand der Struktur
Γ_F	Rand des Fluidgebiets
Γ_h	Neumann-Rand des Fluids
Γ_N	Rand des Netzgebiets
Γ_S	Rand des Strukturgebiets
Γ_t	Neumann-Rand der Struktur
Γ_u	Dirichlet-Rand des Fluids
κ	Schrittweite einer Finiten Differenz
λ_S, μ_S	Lamé-Konstanten
$\mu^{(k)}$	Aitken-Faktor
μ_F	dynamische Viskosität

ν_F	kinematische Viskosität
ν_S	Poisson- oder Querdehnzahl
ω	Relaxationsparameter eines Iterationsverfahrens
Ω_χ	Referenzgebiet in der ALE-Formulierung
Ω_e	Gebiet eines Finiten Elements
Ω_F	Fluidgebiet
Ω_N	Netzgebiet
Ω_S	Strukturgebiet
Ω_X	Materialgebiet
Ω_x	Raumgebiet
$\delta\Pi$	virtuelle Arbeit
$\delta\Pi^\mathrm{dyn}$	virtuelle Arbeit der Trägheitskräfte
$\delta\Pi^\mathrm{ext}$	virtuelle äußere Arbeit
$\delta\Pi^\mathrm{int}$	virtuelle innere Arbeit
$\Phi^{(k)}$	Abbildungsvorschrift eines Iterationsverfahrens
ρ_S	Dichte der Struktur
ρ_∞	Spektralradius
$\boldsymbol{\sigma}_\mathrm{F}$	Cauchy-Spannungstensor im Fluid
$\sigma_\mathrm{n}(\mathbf{A})$	Eigenwerte der Matrix $\mathbf{A}$
$\boldsymbol{\tau}$	viskoser Schubspannungstensor im Fluid
$\tau_{\mathrm{C}e}$	Stabilisierungsparameter für die Massenbilanz des Fluids
$\tau_{\mathrm{M}e}$	Stabilisierungsparameter für die Impulsbilanz des Fluids
θ	Parameter des Einschritt-θ-Verfahrens
ζ	Kontraktionszahl eines Iterationsverfahrens

1

Einleitung

1.1 Motivation und Zielsetzung

Fluid-Struktur-Wechselwirkungen spielen in vielen Bereichen des täglichen Lebens und der Ingenieurwissenschaften eine wichtige Rolle. So basieren z.B. die Atmung und der Blutkreislauf auf der Interaktion von Strömungen mit flexiblen Strukturen. Im technischen Bereich sind die Wechselwirkungen zwischen strömenden Fluiden und flexiblen Strukturen beispielsweise in der Luftfahrt von großer Bedeutung. So können sich die Flügelspitzen großer Verkehrsflugzeuge in Turbulenzen über mehrere Meter auf und ab bewegen. Auch im Bereich des Bauwesens gibt es Problemstellungen, bei denen die Fluid-Struktur-Wechselwirkung einen entscheidenden Einfluss auf die Bemessung eines Tragwerks hat. Beispiele hierfür sind das Schwappen in flüssigkeitsgefüllten Behältern unter dynamischer Anregung, etwa aufgrund von Erdbeben, oder windinduzierte Schwingungen von weit gespannten Brücken, Kühlturmschalen oder Membrandächern.

Diese Problemstellungen aus dem Bereich der Fluid-Struktur-Wechselwirkung (auch Fluid-Struktur-Interaktion, FSI) gehören zur Klasse der oberflächengekoppelten Mehrfeld-Probleme und sind dadurch charakterisiert, dass in mehreren Gebieten mithilfe abhängiger Variablen unterschiedliche physikalische Phänomene beschrieben werden. Streng genommen trifft diese Definition auf fast alle Ingenieuraufgaben zu, da in der Realität immer mehrere unterschiedliche physikalische Phänomene gleichzeitig auftreten und miteinander in Interaktion stehen. Jedoch sind die Wechselwirkungen zwischen den physikalischen Feldern in den meisten Fällen so gering, dass sie in der Praxis bei numerischen Simulationen häufig vernachlässigt werden.

Durch die rasant wachsenden Computerressourcen wurde es jedoch möglich mehr Wechselwirkungen bei der numerischen Simulation zu berücksichtigen und immer detaillierte-

re und damit komplexere physikalische Zusammenhänge zu analysieren. So werden z.B. chemo-hygro-mechanisch gekoppelte Simulationen zur Analyse der Alkali-Kieselsäure-Reaktion in Beton (Bangert u. a. 2004) durchgeführt oder die elektro-mechanische Kopplung bei piezoelektrischen Bauteilen (Schulz u. a. 2008) simuliert. Jedoch stoßen auch heute noch vielfach numerische Simulationen von komplexen, gekoppelten Systemen an die Grenzen der verfügbaren Computerhardware.

Bei einer partitionierten Lösung von gekoppelten Problemen, wie sie in dieser Arbeit verwendet wird, ist es möglich auf vorhandene und bewährte Lösungsalgorithmen für die einzelnen physikalischen Felder zurückzugreifen. Im Rahmen dieser Arbeit ist dies für das Fluid ein stabilisiertes Finite-Element-Verfahren zur Lösung der instationären, inkompressiblen Navier-Stokes-Gleichungen (Wall 1999; Förster 2007). Die Struktur wird mithilfe verschiedener Finite-Element-Formulierungen für die nichtlineare Elastodynamik dünnwandiger Strukturen diskretisiert (Büchter und Ramm 1992; Bischoff 1999).

Insbesondere für die Lösung der strömungsmechanischen Gleichungen werden aufgrund der vorhandenen Skalenunterschiede und der transienten Natur von Strömungen feine Diskretisierungen mit sehr vielen Freiheitsgraden und Zeitschritten benötigt. Auch der Übergang von zweidimensionalen Modellen zu realitätsnahen dreidimensionalen Simulationen bedeutet eine erheblich längere Rechenzeit. Daher zählt in vielen Fällen die numerische Simulation von dreidimensionalen reinen Strömungsproblemen mit ihren hohen Anforderungen an Rechenleistung und Speicherkapazität zum Hochleistungsrechnen. Die Implementierung von Simulationsverfahren, wie z.B. der Finite-Element-Methode, erfordert hier besondere Aufmerksamkeit, um die vorhandene Hardware effizient zu nutzen.

Ein Ziel dieser Arbeit ist es, die numerische Simulation von dreidimensionalen Fragestellungen aus dem Bereich der Strömungsmechanik auf einem parallelen Vektorcomputer zu ermöglichen. Es werden dazu Techniken vorgestellt, mit denen einerseits eine entsprechend feine Diskretisierung von komplexen Problemstellungen erzeugt und andererseits die Methode der Finiten Elemente für einen Vektorcomputer effizient implementiert werden kann. Untersuchungen zur effektiven Lösung der resultierenden linearen Gleichungssysteme mit vorkonditionierten Krylow-Verfahren runden diesen Themenkomplex ab.

Neben den Anforderungen, die bereits die dreidimensionale Simulation des Strömungsgebiets an die Computerressourcen stellt, bedeutet die Berücksichtigung einer impliziten Kopplung einen erheblich größeren Rechenaufwand. So ist die Lösung von Problemen aus dem Bereich der Fluid-Struktur-Wechselwirkung aufgrund der gegenseitigen Beeinflussung der beiden Felder deutlich komplexer als bei den Einzelfeld-Problemen:

$$\text{FSI} \quad > \quad \text{Fluid} \quad + \quad \text{Struktur}$$

Der gleiche Zusammenhang gilt auch für die Rechenzeiten bei den entsprechenden numerischen Untersuchungen, d.h. eine gekoppelte Fluid-Struktur-Wechselwirkungs-Simulation benötigt ein Vielfaches der Zeit, die die ungekoppelten Simulationen der beiden Einzelfelder in Anspruch nehmen. Es ist Ziel von vielfältigen Ansätzen, die aktuell erforscht werden, den Zeitaufwand für eine gekoppelte Simulation zu reduzieren.

In dieser Arbeit werden zunächst verschiedene etablierte Lösungsverfahren zur impliziten Behandlung der Kopplung in einer einheitlichen Herleitung und Schreibweise dargestellt. Darauf aufbauend werden zwei Zwei-Level-Verfahren zur Lösung des Fluid-Struktur-Wechselwirkungs-Problems vorgestellt, die eine Lösung des gekoppelten Problems auf einer groben Diskretisierung nutzen und so die benötigte Rechenzeit deutlich reduzieren.

1.2 Gliederung der Arbeit

In Kapitel 2 werden zunächst für Struktur und Fluid die der Modellierung zugrunde liegenden kontinuumsmechanischen Grundgleichungen eingeführt. Es folgt die Formulierung der zugehörigen Anfangs-Randwert-Probleme in schwacher Form und die darauf aufbauende Diskretisierung in Raum und Zeit.

Das 3. Kapitel beschäftigt sich mit der Strömungssimulation durch Hochleistungsrechnen. Nach einer Einführung in die Grundbegriffe des Hochleistungsrechnens wird ein Verfahren zur Erzeugung von Netzen in zwei Schritten und ein Konzept zur effizienten Implementierung von Finiten Elementen für Vektorcomputer vorgestellt.

Mit einer Übersicht über iterative Verfahren und Vorkonditionierer zur Lösung von linearen Gleichungssystemen beginnt das 4. Kapitel. Im Anschluss wird die Effizienz der vorgestellten Verfahren bei der Anwendung auf Diskretisierungen von Strömungsproblemen verglichen.

Kapitel 5 beginnt mit einem Überblick über Verfahren zur Lösung von gekoppelten Problemen. Im Anschluss werden partitionierte Verfahren ausführlicher beschrieben und zwei neue Zwei-Level-Verfahren zur Lösung von gekoppelten Problemen vorgestellt. Abschließend werden verschiedene Kopplungsverfahren hinsichtlich ihrer Konvergenzgeschwindigkeit und Rechenzeit verglichen.

Kapitel 6 dokumentiert numerische Beispiele aus dem Bereich der Fluid-Struktur-Wechselwirkung, die die Leistungsfähigkeit der vorgestellten Ansätze verdeutlichen.

2

Grundlagen

Zur numerischen Simulation von physikalischen Problemstellungen ist es erforderlich diese durch geeignete mechanisch-mathematische Modelle abzubilden. Die vorliegende Arbeit behandelt Problemstellungen der nichtlinearen Strukturdynamik in Verbindung mit der Dynamik inkompressibler, viskoser Strömungen. Zunächst werden in den folgenden Abschnitten für diese beiden mechanischen Modelle die zugrunde liegenden Differenzialgleichungen und im Anschluss daran deren numerische Behandlung gezeigt. Diese Beschreibung stellt keine vollständige Herleitung der verwendeten Gleichungen dar, sondern nur die Grundlagen für diese Arbeit in einer konformen Notation.

Eine tiefergehende Herleitung der kontinuumsmechanischen Grundlagen und Diskretisierungsverfahren ist in vorhergehenden Dissertationen am Institut für Baustatik und Baudynamik, wie z.B. Wall (1999), Mok (2001), Förster (2007), und einschlägigen Fachbüchern zu finden. Ausführlichere Darstellungen zur Kontinuumsmechanik finden sich unter anderem in Altenbach und Altenbach (1994), Holzapfel (2000), Marsden und Hughes (1983) und Stein und Barthold (1996) und zur Fluiddynamik in Cuvelier u. a. (1986), Ferziger und Peric (1997), Fletcher (1991), Gresho und Sani (1998), Gunzburger (1989) und Warsi (1993).

Die grundlegenden Ideen der in dieser Arbeit als Diskretisierungsverfahren verwendeten Finite-Element-Methode gehen zurück auf Argyris (1954), Turner u. a. (1956), Clough (1960) und Zienkiewicz und Cheung (1967). Zu den wichtigsten aktuellen Lehrbüchern gehören unter anderem Zienkiewicz u. a. (2005), Hughes (2000), Belytschko u. a. (2000) und Wriggers (2001).

2.1 Strukturdynamik

Die in dieser Arbeit hauptsächlich betrachteten Problemstellungen der Fluid-Struktur-Wechselwirkung bei dünnwandigen Strukturen erfordern die Berücksichtigung großer Deformationen im Strukturmodell. Aufgrund der Schlankheit der Struktur wird davon ausgegangen, dass nur kleine Verzerrungen auftreten. Zusätzlich werden ausschließlich homogene, isotrope Kontinua betrachtet. Als passendes Strukturmodell wird die geometrisch nichtlineare Elastodynamik gewählt.

Für die Beschreibung der Bewegung der Struktur im Raum wird die Lagrange[1]- oder auch materielle Betrachtungsweise verwendet. Hierbei bewegt sich der Beobachter mit einem Materiepunkt durch den Raum und misst zu jedem Zeitpunkt die Lage des Materiepunkts und dessen Eigenschaften in der Momentankonfiguration. Die deformierte Momentankonfiguration (Raumgebiet Ω_x) wird dabei in Bezug auf eine Referenzkonfiguration (Materialgebiet Ω_X) beschrieben. Wird, wie in dieser Arbeit, als Referenzkonfiguration der Ausgangszustand der Struktur zur Zeit $t = t_0$ gewählt, wird von der totalen Lagrange-Formulierung gesprochen.

Im Folgenden bezeichnet der Index $(\cdot)_\mathrm{S}$ Größen, die sich auf die Struktur beziehen.

2.1.1 Kontinuumsmechanische Grundgleichungen

Kinematik

Die Lagen der Materiepunkte in der Referenz- und Momentankonfiguration werden durch ihre Ortsvektoren $\mathbf{X}$ bzw. $\mathbf{x}$ in einem raumfesten, orthonormierten, kartesischen Koordinatensystem $\mathbf{e}$ beschrieben (Abbildung 2.1). Hierbei werden alle Größen, die sich auf die Referenzkonfiguration beziehen, durch einen Großbuchstaben gekennzeichnet. Die Differenz zwischen den Ortsvektoren der Momentan- und Referenzkonfiguration wird als Verschiebungsfeld $\mathbf{d}_\mathrm{x} = \mathbf{x} - \mathbf{X}$ bezeichnet. Der Index $(\cdot)_\mathrm{x}$ kennzeichnet dabei, im Gegensatz zum Vektor der Knotenwerte, der keinen Index trägt, die kontinuierlichen Funktionen.

Der unsymmetrische materielle Deformationsgradient $\mathbf{F}$ ist definiert als lineare Abbildung eines differenziellen Linienelements der Referenzkonfiguration $\mathrm{d}\mathbf{X}$ auf das entspre-

[1] Joseph Louis Lagrange, * 25. Januar 1736 in Turin, † 10. April 1813 in Paris, italienischer Mathematiker und Astronom.

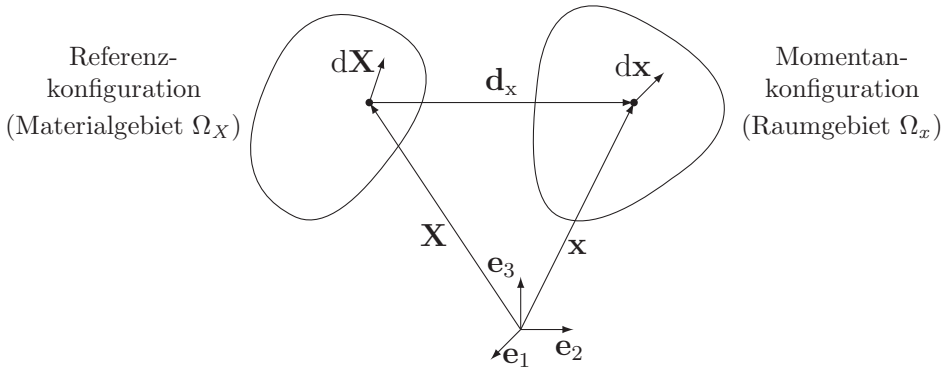


Abbildung 2.1: Referenz- und Momentankonfiguration in der Strukturdynamik.

chende Linienelement der Momentankonfiguration d$\mathbf{x}$:

$$\mathrm{d}\mathbf{x} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} \cdot \mathrm{d}\mathbf{X} =: \mathbf{F} \cdot \mathrm{d}\mathbf{X} \qquad \Rightarrow \qquad \mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} = \nabla_X \mathbf{x} = \nabla_X \mathbf{d}_\mathrm{x} + \mathbf{I}. \tag{2.1}$$

Dabei bezeichnet $\mathbf{I}$ den zweistufigen Einheitstensor und ∇_X den materiellen Gradienten-Operator. Der Deformationsgradient beschreibt den Deformationsvorgang vollständig, ist als Verzerrungsmaß jedoch ungeeignet, da er Starrkörperanteile enthält und unsymmetrisch sowie richtungsabhängig ist. Daher wird der Green[2]-Lagrange-Verzerrungstensor $\mathbf{E}$ als Verzerrungsmaß gewählt, der ein symmetrischer, richtungsunabhängiger und von den Starrkörperanteilen befreiter Tensor zweiter Stufe ist. Er ist durch die Differenz der Quadrate der Linienelemente in der Momentan- und Referenzkonfiguration definiert

$$\mathbf{E} = \frac{1}{2}\left(\mathbf{F}^\mathrm{T} \cdot \mathbf{F} - \mathbf{I}\right). \tag{2.2}$$

Unter Verwendung der Definition des Deformationsgradienten (2.1) lässt er sich auch als nichtlineare Verzerrungs-Verschiebungs-Beziehung schreiben:

$$\mathbf{E} = \frac{1}{2}\left(\nabla_X \mathbf{d}_\mathrm{x} + (\nabla_X \mathbf{d}_\mathrm{x})^\mathrm{T} + (\nabla_X \mathbf{d}_\mathrm{x})^\mathrm{T} \cdot \nabla_X \mathbf{d}_\mathrm{x}\right). \tag{2.3}$$

Der Green-Lagrange-Verzerrungstensor $\mathbf{E}$ stellt eine vollständige Beschreibung des Deformationsvorgangs dar und ist somit auch für große Deformationen geeignet. Durch seine Definition ist die kinematische Feldgleichung der Elastodynamik gegeben.

[2]George Green, * 14. Juli 1793 in Nottingham, † 31. März 1841 in Sneinton, englischer Mathematiker und Physiker.

Impulsbilanz

Die lokale Impulsbilanz oder auch Erste Cauchy[3]-Bewegungsgleichung stellt die lokale, punktweise Form des dynamischen Gleichgewichts des Körpers dar und ist die zweite Feldgleichung der Elastodynamik.

$$\rho_{\mathrm{S}} \ddot{\mathbf{d}}_{\mathrm{x}} = \nabla_X \cdot \mathbf{P} + \rho_{\mathrm{S}} \hat{\mathbf{b}}_{\mathrm{S}} \tag{2.4}$$

Hierin ist ρ_{S} die Dichte der Struktur in der undeformierten Referenzkonfiguration und $\hat{\mathbf{b}}_{\mathrm{S}}$ der Volumenkraftvektor pro Masseneinheit (z.B. aus Eigengewicht). $(\ddot{\bullet})$ bezeichnet die zweifache Zeitableitung und $\mathbf{P}$ den Ersten Piola[4]-Kirchhoff[5]-Spannungstensor (PK1), der die aktuelle Kraft auf ein Flächenelement der Referenzkonfiguration bezieht. Dessen Unsymmetrie führt jedoch zu Nachteilen bei der numerischen Umsetzung z.B. von Materialgesetzen. Daher wird er meistens, so auch in dieser Arbeit, durch den symmetrischen Zweiten Piola-Kirchhoff-Spannungstensor (PK2) $\mathbf{S}$ ersetzt. Dieser bezieht Kraft und Fläche auf die Referenzkonfiguration, ist das zum Green-Lagrange-Verzerrungstensor $\mathbf{E}$ energetisch konjugierte Spannungsmaß und steht über den Deformationsgradienten mit dem PK1-Spannungstensor in folgender Beziehung:

$$\mathbf{S} = \mathbf{F}^{-1} \cdot \mathbf{P}; \qquad \mathbf{P} = \mathbf{F} \cdot \mathbf{S}. \tag{2.5}$$

Unter Verwendung dieser Beziehung zwischen PK1 und PK2 lässt sich die materielle Formulierung der Cauchy-Bewegungsgleichung (2.4) nun wie folgt schreiben:

$$\rho_{\mathrm{S}} \ddot{\mathbf{d}}_{\mathrm{x}} = \nabla_X \cdot (\mathbf{F} \cdot \mathbf{S}) + \rho_{\mathrm{S}} \hat{\mathbf{b}}_{\mathrm{S}}. \tag{2.6}$$

Dies stellt die zweite, die dynamische Feldgleichung des Anfangs-Randwert-Problems der Elastodynamik dar.

Konstitutivgesetz

Das Konstitutivgesetz stellt die Beziehung zwischen den kinematischen und den statischen Feldgrößen her. Bei der hier gewählten Betrachtungsweise, bei der sich alle Größen auf die undeformierte Referenzkonfiguration beziehen, ist dies der Zusammenhang zwischen dem Green-Lagrange-Verzerrungstensor $\mathbf{E}$ und dem Zweiten Piola-Kirchhoff-Spannungstensor $\mathbf{S}$. Im Rahmen dieser Arbeit wird lediglich ein linear elastisches Mate-

[3] Augustin Louis Cauchy, * 21. August 1789 in Paris, † 23. Mai 1857 in Sceaux, französischer Mathematiker.
[4] Gabrio Piola, * 1794 in Mailand, † 1850 in Giussano, italienischer Mathematiker.
[5] Gustav Robert Kirchhoff, * 12. März 1824 in Königsberg, † 17. Oktober 1887 in Berlin, deutscher Physiker.

rialverhalten angenommen. Dies ist bei den hier betrachteten sehr dünnen Strukturen, in denen nur kleine Verzerrungen auftreten, für die meisten Fälle eine vertretbare Vereinfachung.

Für die somit zu betrachtende Green'sche Elastizität wird die Existenz einer spezifischen Verzerrungs- oder Formänderungsarbeit $W_{\mathrm{int}}(\mathbf{E})$ angenommen, die bezüglich des PK2 eine Potentialfunktion darstellt. Mit ihr lässt sich die konstitutive Beziehung in allgemeiner Form schreiben:

$$\mathbf{S} = \frac{\partial W_{\mathrm{int}}(\mathbf{E})}{\partial \mathbf{E}} : \mathbf{E}. \tag{2.7}$$

Die Linearisierung dieser Spannungs-Dehnungs-Beziehung und die Beschränkung auf homogene, isotrope, linear elastische Materialien führt auf die konstitutive Gleichung des Saint-Venant[6]-Kirchhoff-Materials, welche die dritte Feldgleichung darstellt:

$$\mathbf{S} = \mathbb{C} : \mathbf{E} = \lambda_{\mathrm{S}} \operatorname{tr} \mathbf{E}\, \mathbf{I} + 2\mu_{\mathrm{S}} \mathbf{E}. \tag{2.8}$$

$\mathbb{C}$ bezeichnet den vierstufigen Material- bzw. Elastizitätstensor. Die Lamé[7]-Konstanten λ_{S} und μ_{S} können aus den in der Ingenieurliteratur gebräuchlicheren Elastizitätsmaßen Elastizitätsmodul E_{S} und Poisson[8]- bzw. Querdehnzahl ν_{S} berechnet werden:

$$\lambda_{\mathrm{S}} = \frac{E_{\mathrm{S}} \nu_{\mathrm{S}}}{(1+\nu_{\mathrm{S}})\,(1-2\nu_{\mathrm{S}})}, \qquad \mu_{\mathrm{S}} = \frac{E_{\mathrm{S}}}{2\,(1+\nu_{\mathrm{S}})}. \tag{2.9}$$

Das Saint-Venant-Kirchhoff-Materialgesetz (2.8) ist aufgrund der vollständigen, nichtlinearen Dehnungsbeschreibung mittels des Green-Lagrange-Verzerrungstensors für große Deformationen geeignet, aber wegen der linearen Beziehung zwischen Spannungen und Dehnungen nur für kleine Verzerrungen anwendbar.

2.1.2 Anfangs-Randwert-Problem

Starke Form

Für die vollständige Beschreibung der starken, d.h. lokalen Form des Anfangs-Randwert-Problems der nichtlinearen Elastodynamik werden neben den drei Feldgleichungen (2.3),

[6] Adhémar Jean Claude Barré de Saint-Venant, * 23. August 1797 in Villiers-en-Bière, † 6. Januar 1886 in St. Ouen, französischer Mathematiker und Physiker.

[7] Gabriel Lamé, * 22. Juli 1795 in Tours, † 1. Mai 1870 in Paris, französischer Mathematiker und Physiker.

[8] Siméon-Denis Poisson, * 21. Juni 1781 in Pithiviers, † 25. April 1840 in Paris, französischer Physiker und Mathematiker.

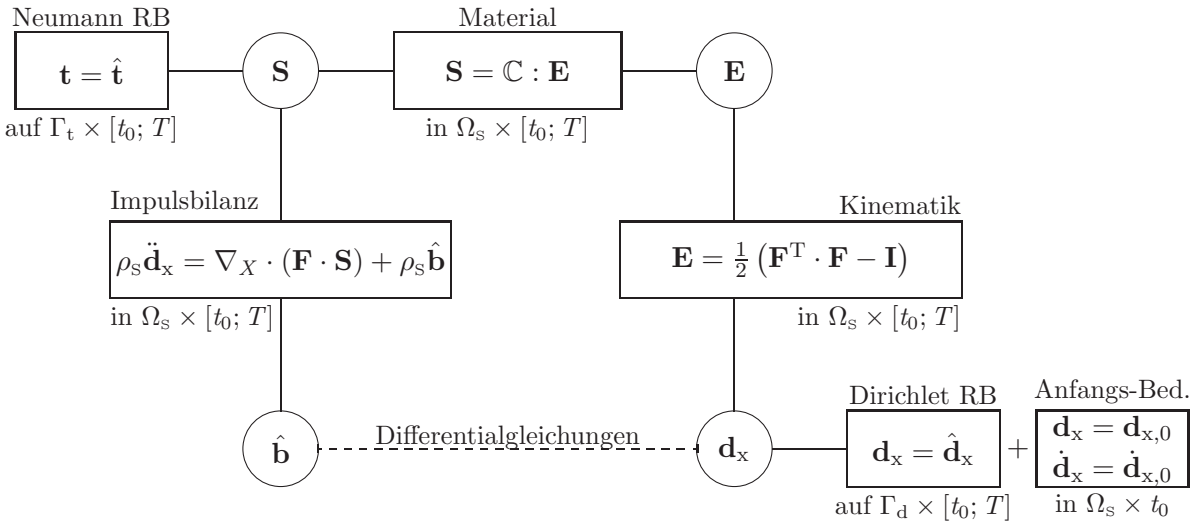


Abbildung 2.2: Tonti-Diagramm der starken Form des Anfangs-Randwert-Problems der Strukturdynamik.

(2.6) und (2.8) noch entsprechende Anfangs- und Randbedingungen benötigt. Anfangsbedingungen sind beispielsweise das Verschiebungsfeld $\mathbf{d}_{\mathrm{x},0}$ und das Geschwindigkeitsfeld $\dot{\mathbf{d}}_{\mathrm{x},0}$ auf dem gesamten Strukturgebiet Ω_s zum Anfangszeitpunkt t_0.

Der Rand $\Gamma_\mathrm{s} = \partial\Omega_\mathrm{s}$ des Strukturgebiets teilt sich in zwei disjunkte Untermengen

$$\Gamma_\mathrm{s} = \Gamma_\mathrm{d} \cup \Gamma_\mathrm{t} \quad \text{mit} \qquad \Gamma_\mathrm{d} \cap \Gamma_\mathrm{t} = 0. \tag{2.10}$$

Auf dem Teil Γ_d sind die Verschiebungen $\hat{\mathbf{d}}_\mathrm{x}$ vorgeschrieben. Diese Randbedingungen werden als geometrische, wesentliche oder auch als Dirichlet[9]-Randbedingungen bezeichnet. Die Neumann[10]-Randbedingungen (auch statisch oder natürlich genannt) schreiben auf dem Rand Γ_t den Randspannungsvektor $\hat{\mathbf{t}}$ vor. Der Spannungsvektor in der Referenzkonfiguration $\mathbf{t}$ ergibt sich aus dem Zweiten Piola-Kirchhoff-Spannungstensor und dem Normalenvektor in der Referenzkonfiguration $\mathbf{n}_X$:

$$\mathbf{t} = \mathbf{F} \cdot \mathbf{S} \cdot \mathbf{n}_X. \tag{2.11}$$

Die starke Form des Anfangs-Randwert-Problems kann in Form eines sogenannten Tonti[11]-Diagramms nach Tonti (1975), wie in Abbildung 2.2 dargestellt, zusammengefasst werden.

[9] Johann Peter Gustav Lejeune Dirichlet, * 13. Februar 1805 in Düren, † 5. Mai 1859 in Göttingen, deutscher Mathematiker.

[10] Carl Gottfried Neumann, * 7. Mai 1832 in Königsberg, † 27. März 1925 in Leipzig, deutscher Mathematiker.

[11] Enzo Tonti,* 30. Oktober 1935 in Mailand, italienischer Ingenieur und Mathematiker.

Schwache Form

Zur numerischen Lösung des Anfangs-Randwert-Problems wird die in Abbildung 2.2 zusammengefasste starke Form in eine sogenannte schwache Form überführt. Für die hier verwendete Verschiebungsformulierung werden die Verschiebungen $\mathbf{d}_\mathrm{x}$ als Primärvariable gewählt. Die Verzerrungen $\mathbf{E}$ und Spannungen $\mathbf{S}$ werden punktweise durch die kinematischen Gleichungen und das Konstitutivgesetz als Funktion der Verschiebungen beschrieben. Die schwache Form des Anfangs-Randwert-Problems wird nun mithilfe der Methode der gewichteten Residuen aufgestellt. Dazu werden die Residuen der dynamischen Feldgleichung $\mathcal{R}_\mathrm{S}$ und der statischen Randbedingungen $\mathcal{R}_\mathrm{RB}$ jeweils mit Testfunktionen $\delta\mathbf{d}_\mathrm{x}$ multipliziert und über das Gebiet Ω_S integriert.

$$\int_{\Omega_\mathrm{S}} \delta\mathbf{d}_\mathrm{x} \cdot \underbrace{\left(\rho_\mathrm{S}\ddot{\mathbf{d}}_\mathrm{x} - \nabla_X \cdot (\mathbf{F}\cdot\mathbf{S}) - \rho_\mathrm{S}\hat{\mathbf{b}}\right)}_{\mathcal{R}_\mathrm{S}} \,\mathrm{d}\Omega_\mathrm{S} + \int_{\Gamma_\mathrm{t}} \delta\mathbf{d}_\mathrm{x} \cdot \underbrace{\left(\mathbf{t}-\hat{\mathbf{t}}\right)}_{\mathcal{R}_\mathrm{RB}} \,\mathrm{d}\Gamma_\mathrm{t} = 0 \tag{2.12}$$

Dadurch werden das Gleichgewicht und die statischen Randbedingungen nur noch im integralen Sinn über das Gebiet und nicht mehr punktweise getestet. Wird die Menge der Testfunktionen $\delta\mathbf{d}_\mathrm{x}$ nicht eingeschränkt, ist diese Aussage mit der starken Form identisch. Die Anwendung der partiellen Integration (Gauß'scher[12] Integralsatz und Produktregel für die Divergenz) und das Einsetzen der punktweise erfüllten kinematischen und konstitutiven Beziehungen sowie der geometrischen Randbedingungen führt auf die folgende Darstellung der schwachen Form:

$$\underbrace{\int_{\Omega_\mathrm{S}} \delta\mathbf{d}_\mathrm{x}\cdot\ddot{\mathbf{d}}_\mathrm{x}\rho_\mathrm{S}\,\mathrm{d}\Omega_\mathrm{S}}_{\delta\Pi^\mathrm{dyn}} + \underbrace{\int_{\Omega_\mathrm{S}} \delta\mathbf{E}:\mathbf{S}\,\mathrm{d}\Omega_\mathrm{S}}_{\delta\Pi^\mathrm{int}} \underbrace{- \int_{\Gamma_\mathrm{t}} \delta\mathbf{d}_\mathrm{x}\cdot\hat{\mathbf{t}}\,\mathrm{d}\Gamma_\mathrm{S} - \int_{\Omega_\mathrm{S}} \delta\mathbf{d}_\mathrm{x}\cdot\hat{\mathbf{b}}\rho_\mathrm{S}\,\mathrm{d}\Omega_\mathrm{S}}_{-\delta\Pi^\mathrm{ext}} = 0. \tag{2.13}$$

Dieses Ergebnis entspricht dem Prinzip der virtuellen Verschiebungen, wenn die Testfunktionen $\delta\mathbf{d}_\mathrm{x}$ als virtuelle Verschiebung interpretiert werden. Das Prinzip besagt, dass die virtuelle Arbeit $\delta\Pi$, die aus der Summe der virtuellen Arbeit der Trägheitskräfte $\delta\Pi^\mathrm{dyn}$ und der virtuellen inneren Arbeit $\delta\Pi^\mathrm{int}$ abzüglich der virtuellen äußeren Arbeit $\delta\Pi^\mathrm{ext}$ besteht, im Gleichgewichtszustand null wird.

[12] Johann Carl Friedrich Gauß, * 30. April 1777 in Braunschweig, † 23. Februar 1855 in Göttingen, deutscher Mathematiker, Astronom, Geodät und Physiker.

2.1.3 Diskretisierung im Raum

Die numerische Lösung des Anfangs-Randwert-Problems der nichtlinearen Elastodynamik wird im Rahmen dieser Arbeit in einem sequenziellen Diskretisierungkonzept im Raum mittels der Finite-Element-Methode (FEM) und in der Zeit mit dem im folgenden Abschnitt vorgestellten Generalisierten-α-Verfahren vorgenommen.

Die Grundlage der Methode der Finiten Elemente bildet die Unterteilung des zu analysierenden Gebiets Ω_S in eine endliche Anzahl n_ele Elemente Ω_e:

$$\Omega_\mathrm{S} \approx \Omega_\mathrm{S}^\mathrm{h} = \bigcup_{e=1}^{n_\mathrm{ele}} \Omega_\mathrm{e}. \tag{2.14}$$

Mit dem oberen Index h wird von nun an eine diskretisierte Größe bezeichnet, wohingegen der untere Index e Größen bezeichnet, die sich auf ein Element beziehen. Der Verschiebungsverlauf in jedem Element $\mathbf{d}_\mathrm{x,e}$ wird approximativ durch polynomische Ansatzfunktionen $\mathbf{N}$ zwischen den diskreten Knotenverschiebungen des Elements $\mathbf{d}_\mathrm{e}$, den Freiheitsgraden der Verschiebungsformulierung, interpoliert. Bei einer Bubnow[13]-Galerkin[14]-Diskretisierung wird dieselbe Interpolation auch für die Testfunktionen $\delta\mathbf{d}_\mathrm{x,e}$ verwendet.

$$\mathbf{d}_\mathrm{x,e} \approx \mathbf{d}_\mathrm{x,e}^\mathrm{h} = \mathbf{N}\mathbf{d}_\mathrm{e} \qquad\qquad \delta\mathbf{d}_\mathrm{x,e} \approx \delta\mathbf{d}_\mathrm{x,e}^\mathrm{h} = \mathbf{N}\delta\mathbf{d}_\mathrm{e} \tag{2.15}$$

Die semidiskrete Darstellung der schwachen Form des Gleichgewichts folgt aus Einsetzen der Ansätze (2.15) in Gleichung (2.13).

$$\delta\Pi^\mathrm{h} = \bigcup_{e=1}^{n_\mathrm{ele}} \delta\mathbf{d}_\mathrm{e}^\mathrm{T} \left[\mathbf{M}_\mathrm{e}\ddot{\mathbf{d}}_\mathrm{e} + \mathbf{f}_\mathrm{e}^\mathrm{int}(\mathbf{d}_\mathrm{e}) - \mathbf{f}_\mathrm{e}^\mathrm{ext}\right] = 0 \tag{2.16}$$

Das Symbol $\bigcup$ steht für den Zusammenbau (Assemblierung) der elementweisen, inneren $\mathbf{f}_\mathrm{e}^\mathrm{int}(\mathbf{d}_\mathrm{e})$ und äußeren $\mathbf{f}_\mathrm{e}^\mathrm{ext}$ Kräfte sowie der Massenmatrizen $\mathbf{M}_\mathrm{e}$ zu den globalen Systemgrößen unter Berücksichtigung der kinematischen Verträglichkeit und des Knotengleichgewichts (direkte Steifigkeitsmethode). Durch Anwendung des Fundamentallemmas der Variationsrechnung kann Gleichung (2.16) in das semidiskrete Anfangs-Randwert-Problem der nichtlinearen Elastodynamik überführt werden:

$$\mathbf{M}_\mathrm{S}\ddot{\mathbf{d}} + \mathbf{f}^\mathrm{int}(\mathbf{d}) = \mathbf{f}^\mathrm{ext}. \tag{2.17}$$

[13] I.G. Bubnow, * 1872, † 1919, russischer Schiffsbau-Ingenieur.

[14] Boris Grigorjewitsch Galerkin (auch Galjorkin), * 4. März 1871 in Polozk, † 12. Juli 1945 in Leningrad, sowjetischer Ingenieur und Mathematiker.

Insbesondere niedrig interpolierte Finite Elemente, die auf dem Prinzip der virtuellen Verschiebungen basieren, weisen unphysikalische Versteifungseffekte („locking“) auf. Diese beeinträchtigen die Qualität der Strukturantwort. Einen Überblick über Methoden zur Reduzierung dieser Versteifungseffekte mit Verweisen auf weiterführende Literatur befindet sich z.B. in Hartmann (2007).

Für die Diskretisierung der in dieser Arbeit betrachteten dünnwandigen Strukturen werden häufig volumenorientierte Schalenelemente verwendet. Die Berücksichtigung des dreidimensionalen Schalenkörpers bringt im Gegensatz zu einer Schalenformulierung mit Dimensionsreduzierung Vorteile insbesondere bei der Diskretisierung von dünnen Strukturen, die auf beiden Seiten umströmt werden. Verwendung findet eine 7-Parameter-Schalenformulierung, die auf Büchter und Ramm (1992) und Büchter u. a. (1994) zurückgeht und u.a. von Bischoff (1999) weiterentwickelt wurde. Aktuelle Beschreibungen dieser Formulierung finden sich z.B. in Gee (2004) und Hartmann (2007).

2.1.4 Diskretisierung in der Zeit

Die numerische Integration der semidiskreten Bewegungsgleichung (2.17) erfordert die Diskretisierung in der Zeit. Dazu wird der betrachtete Zeitraum $[t_0; T]$ in n_t Zeitintervalle $[t_n; t_{n+1}]$ der Größe Δt unterteilt. Diese werden der Einfachheit halber als gleich groß angenommen. Sind zu Beginn eines Zeitschritts die diskreten Zustandsgrößen $\mathbf{d}_n$, $\dot{\mathbf{d}}_n$ und $\ddot{\mathbf{d}}_n$ bekannt, wird mithilfe der zeitlichen Diskretisierung die semidiskrete Form der Bewegungsgleichung (2.17) für die unbekannten Zustandsvektoren am Ende des Zeitintervalls $\mathbf{d}_{n+1}$, $\dot{\mathbf{d}}_{n+1}$ und $\ddot{\mathbf{d}}_{n+1}$ in impliziter Form gelöst. Dazu werden, ausgehend von den Zeitintegrationsansätzen von Newmark (1959), die Geschwindigkeiten und Beschleunigungen zum Zeitpunkt t_{n+1} in Abhängigkeit der unbekannten Verschiebungen $\mathbf{d}_{n+1}$ ausgedrückt:

$$\begin{aligned}
\dot{\mathbf{d}}_{n+1}(\mathbf{d}_{n+1}) &= \frac{\gamma}{\beta\,\Delta t}(\mathbf{d}_{n+1} - \mathbf{d}_n) - \left(\frac{\gamma}{\beta} - 1\right)\dot{\mathbf{d}}_n - \left(\frac{\gamma}{2\beta} - 1\right)\Delta t\;\ddot{\mathbf{d}}_n \\
\ddot{\mathbf{d}}_{n+1}(\mathbf{d}_{n+1}) &= \frac{1}{\beta\,\Delta t^2}(\mathbf{d}_{n+1} - \mathbf{d}_n) - \frac{1}{\beta\,\Delta t}\dot{\mathbf{d}}_n - \left(\frac{1}{2\beta} - 1\right)\ddot{\mathbf{d}}_n.
\end{aligned} \tag{2.18}$$

Die Konstanten β und $\gamma \in \mathbb{R}_0^+$ werden als Newmark[15]-Parameter bezeichnet und ermöglichen die Konstruktion einer Vielzahl von Zeitintegrationsverfahren mit unterschiedlichen numerischen Eigenschaften.

Um gleichzeitig numerische Dissipation und eine Genauigkeit zweiter Ordnung zu gewährleisten, wird beim Generalisierten-α-Verfahren (Chung und Hulbert 1993) die

[15]Nathan M. Newmark, * 22. September 1910 in Plainfield (New Jersey), † 25. Januar 1981, US-amerikanischer Bauingenieur.

nichtlineare, semidiskrete Bewegungsgleichung (2.17) modifiziert und an einem generalisierten Mittelpunkt ausgewertet, der durch die beiden Shift-Parameter α_m und α_f charakterisiert wird:

$$\mathbf{M}_\mathrm{S}\ddot{\mathbf{d}}_{n+1-\alpha_\mathrm{m}} + \mathbf{f}^\mathrm{int}(\mathbf{d}_{n+1-\alpha_\mathrm{f}}) = \mathbf{f}^\mathrm{ext}_{n+1-\alpha_\mathrm{f}}. \tag{2.19}$$

Die Zustandsgrößen zu diesem Zeitpunkt werden durch eine Linearkombination der entsprechenden Größen an den Intervallgrenzen t_n und t_{n+1} ausgedrückt:

$$\begin{aligned}
\ddot{\mathbf{d}}_{n+1-\alpha_\mathrm{m}} &= (1-\alpha_\mathrm{m})\,\ddot{\mathbf{d}}_{n+1} + \alpha_\mathrm{m}\,\ddot{\mathbf{d}}_n \\
\dot{\mathbf{d}}_{n+1-\alpha_\mathrm{f}} &= (1-\alpha_\mathrm{f}\,)\,\dot{\mathbf{d}}_{n+1} + \alpha_\mathrm{f}\,\dot{\mathbf{d}}_n \\
\mathbf{d}_{n+1-\alpha_\mathrm{f}} &= (1-\alpha_\mathrm{f}\,)\,\mathbf{d}_{n+1} + \alpha_\mathrm{f}\,\mathbf{d}_n \\
\mathbf{f}^\mathrm{ext}_{n+1-\alpha_\mathrm{f}} &= (1-\alpha_\mathrm{f}\,)\,\mathbf{f}^\mathrm{ext}_{n+1} + \alpha_\mathrm{f}\,\mathbf{f}^\mathrm{ext}_n.
\end{aligned} \tag{2.20}$$

Die Bestimmung der inneren Kräfte kann dabei entweder direkt über die generalisierten Mittelpunktsverschiebungen erfolgen

$$\mathbf{f}^\mathrm{int}(\mathbf{d}_{n+1-\alpha_\mathrm{f}}) = \mathbf{f}^\mathrm{int}\left((1-\alpha_\mathrm{f})\,\mathbf{d}_{n+1} + \alpha_\mathrm{f}\mathbf{d}_n\right) \tag{2.21}$$

oder entsprechend den Ansätzen für die kinematischen Größen mithilfe einer Linearkombination aus den inneren Kräften an den Intervallgrenzen.

$$\mathbf{f}^\mathrm{int}(\mathbf{d}_{n+1-\alpha_\mathrm{f}}) = (1-\alpha_\mathrm{f})\,\mathbf{f}^\mathrm{int}(\mathbf{d}_{n+1}) + \alpha_\mathrm{f}\mathbf{f}^\mathrm{int}(\mathbf{d}_n) \tag{2.22}$$

Für den linearen Fall führen beide Ansätze zu äquivalenten Ergebnissen, wohingegen im nichtlinearen Fall unterschiedliche Auswirkungen auf die algorithmische Umsetzung folgen (Kuhl 1996; Kuhl und Crisfield 1999). In dieser Arbeit werden die inneren Kräfte durch Interpolation gemäß Gleichung (2.22) bestimmt, wodurch sich der große Vorteil ergibt, dass keine Modifikationen in den Elementroutinen erforderlich sind.

Mithilfe der beiden Newmark-Parameter β und γ sowie der beiden Shift-Parameter α_f und α_m kann das Maß an numerischer Dissipation des Zeitintegrationsalgorithmus gezielt gesteuert werden. Dabei lassen sich die vier Parameter in Abhängigkeit des sogenannten Spektralradius $\rho_\infty \in [0;1]$ bestimmen (Kuhl 1996; Mok 2001):

$$\alpha_\mathrm{m} = \frac{2\rho_\infty - 1}{\rho_\infty + 1};\quad \alpha_\mathrm{f} = \frac{\rho_\infty}{\rho_\infty + 1};\quad \beta = \frac{1}{4}\left(1 - \alpha_\mathrm{m} + \alpha_\mathrm{f}\right)^2;\quad \gamma = \frac{1}{2} - \alpha_\mathrm{m} + \alpha_\mathrm{f}. \tag{2.23}$$

Wird der Spektralradius $\rho_\infty = 1{,}0$ gewählt, ist das Verfahren zwar dissipationsfrei, aber im nichtlinearen Fall nicht stabil. Mit $\rho_\infty \in [0{,}85; 0{,}95]$ wird eine numerische Dissipation in den hochfrequenten Moden erreicht, die in den meisten Fällen zur Stabilisierung des Zeitintegrationsverfahrens ausreicht (Gee 2004).

2.1.5 Iteratives Lösungsverfahren

Nach der Diskretisierung in der Zeit ist es erforderlich, in jedem Zeitschritt $[t_n; t_{n+1}]$ die Gleichung (2.19) unter Berücksichtigung der Zeitintegrationsansätze (2.18) zu lösen. Dazu wird das Residuum $\mathcal{R}$ dieser Gleichung definiert als

$$\mathcal{R}(\mathbf{d}_{n+1}) = \mathbf{M}_\mathrm{S}\ddot{\mathbf{d}}_{n+1-\alpha_\mathrm{m}}(\mathbf{d}_{n+1}) + \mathbf{f}^\mathrm{int}(\mathbf{d}_{n+1-\alpha_\mathrm{f}}(\mathbf{d}_{n+1})) - \mathbf{f}^\mathrm{ext}_{n+1-\alpha_\mathrm{f}} = \mathbf{0}. \tag{2.24}$$

Dieses nichtlineare Gleichungssystem ist implizit abhängig von den unbekannten Verschiebungen $\mathbf{d}_{n+1}$ am Ende des Zeitschritts und wird mit dem Newton[16]-Raphson[17]-Verfahren iterativ gelöst.

Dazu wird das Residuum $\mathcal{R}$ in jedem Iterationsschritt k in eine Taylorreihe[18] um den zuletzt bekannten Verschiebungszustand $\mathbf{d}^{(k)}_{n+1}$ entwickelt, die nach dem linearen Term abgebrochen wird:

$$\mathcal{R}(\mathbf{d}^{(k+1)}_{n+1}) \approx \mathcal{R}(\mathbf{d}^{(k)}_{n+1}) + \mathbf{K}_t(\mathbf{d}^{(k)}_{n+1})\, \Delta\mathbf{d}^{(k+1)}_{n+1} = \mathbf{0}. \tag{2.25}$$

Darin bezeichnet $\mathbf{K}_t(\mathbf{d}^{(k)}_{n+1})$ die tangentiale Steifigkeitsmatrix und $\Delta\mathbf{d}^{(k+1)}_{n+1}$ das Verschiebungsinkrement.

$$\mathbf{K}_t(\mathbf{d}^{(k)}_{n+1}) = \frac{\partial\mathcal{R}(\mathbf{d}^{(k)}_{n+1})}{\partial\mathbf{d}^{(k)}_{n+1}} \qquad\qquad \Delta\mathbf{d}^{(k+1)}_{n+1} = \mathbf{d}^{(k+1)}_{n+1} - \mathbf{d}^{(k)}_{n+1} \tag{2.26}$$

Die Lösung des linearen Gleichungssystems (2.25) liefert das Verschiebungsinkrement $\Delta\mathbf{d}^{(k+1)}_{n+1}$, mit dem wiederum eine neue Approximation $\mathbf{d}^{(k+1)}_{n+1}$ der gesuchten Verschiebungen $\mathbf{d}_{n+1}$ berechnet werden kann:

$$\mathbf{d}^{(k+1)}_{n+1} = \mathbf{d}^{(k)}_{n+1} + \Delta\mathbf{d}^{(k+1)}_{n+1}. \tag{2.27}$$

Ist eine Norm des Residuums $\mathcal{R}(\mathbf{d}^{(k+1)}_{n+1})$ kleiner als eine vorgegebene Schranke ε, wird die Iteration beendet und der aktuelle Verschiebungsvektor als Lösung von Gleichung (2.24) und damit als Verschiebungszustand zum Zeitpunkt t_{n+1} betrachtet. Mit den nun bekannten Verschiebungen werden die restlichen Feldgrößen nach Gleichung (2.18) aktualisiert. Der nächste Zeitschritt kann nach dem gleichen Schema iterativ gelöst werden.

[16] Isaac Newton, * 4. Januar 1643 in Woolsthorpe-by-Colsterworth in Lincolnshire, † 31. März 1727 in Kensington, englischer Physiker, Mathematiker, Astronom, Alchemist, Philosoph und Verwaltungsbeamter.

[17] Joseph Raphson, * 1648 in Middlesex, † 1715, englischer Mathematiker.

[18] Brook Taylor, * 18. August 1685 in Edmonton, † 29. Dezember 1731 in London, britischer Mathematiker.

2.1.6 Reduktion auf den Kopplungsrand

Für den Fall der Fluid-Struktur-Wechselwirkung wird der benetzte Rand der Struktur auch als Kopplungsrand Γ mit dem Fluid bezeichnet. Dieser Teil der Berandung wird, je nach Kopplungsalgorithmus, als eine Untermenge des Neumann-Rands ($\Gamma \subset \Gamma_\mathrm{t}$) oder des Dirichlet-Rands betrachtet ($\Gamma \subset \Gamma_\mathrm{d}$).

Im Folgenden wird ein Operator eingeführt, der die in den vorherigen Abschnitten beschriebene iterative Lösung der diskreten, nichtlinearen elastodynamischen Gleichungen zusammenfasst. Dazu werden die Vektoren der Verschiebungen sowie der inneren und äußeren Kräfte in die Knoten auf dem Kopplungsrand $(\bullet)_\Gamma$ und alle übrigen Knoten $(\bullet)_\mathrm{I}$ aufgeteilt:

$$\mathbf{d} = \begin{pmatrix} \mathbf{d}_\mathrm{I} \\ \mathbf{d}_\Gamma \end{pmatrix}; \qquad \mathbf{f} = \begin{pmatrix} \mathbf{f}_\mathrm{I} \\ \mathbf{f}_\Gamma \end{pmatrix}. \tag{2.28}$$

Für die in dieser Arbeit vorgestellten Kopplungsalgorithmen ist der Kopplungsrand ein Teil des Neumann-Rands Γ_t, d.h. die unbekannten Verschiebungen am Kopplungsrand werden aus einer Kräfte-Randbedingung auf diesen Rand bestimmt. Dazu wird der nichtlineare Operator $\mathcal{S}$ eingeführt

$$\mathbf{d}_{\Gamma,n+1} = \mathcal{S}_{n+1}(\mathbf{f}_{\Gamma,n+1}), \tag{2.29}$$

der die Verschiebungen $\mathbf{d}_{\Gamma,n+1}$ des Kopplungsrands für einen bestimmten Zeitschritt aus den gegebenen Kopplungskräften $\mathbf{f}_{\Gamma,n+1}$, unter Berücksichtigung aller übrigen äußeren Kräfte $\mathbf{f}^\mathrm{ext}_{n+1}$ (Volumenkräfte und sonstige Randkräfte) berechnet.

2.2 Fluiddynamik

Für die Beschreibung der kontinuumsmechanischen Grundlagen des Fluids und der verwendeten Diskretisierungsverfahren wird nun eine eher mathematisch orientierte Notation unter Verwendung des inneren Produkts verwendet. Dies bietet sich aufgrund der höheren Komplexität, besonders im Fall der Stabilisierung, an, um eine übersichtliche Darstellung zu gewährleisten. Die verwendete Formulierung und Schreibweise basiert auf den Arbeiten von Wall (1999) und Förster (2007).

Die in dieser Arbeit betrachteten Fluide werden als inkompressibel angesehen. Weiterhin werden Strömungen mit niedriger oder moderater Reynolds[19]-Zahl betrachtet, bei denen

[19] Osborne Reynolds, * 23. August 1842 in Belfast, † 21. Februar 1912 in Watchet in Somerset, englischer Physiker.

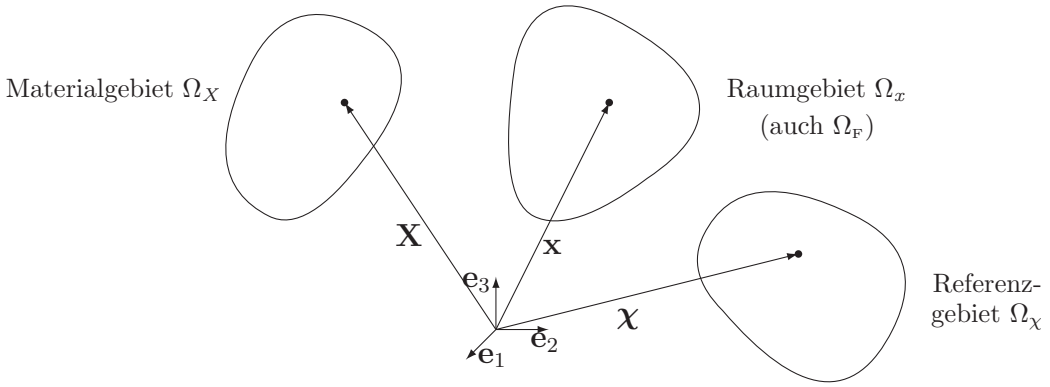


Abbildung 2.3: Kontinuumsmechanische Gebiete für die ALE-Formulierung der Fluiddynamik.

die Viskosität nicht vernachlässigt werden kann. Besonders im Bereich von Strukturen ist es möglich, dass viskose Randschichten auftreten, die das Verhalten des gesamten gekoppelten Problems beeinflussen.

Für die Beschreibung der Strömung eines reinen Fluids wird üblicherweise die Euler[20]- oder auch räumliche Betrachtungsweise gewählt. Dabei sitzt der Beobachter fest an einem Raumpunkt und misst die Eigenschaften der materiellen Punkte, die beim Durchströmen des Kontrollvolumens im Laufe der Zeit diesen Punkt einnehmen. Für die Diskretisierung mit Finiten Elementen bedeutet dies, dass das Fluid durch ein raumfestes Netz hindurchströmt.

Bei der Simulation von Fluid-Struktur-Wechselwirkungen verändert sich jedoch das Fluidgebiet im Laufe der Simulation durch die Deformation der Struktur. Das Fluidnetz muss an das neue Fluidgebiet angepasst werden. Daher wird in dieser Arbeit für das Fluid eine „Arbitrary Lagrangean Eulerian"-Formulierung (ALE) (Donea 1983; Hughes u. a. 1981) verwendet. Dabei werden die Strömungsgleichungen in einem beliebig bewegten Referenzgebiet Ω_χ mit den Ortsvektoren χ formuliert. Dieses entspricht dem zeitabhängigen Fluidnetz (Abbildung 2.3).

[20]Leonhard Euler, * 15. April 1707 in Basel, † 18. September 1783 in Sankt Petersburg, schweizer Mathematiker.

2.2.1 Kontinuumsmechanische Grundgleichungen

Kinematik

Die primäre kinematische Variable im Fluid ist das Geschwindigkeitsfeld $\mathbf{u}_{\mathrm{x}}$, dessen symmetrischer Gradient die Deformationsgeschwindigkeit $\boldsymbol{\varepsilon}$ definiert

$$\boldsymbol{\varepsilon}(\mathbf{u}_{\mathrm{x}}) = \frac{1}{2}\left(\nabla_x \mathbf{u}_{\mathrm{x}} + (\nabla_x \mathbf{u}_{\mathrm{x}})^{\mathrm{T}}\right), \tag{2.30}$$

wobei ∇_x den räumlichen Gradienten-Operator bezeichnet.

Konstitutivgesetz

Da Fluide in der Ruhelage keinen Scherkräften widerstehen können und daher in diesem Zustand ein reiner hydrostatischer Spannungszustand herrscht, werden die Cauchy-Spannungen $\boldsymbol{\sigma}_{\mathrm{F}}$ unter Verwendung des Einheitstensors $\mathbf{I}$ in den hydrostatischen Druck $\bar{p}_{\mathrm{x}}$ und den Tensor der viskosen Schubspannungen $\boldsymbol{\tau}$ aufgeteilt:

$$\boldsymbol{\sigma}_{\mathrm{F}} = -\mathbf{I}\bar{p}_{\mathrm{x}} + \boldsymbol{\tau}. \tag{2.31}$$

Für Newton'sche Fluide wird eine lineare Beziehung zwischen der Deformationsgeschwindigkeit $\boldsymbol{\varepsilon}$ und den Schubspannungen $\boldsymbol{\tau}$ angenommen. Ist das Fluid inkompressibel, führt dies auf die folgende konstitutive Gleichung:

$$\boldsymbol{\sigma}_{\mathrm{F}} = -\mathbf{I}\bar{p}_{\mathrm{x}} + 2\mu_{\mathrm{F}}\boldsymbol{\varepsilon}. \tag{2.32}$$

Der einzige Materialparameter für das Fluid ist somit die dynamische Viskosität μ_{F}. Im Folgenden werden die auf die Dichte des Fluids ρ_{F} bezogenen Größen der kinematischen Viskosität ν_{F} und des kinematischen Drucks p_{x} verwendet.

$$\nu_{\mathrm{F}} = \frac{\mu_{\mathrm{F}}}{\rho_{\mathrm{F}}}; \qquad p_{\mathrm{x}} = \frac{\bar{p}_{\mathrm{x}}}{\rho_{\mathrm{F}}} \tag{2.33}$$

Massen- und Impulsbilanz

Neben der kinematischen Feldgleichung und der konstitutiven Gleichung werden zur Formulierung des Anfangs-Randwert-Problems für inkompressible, viskose Strömungen die lokale Massen- und die lokale Impulsbilanz benötigt.

Die lokale Massenbilanz in räumlicher Darstellung ist

$$\left.\frac{\partial \rho_{\mathrm{F}}}{\partial t}\right|_x + \nabla_x \cdot (\rho_{\mathrm{F}} \mathbf{u}_{\mathrm{x}}) = 0. \tag{2.34}$$

Für den hier betrachteten Fall einer konstanten Dichte im gesamten Fluidgebiet und einer maximalen Geschwindigkeit, die deutlich kleiner ist als die Schallgeschwindigkeit im Fluid, d.h. einer kleinen Mach[21]-Zahl, vereinfacht sich diese zur Kontinuitätsgleichung oder auch Inkompressibilitätsbedingung:

$$\nabla_x \cdot \mathbf{u}_{\mathrm{x}} = 0. \tag{2.35}$$

Die lokale Impulsbilanz in räumlicher Form lautet

$$\rho_{\mathrm{F}} \left.\frac{\partial \mathbf{u}_{\mathrm{x}}}{\partial t}\right|_x + \rho_{\mathrm{F}} \mathbf{u}_{\mathrm{x}} \cdot \nabla_x \mathbf{u}_{\mathrm{x}} - \nabla_x \cdot \boldsymbol{\sigma}_{\mathrm{F}} = \rho_{\mathrm{F}} \hat{\mathbf{b}}_{\mathrm{F}}, \tag{2.36}$$

wobei $\hat{\mathbf{b}}_{\mathrm{F}}$ Volumenlasten im Fluid bezeichnet. Diese räumliche Darstellung unterscheidet sich von der materiellen Formulierung der lokalen Impulsbilanz (vgl. (2.6)) vor allem durch den konvektiven Term $\mathbf{u}_{\mathrm{x}} \cdot \nabla_x \mathbf{u}_{\mathrm{x}}$, der aus der räumlichen Form der materiellen Zeitableitung der Geschwindigkeit resultiert:

$$\dot{\mathbf{u}}_{\mathrm{x}} = \frac{\mathrm{D}\mathbf{u}_{\mathrm{x}}}{\mathrm{D}t} = \left.\frac{\partial \mathbf{u}_{\mathrm{x}}}{\partial t}\right|_X = \left.\frac{\partial \mathbf{u}_{\mathrm{x}}}{\partial t}\right|_x + \left.\frac{\partial \mathbf{u}_{\mathrm{x}}}{\partial \mathbf{x}} \cdot \frac{\partial \mathbf{x}}{\partial t}\right|_X = \left.\frac{\partial \mathbf{u}_{\mathrm{x}}}{\partial t}\right|_x + \mathbf{u}_{\mathrm{x}} \cdot \nabla_x \mathbf{u}_{\mathrm{x}}. \tag{2.37}$$

Um von dieser reinen Euler-Formulierung auf die ALE-Formulierung überzugehen, werden alle räumlichen Ableitungen weiterhin im räumlichen Gebiet Ω_x ausgeführt. Nur die materielle Zeitableitung der Geschwindigkeit wird in das Referenzsystem Ω_χ transformiert.

$$\dot{\mathbf{u}}_{\mathrm{x}} = \frac{\mathrm{D}\mathbf{u}_{\mathrm{x}}}{\mathrm{D}t} = \left.\frac{\partial \mathbf{u}_{\mathrm{x}}}{\partial t}\right|_X = \left.\frac{\partial \mathbf{u}_{\mathrm{x}}}{\partial t}\right|_\chi + \mathbf{c}_{\mathrm{x}} \cdot \nabla_x \mathbf{u}_{\mathrm{x}} \tag{2.38}$$

Dabei bezeichnet $\mathbf{c}_{\mathrm{x}}$ die konvektive Geschwindigkeit, die als Differenz zwischen der materiellen Partikelgeschwindigkeit $\mathbf{u}_{\mathrm{x}}$ und der Geschwindigkeit des Referenzgebiets bzw. des Netzes $\mathbf{u}_{\mathrm{x}}^{\mathrm{G}}$ definiert ist:

$$\mathbf{c}_{\mathrm{x}} = \mathbf{u}_{\mathrm{x}} - \mathbf{u}_{\mathrm{x}}^{\mathrm{G}}. \tag{2.39}$$

[21] Ernst Mach, * 18. Februar 1838 in Chirlitz-Turas, † 19. Februar 1916 in Haar (bei München), österreichischer Physiker, Philosoph und Wissenschaftstheoretiker.

Somit können für die ALE-Formulierung die konstitutive Gleichung und die lokale Massenbilanz unverändert in der räumlichen Darstellung verwendet werden, nur die Impulsbilanz wird zur lokalen ALE-Impulsbilanz umformuliert.

$$\rho_{\mathrm{F}} \left.\frac{\partial \mathbf{u}_{\mathrm{x}}}{\partial t}\right|_{\chi} + \rho_{\mathrm{F}} \mathbf{c}_{\mathrm{x}} \cdot \nabla_x \mathbf{u}_{\mathrm{x}} - \nabla_x \cdot \boldsymbol{\sigma}_{\mathrm{F}} = \rho_{\mathrm{F}} \hat{\mathbf{b}}_{\mathrm{F}} \tag{2.40}$$

Geometrische Bilanzgleichungen

Die geometrischen Bilanzgleichungen (GCL, „geometric conservation laws“) stellen Bedingungen an die Bestimmung der geometrischen Größen des zeitveränderlichen Fluidgebiets, die Netzposition und -geschwindigkeit $\mathbf{u}_{\mathrm{x}}^{\mathrm{G}}$. Dadurch wird sichergestellt, dass ein konstanter Strömungszustand auch auf einem bewegten Netz exakt wiedergegeben werden kann. Eine Verletzung der GCL kann bei der Simulation von inkompressiblen Strömungen zu einer künstlichen Massenproduktion und zu parasitären Oszillationen führen (Mok 2001).

Wird, wie in dieser Arbeit, eine konvektive ALE-Formulierung der Navier[22]-Stokes[23]-Gleichungen verwendet, sind die geometrischen Bilanzgleichungen nach Förster u. a. (2006b) unabhängig vom verwendeten Zeitintegrationsverfahren erfüllt. Außerdem hat die Berechnung der Netzgeschwindigkeit $\mathbf{u}_{\mathrm{x}}^{\mathrm{G}}$ in diesem Fall keinen Einfluss auf die Erfüllung der geometrischen Bilanzgleichungen.

2.2.2 Anfangs-Randwert-Problem

Starke Form

Das Anfangs-Randwert-Problem für inkompressible, viskose Strömungen wird hier nur in der Formulierung auf zeitveränderlichen Gebieten angegeben, da es in dieser Form für die gekoppelten Fluid-Struktur-Interaktions-Probleme verwendet wird. Es setzt sich zusammen aus den Feldgleichungen (2.30), (2.32), (2.35) und (2.40) sowie den Anfangs- und Randbedingungen.

Durch Einsetzen der konstitutiven Gleichung (2.32) in die lokale Impulsbilanz (2.40) und Division durch die Dichte ρ_{F} ergibt sich die konvektive Formulierung der Impulsbilanz,

[22] Claude Louis Marie Henri Navier, * 10. Februar 1785 in Dijon, † 21. August 1836 in Paris, französischer Mathematiker und Physiker.

[23] George Gabriel Stokes, * 13. August 1819 in Skreen, † 1. Februar 1903 in Cambridge, irischer Mathematiker und Physiker.

die zusammen mit der Kontinuitätsgleichung die instationären, inkompressiblen Navier-Stokes-Gleichungen bildet:

$$\left.\frac{\partial \mathbf{u}_{\mathrm{x}}}{\partial t}\right|_{\chi} + \mathbf{c}_{\mathrm{x}} \cdot \nabla_x \mathbf{u}_{\mathrm{x}} - 2\nu_{\mathrm{F}} \nabla_x \cdot \boldsymbol{\varepsilon}(\mathbf{u}_{\mathrm{x}}) + \nabla_x p_{\mathrm{x}} = \hat{\mathbf{b}}_{\mathrm{F}} \qquad \text{in } \Omega_{\mathrm{F}} \times T \tag{2.41}$$

$$\nabla_x \cdot \mathbf{u}_{\mathrm{x}} = 0 \qquad \text{in } \Omega_{\mathrm{F}} \times T. \tag{2.42}$$

Als Anfangsbedingung wird die Geschwindigkeit $\mathbf{u}_{\mathrm{x},0}$ auf dem gesamten Fluidgebiet Ω_{F} zur Zeit t_0 vorgeschrieben, wobei auch hier die Inkompressibilitätsbedingung erfüllt sein muss. Für den Druck p_{x} gibt es keine Anfangsbedingungen in einer inkompressiblen Strömung. Er fungiert vielmehr als Lagrange-Multiplikator für die Inkompressibilitätsbedingung und passt sich an das Geschwindigkeitsfeld an.

Der Rand $\Gamma_{\mathrm{F}} = \partial\Omega_{\mathrm{F}}$ des Fluidgebiets teilt sich, wie auch bei der Struktur, in zwei disjunkte Untermengen auf.

$$\Gamma_{\mathrm{F}} = \Gamma_{\mathrm{u}} \cup \Gamma_{\mathrm{h}} \quad \text{mit} \qquad \Gamma_{\mathrm{u}} \cap \Gamma_{\mathrm{h}} = 0. \tag{2.43}$$

Als Dirichlet-Randbedingungen werden im Fluid auf dem Rand Γ_{u} die Geschwindigkeiten $\hat{\mathbf{u}}_{\mathrm{x}}$ vorgeschrieben, wohingegen auf dem anderen Teil des Rands Γ_{h} durch Vorgabe der Randlasten $\hat{\mathbf{h}}$ eine Neumann-Randbedingung vorliegt.

Für die Simulation von inkompressiblen Strömungen werden üblicherweise folgende Randbedingungen verwendet, deren physikalische Bedeutung hier kurz beschrieben wird. Ausführlichere Behandlungen von Randbedingungen finden sich in Wall (1999) und Förster (2007) oder auch Gresho und Sani (1998).

- **Haft- oder „no-slip“-Randbedingungen** sind für die Navier-Stokes-Gleichungen möglich, da viskose Effekte berücksichtigt werden. Es wird angenommen, dass ein Fluidpartikel an einem festen oder beweglichen Rand haftet, und es werden daher Dirichlet-Randbedingungen für die tangentiale und normale Geschwindigkeit vorgegeben.

- **Gleit- oder „slip“-Randbedingungen** schreiben nur die Normalgeschwindigkeit an einem Rand vor und werden z.B. auch an künstlichen Rändern des Simulationsgebiets verwendet.

- **Ausfluss-Randbedingungen** stellen eine besondere Herausforderung dar, da sie nicht von der Natur vorgegeben werden, sondern aus der Notwendigkeit entstehen, das Simulationsgebiet in der Größe zu beschränken. In dieser Arbeit wird die weit verbreitete „do nothing“-Randbedingung verwendet, die eine homogene

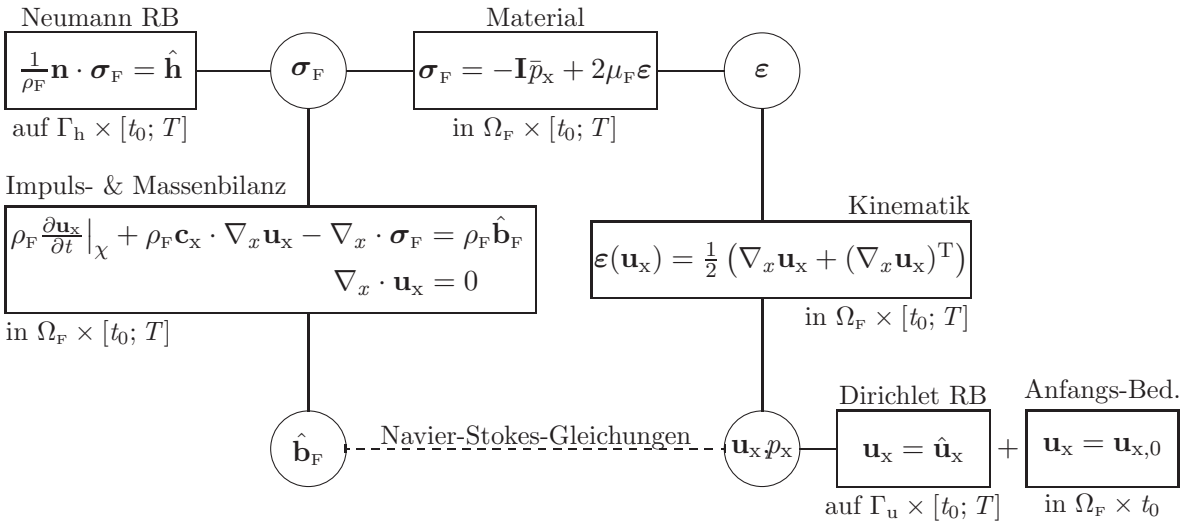


Abbildung 2.4: Tonti-Diagramm der starken Form des Anfangs-Randwert-Problems für instationäre, inkompressible Strömungen.

Neumann-Randbedingung impliziert. Eine ausführlichere Diskussion zu Ausfluss-Randbedingungen befindet sich in Heywood u. a. (1996) oder Gresho (1992).

- **Druck-Randbedingungen** werden benötigt, da der Druck durch die inkompressiblen Navier-Stokes-Gleichungen bis auf eine Konstante bestimmt ist. Für reine Fluid-Probleme wird der Druck häufig implizit durch Neumann-Randbedingungen bestimmt oder an einem Punkt vorgegeben. Bei der Fluid-Struktur-Interaktion hingegen wird das Druckniveau in manchen Fällen durch die Kopplungsbedingungen aus den Spannungen in der Struktur bestimmt.

Die starke Form des Anfangs-Randwert-Problems für inkompressible, viskose Strömungen kann, wie auch schon für die Struktur, in einem Tonti-Diagramm (Abbildung 2.4) zusammengefasst werden.

Schwache Form

Da im Allgemeinen keine analytische Lösung für dieses System gekoppelter, partieller Differenzialgleichungen gefunden werden kann, die diese Gleichungen punktweise erfüllt, wird die abgeschwächte Bedingung eingeführt, dass das Residuum der Gleichungen im Integral über das Gebiet zu null werden muss. Dazu werden die Residuen jeweils mit einer Testfunktion multipliziert und über das Raumgebiet des Fluids Ω_F integriert. Die

Lösungs- und Testfunktionen werden dabei aus den folgenden Funktionsräumen gewählt:

$$\begin{aligned}\mathbb{V} &= \left\{\mathbf{u}_{\mathrm{x}} \in \mathbb{H}^1(\Omega_{\mathrm{F}}) : \mathbf{u}_{\mathrm{x}} = \hat{\mathbf{u}}_{\mathrm{x}} \text{ auf } \Gamma_{\mathrm{u}}\right\} \\ \mathbb{V}_0 &= \left\{\mathbf{v}_{\mathrm{x}} \in \mathbb{H}^1(\Omega_{\mathrm{F}}) : \mathbf{v}_{\mathrm{x}} = 0 \text{ auf } \Gamma_{\mathrm{u}}\right\} \\ \mathbb{P} &= \left\{p_{\mathrm{x}} \in \mathbb{L}^2(\Omega_{\mathrm{F}})\right\}.\end{aligned} \tag{2.44}$$

$\mathbb{H}^1(\Omega_{\mathrm{F}})$ bezeichnet den Sobolew[24]-Raum von Funktionen auf Ω_{F} mit quadratintegrierbaren Werten und ersten Ableitungen. $\mathbb{L}^2(\Omega_{\mathrm{F}})$ bezeichnet den Raum von quadratintegrierbaren Funktionen auf Ω_{F}.

Werden als Testfunktionen für die Impulsbilanz die Funktionen $\mathbf{v}_{\mathrm{x}} \in \mathbb{V}_0$ und für die Massenbilanz $q_{\mathrm{x}} \in \mathbb{P}$ gewählt, ergibt sich:

$$\begin{aligned}\left(\dot{\mathbf{u}}_{\mathrm{x}},\mathbf{v}_{\mathrm{x}}\right)_{\Omega_{\mathrm{F}}} + \left(\mathbf{c}_{\mathrm{x}} \cdot \nabla_x \mathbf{u}_{\mathrm{x}},\mathbf{v}_{\mathrm{x}}\right)_{\Omega_{\mathrm{F}}} - \left(2\nu_{\mathrm{F}} \nabla_x \cdot \boldsymbol{\varepsilon}(\mathbf{u}_{\mathrm{x}}),\mathbf{v}_{\mathrm{x}}\right)_{\Omega_{\mathrm{F}}} \\ + \left(\nabla_x p_{\mathrm{x}},\mathbf{v}_{\mathrm{x}}\right)_{\Omega_{\mathrm{F}}} + \left(\nabla_x \cdot \mathbf{u}_{\mathrm{x}},q_{\mathrm{x}}\right)_{\Omega_{\mathrm{F}}} = \left(\hat{\mathbf{b}}_{\mathrm{F}},\mathbf{v}_{\mathrm{x}}\right)_{\Omega_{\mathrm{F}}}.\end{aligned} \tag{2.45}$$

Darin bedeuten $\dot{\mathbf{u}}_{\mathrm{x}}$ die Beschleunigungen und $(\bullet,\bullet)_{\Omega_{\mathrm{F}}}$ das L^2-innere Produkt, ausgewertet über dem Gebiet Ω_{F}. Für die verwendete ALE-Formulierung wird die Integration über das zeitveränderliche Raumgebiet ausgeführt, d.h. Ω_{F} entspricht Ω_x. Durch die Anwendung des Gauß'schen Integralsatzes auf den Viskositäts- und Druckgradiententerm, werden dort Ableitungen auf die Testfunktion verschoben und dadurch die Differenzierbarkeitsanforderungen an die Funktionen abgeschwächt. Die resultierende schwache Form lautet:

Finde $\mathbf{u}_{\mathrm{x}} \in \mathbb{V}$ und $p_{\mathrm{x}} \in \mathbb{P}$, so dass

$$B_{\mathrm{gal}}\left(\{\mathbf{u}_{\mathrm{x}},p_{\mathrm{x}}\},\{\mathbf{v}_{\mathrm{x}},q_{\mathrm{x}}\}\right) = F\left(\mathbf{v}_{\mathrm{x}}\right) \qquad \forall\left(\mathbf{v}_{\mathrm{x}},q_{\mathrm{x}}\right) \in \left(\mathbb{V}_0,\mathbb{P}\right) \tag{2.46}$$

mit

$$\begin{aligned}B_{\mathrm{gal}}\left(\{\mathbf{u}_{\mathrm{x}},p_{\mathrm{x}}\},\{\mathbf{v}_{\mathrm{x}},q_{\mathrm{x}}\}\right) = \left(\dot{\mathbf{u}}_{\mathrm{x}},\mathbf{v}_{\mathrm{x}}\right)_{\Omega_{\mathrm{F}}} + \left(\mathbf{c}_{\mathrm{x}} \cdot \nabla_x \mathbf{u}_{\mathrm{x}},\mathbf{v}_{\mathrm{x}}\right)_{\Omega_{\mathrm{F}}} + \left(2\nu_{\mathrm{F}}\boldsymbol{\varepsilon}(\mathbf{u}_{\mathrm{x}}),\boldsymbol{\varepsilon}(\mathbf{v}_{\mathrm{x}})\right)_{\Omega_{\mathrm{F}}} \\ - \left(p_{\mathrm{x}},\nabla_x \cdot \mathbf{v}_{\mathrm{x}}\right)_{\Omega_{\mathrm{F}}} - \left(\nabla_x \cdot \mathbf{u}_{\mathrm{x}},q_{\mathrm{x}}\right)_{\Omega_{\mathrm{F}}}\end{aligned} \tag{2.47}$$

$$F\left(\mathbf{v}_{\mathrm{x}}\right) = \left(\hat{\mathbf{b}}_{\mathrm{F}},\mathbf{v}_{\mathrm{x}}\right)_{\Omega_{\mathrm{F}}} + \left(\hat{\mathbf{h}},\mathbf{v}_{\mathrm{x}}\right)_{\Gamma_{\mathrm{h}}}. \tag{2.48}$$

[24]Sergei Lwowitsch Sobolew, * 6. Oktober 1908 in Sankt Petersburg, † 3. Januar 1989 in Moskau, russischer Mathematiker.

2.2.3 Diskretisierung im Raum

Wie bereits bei der Struktur wird das Fluidgebiet Ω_F durch das diskretisierte Gebiet $\Omega_\mathrm{F}^\mathrm{h}$ approximiert, welches aus n_ele Finiten Elementen Ω_e besteht:

$$\Omega_\mathrm{F} \approx \Omega_\mathrm{F}^\mathrm{h} = \bigcup_{e=1}^{n_\mathrm{ele}} \Omega_\mathrm{e}. \tag{2.49}$$

Dabei werden die Räume der Lösungs- und Testfunktionen $\mathbb{V}$, $\mathbb{V}_0$ und $\mathbb{P}$ durch die endlich-dimensionalen (finiten) Unterräume $\mathbb{V}^\mathrm{h}$, $\mathbb{V}_0^\mathrm{h}$ und $\mathbb{P}^\mathrm{h}$ ersetzt. Der Geschwindigkeits- und Druckverlauf in jedem Element $\mathbf{u}_\mathrm{x,e}$ und $p_\mathrm{x,e}$ wird approximativ durch polynomische Ansatzfunktionen $\mathbf{N}$ zwischen den Knotenwerten des Elements für die Geschwindigkeiten und den Druck ($\mathbf{u}_\mathrm{e}$ und p_e) interpoliert.

$$\begin{aligned} \mathbf{u}_\mathrm{x,e} &\approx \mathbf{u}_\mathrm{x,e}^\mathrm{h} = \mathbf{N}\mathbf{u}_\mathrm{e} & \qquad \mathbf{v}_\mathrm{x,e} &\approx \mathbf{v}_\mathrm{x,e}^\mathrm{h} = \mathbf{N}\mathbf{v}_\mathrm{e} \\ p_\mathrm{x,e} &\approx p_\mathrm{x,e}^\mathrm{h} = \mathbf{N}p_\mathrm{e} & \qquad q_\mathrm{x,e} &\approx q_\mathrm{x,e}^\mathrm{h} = \mathbf{N}q_\mathrm{e} \end{aligned} \tag{2.50}$$

Die Verwendung einer reinen Bubnow-Galerkin-Diskretisierung für die schwache Form des Anfangs-Randwert-Problems für inkompressible, viskose Strömungen führt zu zwei Arten von numerischen Problemen. Eine ausführlichere Beschreibung und mathematische Analyse dieser Schwächen findet sich z.B. in Wall (1999), Gravemeier (2003) und Förster (2007) oder in Donea und Huerta (1989).

Das erste dieser Probleme hängt mit dem Konvektionsterm zusammen und verursacht künstliche Geschwindigkeitsoszillationen in konvektionsdominanten Strömungen. Diese können nur durch eine beträchtliche Netzverfeinerung vermieden werden, so dass die Konvektion bezogen auf die Elementgröße nicht mehr dominant ist und Grenzschichten aufgelöst werden.

Die zweite numerische Schwierigkeit resultiert aus den in dieser Arbeit gewählten gleichen Interpolationsordnungen für Geschwindigkeit und Druck. In diesem Fall ist die Ladyschenskaja[25]-Babuška[26]-Brezzi[27]-Bedingung (LBB-Bedingung), die für die vorliegende gemischte Methode eine notwendige Bedingung für die Existenz und Eindeutigkeit der Lösung ist, nicht erfüllt. Es treten starke, unphysikalische Oszillationen des Druckfelds in Form eines Schachbrettmusters auf, die auch als Schachbrettmoden oder „checkerboard modes“ bezeichnet werden. In Brezzi und Fortin (1991) und Girault und Raviert (1986) ist eine umfassende Diskussion der Theorie der gemischten Methoden zu finden.

[25] Olga Alexandrowna Ladyschenskaja, * 7. März 1922 in Kologriw, † 12. Januar 2004 in Sankt Petersburg, russische Mathematikerin und Physikerin.
[26] Ivo M. Babuška, * 22. März 1926 in Prag, tschechisch-amerikanischer Mathematiker.
[27] Franco Brezzi, * 29. April 1945 in Vimercate, italienischer Mathematiker.

Stabilisierung

Zur Vermeidung dieser Instabilitäten stehen in der Literatur verschiedene Ansätze zur Verfügung. Die in dieser Arbeit verwendeten residuenbasierten Ansätze gehen zurück auf Arbeiten von Hughes und Mitarbeitern zur Advektions-Diffusions-Gleichung (Hughes und Brooks 1976, 1982) oder zur Umgehung der LBB-Bedingung beim Stokes-Problem (Hughes u. a. 1986). Auf die Navier-Stokes-Gleichungen wurde eine Kombination dieser beiden Ideen von Tezduyar u. a. (1992) angewandt: das SUPG/PSPG-Verfahren (Streamline Upwind Petrow[28]-Galerkin/Pressure Stabilized Petrow-Galerkin). Auch das Galerkin/Least-Squares-Verfahren (GLS), entwickelt von Hughes u. a. (1989), und die Unusual Stabilized Finite Element Method (USFEM) von Franca und Farhat (1995) sind in der hier verwendeten Formulierung der Stabilisierung enthalten.

Alternativ können Stabilisierungsterme auch durch Anwendung des sogenannten „finite increment calculus“, vorgeschlagen von Oñate (1998), oder eine polynomische Projektion des Drucks, wie von Dohrmann und Bochev (2004) entwickelt, hergeleitet werden. Einen umfassenden Überblick über Stabilisierungsverfahren, deren Herleitungen und weiterführende Literatur finden sich in Förster (2007) oder Wall (1999).

Die semidiskrete schwache Form des stabilisierten ALE-Finite-Element-Verfahrens lautet:

Finde $\mathbf{u}_{\mathrm{x}}^{\mathrm{h}} \in \mathbb{V}^{\mathrm{h}}$ und $p_{\mathrm{x}}^{\mathrm{h}} \in \mathbb{P}^{\mathrm{h}}$, so dass

$$\begin{aligned} B_{\mathrm{gal}}\left(\left\{\mathbf{u}_{\mathrm{x}}^{\mathrm{h}},p_{\mathrm{x}}^{\mathrm{h}}\right\},\left\{\mathbf{v}_{\mathrm{x}}^{\mathrm{h}},q_{\mathrm{x}}^{\mathrm{h}}\right\}\right) &- \sum_e \left(\tau_{\mathrm{M}e}\mathcal{R}_{\mathrm{M}}\left(\mathbf{u}_{\mathrm{x}}^{\mathrm{h}},p_{\mathrm{x}}^{\mathrm{h}}\right),\mathcal{L}_{\mathrm{M}}^{\mathrm{stab}}\left(\mathbf{u}_{\mathrm{x}}^{\mathrm{h}},\left\{\mathbf{v}_{\mathrm{x}}^{\mathrm{h}},q_{\mathrm{x}}^{\mathrm{h}}\right\}\right)\right)_{\Omega_e} \\ &+ \sum_e \left(\tau_{\mathrm{C}e}\mathcal{R}_{\mathrm{C}}\left(\mathbf{u}_{\mathrm{x}}^{\mathrm{h}}\right),\mathcal{L}_{\mathrm{C}}^{\mathrm{stab}}\left(\mathbf{v}_{\mathrm{x}}^{\mathrm{h}}\right)\right)_{\Omega_e} \\ &= F\left(\mathbf{v}_{\mathrm{x}}^{\mathrm{h}}\right) \qquad\qquad \forall\left(\mathbf{v}_{\mathrm{x}}^{\mathrm{h}},q_{\mathrm{x}}^{\mathrm{h}}\right) \in \left(\mathbb{V}_0^{\mathrm{h}},\mathbb{P}^{\mathrm{h}}\right). \end{aligned} \tag{2.51}$$

Hierin bezeichnet $\mathcal{R}_{\mathrm{M}}\left(\mathbf{u}_{\mathrm{x}}^{\mathrm{h}},p_{\mathrm{x}}^{\mathrm{h}}\right)$ das Residuum der Impulsbilanz („momentum balance equation“) und $\mathcal{R}_{\mathrm{C}}\left(\mathbf{u}_{\mathrm{x}}^{\mathrm{h}}\right)$ das Residuum der Kontinuitätsgleichung („continuity equation“):

$$\mathcal{R}_{\mathrm{M}}\left(\mathbf{u}_{\mathrm{x}}^{\mathrm{h}},p_{\mathrm{x}}^{\mathrm{h}}\right) = \dot{\mathbf{u}}_{\mathrm{x}}^{\mathrm{h}} + \mathbf{c}_{\mathrm{x}}^{\mathrm{h}} \cdot \nabla_x \mathbf{u}_{\mathrm{x}}^{\mathrm{h}} - 2\nu_{\mathrm{F}}\nabla_x \cdot \boldsymbol{\varepsilon}(\mathbf{u}_{\mathrm{x}}^{\mathrm{h}}) + \nabla_x p_{\mathrm{x}}^{\mathrm{h}} - \hat{\mathbf{b}}_{\mathrm{F}} \tag{2.52}$$

$$\mathcal{R}_{\mathrm{C}}\left(\mathbf{u}_{\mathrm{x}}^{\mathrm{h}}\right) = \nabla_x \cdot \mathbf{u}_{\mathrm{x}}^{\mathrm{h}}. \tag{2.53}$$

[28] Georgi Iwanowitsch Petrow, * 31. Mai 1912, † 17. Mai 1987, sowjetischer Ingenieur.

Die Stabilisierungs-Operatoren für die Gleichungen werden mit $\mathcal{L}_{\mathrm{M}}^{\mathrm{stab}}\left(\mathbf{u}_{\mathrm{x}}^{\mathrm{h}}, \left\{\mathbf{v}_{\mathrm{x}}^{\mathrm{h}}, q_{\mathrm{x}}^{\mathrm{h}}\right\}\right)$ und $\mathcal{L}_{\mathrm{C}}^{\mathrm{stab}}\left(\mathbf{v}_{\mathrm{x}}^{\mathrm{h}}\right)$ bezeichnet und sind gegeben als

$$\mathcal{L}_{\mathrm{M}}^{\mathrm{stab}}\left(\mathbf{u}_{\mathrm{x}}^{\mathrm{h}}, \left\{\mathbf{v}_{\mathrm{x}}^{\mathrm{h}}, q_{\mathrm{x}}^{\mathrm{h}}\right\}\right) = -\mathbf{c}_{\mathrm{x}}^{\mathrm{h}} \cdot \nabla_x \mathbf{v}_{\mathrm{x}}^{\mathrm{h}} - \alpha\, 2\nu_{\mathrm{F}} \nabla_x \cdot \boldsymbol{\varepsilon}(\mathbf{v}_{\mathrm{x}}^{\mathrm{h}}) - \nabla_x q_{\mathrm{x}}^{\mathrm{h}} \tag{2.54}$$

$$\mathcal{L}_{\mathrm{C}}^{\mathrm{stab}}\left(\mathbf{v}_{\mathrm{x}}^{\mathrm{h}}\right) = \nabla_x \cdot \mathbf{v}_{\mathrm{x}}^{\mathrm{h}}. \tag{2.55}$$

Durch die Wahl des Parameters $\alpha \in \{-1; 0; 1\}$ lassen sich mit dieser Schreibweise verschiedene Varianten der Stabilisierung darstellen. Wird $\alpha = 0$ gewählt, folgt das klassische SUPG/PSPG-Verfahren (Tezduyar u. a. 1992), das auch unter dem Namen „streamline diffusion“ (Hansbo und Szepessy 1990) bekannt ist.

Bei zusätzlicher Berücksichtigung des viskosen Terms der Impulsbilanz im Stabilisierungs-Operator ($\alpha = -1$), d.h. bei Verwendung des kompletten negativen Operators der stationären Impulsbilanz, entspricht dies einem GLS-Verfahren (Hughes u. a. 1989).

Umdrehen des Vorzeichens des viskosen Terms ($\alpha = +1$) führt auf eine Variante der „unusual stabilized finite element method“ (USFEM) (Franca und Farhat 1995), deren ursprüngliche Formulierung auch den instationären Term der Impulsbilanz im Stabilisierungs-Operator berücksichtigt.

In dieser Arbeit wird hauptsächlich das SUPG/PSPG-Verfahren verwendet.

Die Wahl der Stabilisierungsparameter $\tau_{\mathrm{M}e}$ und $\tau_{\mathrm{C}e}$ ist entscheidend für die Stabilität der Methode. Es existieren in der Literatur zahlreiche Definitionen für diese Parameter, viele davon liefern stabile Ergebnisse unabhängig von der Problemstellung. Eine Sammlung von Literaturstellen, die Stabilisierungsparameter für die inkompressiblen Navier-Stokes-Gleichungen vorschlagen, findet sich in Wall (1999) und Förster (2007).

Der in dieser Arbeit verwendete Parameter für die Impulsbilanz basiert auf Stabilisierungsparametern, die von Barrenechea und Valentin (2002) für das Stokes-Problem und von Franca und Valentin (2000) für die Reaktions-Diffusions-Gleichung durch die Kondensation einer „bubble“ hergeleitet wurden. Förster (2007) hat diesen für den ALE-Fall verallgemeinert.

$$\tau_{\mathrm{M}e} = \frac{h_e^2}{h_e^2 \xi_1 + \frac{4\nu_{\mathrm{f}}}{m_e} \xi_2} \tag{2.56}$$

Für die Kontinuitätsgleichung wird eine Definition des Stabilisierungsparameters verwendet, wie sie von Codina (2002) vorgeschlagen wurde:

$$\tau_{\mathrm{C}e} = \sqrt{4\nu_{\mathrm{F}}^2 + \left(\frac{c_2}{c_1} |\mathbf{c}_{\mathrm{x}}^{\mathrm{h}}| h_e\right)^2}. \tag{2.57}$$

Die Parameter ξ_1 und ξ_2 hängen von der im Element dominierenden Strömungseigenschaft (Diffusion oder Konvektion) ab, wohingegen m_e den Einfluss der Diskretisierung berücksichtigt. Die genaue Definition der für diese beiden Stabilisierungsparameter verwendeten Größen, insbesondere auch der charakteristischen Elementlänge h_e, sind zusammen mit einer ausführlichen Beschreibung der Eigenschaften und Asymptotik der Stabilisierungsparameter in FÖRSTER (2007) zu finden. Für die in dieser Arbeit vorgestellten Simulationen wurden die Konstanten zu $c_1 = 2.0$ und $c_2 = 1.0$ gewählt.

Einsetzen der Approximationen (2.50) in die semidiskrete schwache Form (2.51) führt auf die semidiskrete Matrixdarstellung des stabilisierten ALE-Finite-Element-Verfahrens:

$$\begin{aligned} \mathbf{M}_{\mathrm{M}}(\mathbf{u})\ \dot{\mathbf{u}} + \mathbf{K}_{\mathrm{M}}(\mathbf{u})\ \mathbf{u} + \mathbf{N}_{\mathrm{M}}(\mathbf{u}) + \mathbf{G}_{\mathrm{M}}(\mathbf{u})\ \mathbf{p} &= \mathbf{r}_{\mathrm{M,b}}(\mathbf{u}) + \mathbf{r}_{\mathrm{M,h}} \\ \mathbf{M}_{\mathrm{C}}\ \dot{\mathbf{u}} + \mathbf{K}_{\mathrm{C}}(\mathbf{u})\ \mathbf{u} + \mathbf{C}\ \mathbf{p} &= \mathbf{r}_{\mathrm{C}}. \end{aligned} \tag{2.58}$$

Die meisten hierin verwendeten Matrizen und Vektoren auf der rechten Seite enthalten sowohl Terme aus der reinen Galerkin-Approximation als auch Anteile aus der Stabilisierung. In den folgenden Formeln stehen die Galerkin-Terme immer in der ersten Zeile, die Anteile aus der Stabilisierung in den darauffolgenden Zeilen.

Der Energieausdruck mit der Massenmatrix $\mathbf{M}_{\mathrm{M}}(\mathbf{u})$ lautet für das stabilisierte Verfahren:

$$\begin{aligned} \mathbf{v}^{\mathrm{T}}\mathbf{M}_{\mathrm{M}}(\mathbf{u})\dot{\mathbf{u}} = {} & (\dot{\mathbf{u}}_{\mathrm{x}}^{\mathrm{h}}, \mathbf{v}_{\mathrm{x}}^{\mathrm{h}})_{\Omega_{\mathrm{F}}} \\ & + \sum_e \tau_{\mathrm{M}e}(\dot{\mathbf{u}}_{\mathrm{x,e}}^{\mathrm{h}}, \mathbf{c}_{\mathrm{x,e}}^{\mathrm{h}} \cdot \nabla \mathbf{v}_{\mathrm{x,e}}^{\mathrm{h}})_{\Omega_e} \\ & + \alpha \sum_e \tau_{\mathrm{M}e}(\dot{\mathbf{u}}_{\mathrm{x,e}}^{\mathrm{h}}, 2\nu_{\mathrm{F}} \nabla \cdot \varepsilon(\mathbf{v}_{\mathrm{x,e}}^{\mathrm{h}}))_{\Omega_e}. \end{aligned} \tag{2.59}$$

In der Matrix $\mathbf{K}_{\mathrm{M}}(\mathbf{u})$ werden alle viskosen Anteile der stabilisierten Impulsbilanz zusammengefasst:

$$\begin{aligned} \mathbf{v}^{\mathrm{T}}\mathbf{K}_{\mathrm{M}}(\mathbf{u})\mathbf{u} = {} & (2\nu_{\mathrm{F}}\varepsilon(\mathbf{u}_{\mathrm{x}}^{\mathrm{h}}), \varepsilon(\mathbf{v}_{\mathrm{x}}^{\mathrm{h}}))_{\Omega_{\mathrm{F}}} \\ & - \sum_e \tau_{\mathrm{M}e}(2\nu_{\mathrm{F}} \nabla \cdot \varepsilon(\mathbf{u}_{\mathrm{x,e}}^{\mathrm{h}}), \mathbf{c}_{\mathrm{x,e}}^{\mathrm{h}} \cdot \nabla \mathbf{v}_{\mathrm{x,e}}^{\mathrm{h}})_{\Omega_e} \\ & - \alpha \sum_e \tau_{\mathrm{M}e}(2\nu_{\mathrm{F}} \nabla \cdot \varepsilon(\mathbf{u}_{\mathrm{x,e}}^{\mathrm{h}}), 2\nu_{\mathrm{F}} \nabla \cdot \varepsilon(\mathbf{v}_{\mathrm{x,e}}^{\mathrm{h}}))_{\Omega_e} \\ & + \sum_e \tau_{\mathrm{C}e}(\nabla \cdot \mathbf{u}_{\mathrm{x,e}}^{\mathrm{h}}, \nabla \cdot \mathbf{v}_{\mathrm{x,e}}^{\mathrm{h}})_{\Omega_e}, \end{aligned} \tag{2.60}$$

wohingegen die konvektiven Anteile in der Matrix $\mathbf{N}_{\mathrm{M}}(\mathbf{u})$ stehen:

$$\begin{aligned}\mathbf{v}^{\mathrm{T}}\mathbf{N}_{\mathrm{M}}(\mathbf{u}) = &(\mathbf{c}_{\mathrm{x}}^{\mathrm{h}} \cdot \nabla \mathbf{u}_{\mathrm{x}}^{\mathrm{h}}, \mathbf{v}_{\mathrm{x}}^{\mathrm{h}})_{\Omega_{\mathrm{F}}} \\ &+ \sum_e \tau_{\mathrm{M}e}(\mathbf{c}_{\mathrm{x,e}}^{\mathrm{h}} \cdot \nabla \mathbf{u}_{\mathrm{x,e}}^{\mathrm{h}}, \mathbf{c}_{\mathrm{x,e}}^{\mathrm{h}} \cdot \nabla \mathbf{v}_{\mathrm{x,e}}^{\mathrm{h}})_{\Omega_e} \\ &+ \alpha \sum_e \tau_{\mathrm{M}e}(\mathbf{c}_{\mathrm{x,e}}^{\mathrm{h}} \cdot \nabla \mathbf{u}_{\mathrm{x,e}}^{\mathrm{h}}, 2\nu_{\mathrm{F}} \nabla \cdot \varepsilon(\mathbf{v}_{\mathrm{x,e}}^{\mathrm{h}}))_{\Omega_e}.\end{aligned} \tag{2.61}$$

Der semidiskrete stabilisierte Gradienten-Operator ergibt sich zu:

$$\begin{aligned}\mathbf{q}^{\mathrm{T}}\mathbf{K}_{\mathrm{C}}(\mathbf{u})\mathbf{u} = &-(\nabla \cdot \mathbf{u}_{\mathrm{x}}^{\mathrm{h}}, q_{\mathrm{x}}^{\mathrm{h}})_{\Omega_{\mathrm{F}}} \\ &+ \sum_e \tau_{\mathrm{M}e}(\mathbf{c}_{\mathrm{x,e}}^{\mathrm{h}} \cdot \nabla \mathbf{u}_{\mathrm{x,e}}^{\mathrm{h}}, \nabla q_{\mathrm{x,e}}^{\mathrm{h}})_{\Omega_e} \\ &- \sum_e \tau_{\mathrm{M}e}(2\nu_{\mathrm{F}} \nabla \cdot \varepsilon(\mathbf{u}_{\mathrm{x,e}}^{\mathrm{h}}), \nabla q_{\mathrm{x,e}}^{\mathrm{h}})_{\Omega_e},\end{aligned} \tag{2.62}$$

und der entsprechende Divergenz-Operator lautet:

$$\begin{aligned}\mathbf{v}^{\mathrm{T}}\mathbf{G}_{\mathrm{M}}(\mathbf{u})\mathbf{p} = &-(p_{\mathrm{x}}^{\mathrm{h}}, \nabla \cdot \mathbf{v}_{\mathrm{x}}^{\mathrm{h}})_{\Omega_{\mathrm{F}}} \\ &+ \sum_e \tau_{\mathrm{M}e}(\nabla p_{\mathrm{x,e}}^{\mathrm{h}}, \mathbf{c}_{\mathrm{x,e}}^{\mathrm{h}} \cdot \nabla \mathbf{v}_{\mathrm{x,e}}^{\mathrm{h}})_{\Omega_e} \\ &+ \alpha \sum_e \tau_{\mathrm{M}e}(\nabla p_{\mathrm{x,e}}^{\mathrm{h}}, 2\nu_{\mathrm{F}} \nabla \cdot \varepsilon(\mathbf{v}_{\mathrm{x,e}}^{\mathrm{h}}))_{\Omega_e}.\end{aligned} \tag{2.63}$$

Die rechte Seite der schwachen Impulsbilanz setzt sich aus zwei Anteilen zusammen. Der erste enthält die Volumenlasten $\hat{\mathbf{b}}_{\mathrm{F}}$ und wird mit $\mathbf{r}_{\mathrm{M,b}}(\mathbf{u})$ bezeichnet:

$$\begin{aligned}\mathbf{v}^{\mathrm{T}}\mathbf{r}_{\mathrm{M,b}}(\mathbf{u}) = &(\hat{\mathbf{b}}_{\mathrm{F}}, \mathbf{v}_{\mathrm{x}}^{\mathrm{h}})_{\Omega_{\mathrm{F}}} \\ &+ \sum_e \tau_{\mathrm{M}e}(\hat{\mathbf{b}}_{\mathrm{F}}, \mathbf{c}_{\mathrm{x,e}}^{\mathrm{h}} \cdot \nabla \mathbf{v}_{\mathrm{x,e}}^{\mathrm{h}})_{\Omega_e} \\ &+ \alpha \sum_e \tau_{\mathrm{M}e}(\hat{\mathbf{b}}_{\mathrm{F}}, 2\nu_{\mathrm{F}} \nabla \cdot \varepsilon(\mathbf{v}_{\mathrm{x,e}}^{\mathrm{h}}))_{\Omega_e}.\end{aligned} \tag{2.64}$$

Der zweite Anteil enthält die Randlasten des Fluids und beinhaltet keine Terme aus der Stabilisierung:

$$\mathbf{v}^{\mathrm{T}}\mathbf{r}_{\mathrm{M,h}} = (\hat{\mathbf{h}}, \mathbf{v}_{\mathrm{x}}^{\mathrm{h}})_{\Gamma_{\mathrm{h}}}. \tag{2.65}$$

Aufgrund der Stabilisierung entstehen einige Matrizen, die bei einer reinen Galerkin-Formulierung nicht benötigt werden. Dies sind ein Massenmatrix-Anteil für die Konti-

nuitätsgleichung

$$\mathbf{q}^{\mathrm{T}}\mathbf{M}_{\mathrm{C}}\dot{\mathbf{u}} = \sum_{e} \tau_{\mathrm{M}e}(\dot{\mathbf{u}}^{\mathrm{h}}_{\mathrm{x,e}}, \nabla q^{\mathrm{h}}_{\mathrm{x,e}})_{\Omega_e}, \tag{2.66}$$

die Matrix zur Druckstabilisierung

$$\mathbf{q}^{\mathrm{T}}\mathbf{C}\mathbf{p} = \sum_{e} \tau_{\mathrm{M}e}(\nabla p^{\mathrm{h}}_{\mathrm{x,e}}, \nabla q^{\mathrm{h}}_{\mathrm{x,e}})_{\Omega_e} \tag{2.67}$$

und ein Beitrag zur rechten Seite der Kontinuitätsgleichung

$$\mathbf{q}^{\mathrm{T}}\mathbf{r}_{\mathrm{C}} = \sum_{e} \tau_{\mathrm{M}e}(\hat{\mathbf{b}}_{\mathrm{F}}, \nabla q^{\mathrm{h}}_{\mathrm{x,e}})_{\Omega_e}. \tag{2.68}$$

2.2.4 Diskretisierung in der Zeit

In der Zeit stellen die inkompressiblen Navier-Stokes-Gleichungen ein gekoppeltes Problem aus einer partiellen Differenzialgleichung und der Nebenbedingung der Inkompressibilität dar. Die Inkompressibilität führt zu infinitesimal kleinen charakteristischen Zeitskalen, die durch die stabilisierte räumliche Diskretisierung ein wenig relaxiert werden. Die verbleibenden sehr kleinen, aber finiten Zeitskalen sind für gewöhnliche Differenzialgleichungen in der Zeit ein Anzeichen für eine sehr große Steifigkeit. Daher werden hier implizite Zeitintegrationsverfahren verwendet, mit denen eine Genauigkeit von zweiter Ordnung auf dem zeitveränderlichen Gebiet erreicht werden soll.

Einschritt-θ-Verfahren

Das Einschritt-θ-Verfahren gehört zu den am häufigsten verwendeten impliziten Zeitintegrationsverfahren. Auf eine allgemeine Differenzialgleichung erster Ordnung $\dot{y} = f(y,t)$ angewendet lautet es

$$\frac{y_{n+1} - y_n}{\Delta t} = \theta f(y_{n+1}, t_{n+1}) + (1-\theta) f(y_n, t_n), \tag{2.69}$$

wobei Δt den Zeitschritt bezeichnet. Es ist für den Bereich $1/2 \leq \theta \leq 1$ A-stabil und enthält eine Reihe von Spezialfällen, die auch unter eigenen Namen bekannt sind. Mit $\theta = 1$ führt dies auf das Euler-Rückwärts-Verfahren („Backward Euler“, BE) und mit $\theta = 1/2$ auf die Trapezregel (TR), die zweiter Ordnung genau ist und keine künstliche Dämpfung aufweist. Dadurch entstehen jedoch häufig ungewollte Oszillationen, die durch Störungen und schlecht gestellte Anfangsbedingungen verursacht werden können.

BDF2

Die zweite Ordnung genaue Variante der „backward differentiation formulae“ (BDF2) ist A-stabil und enthält geringe numerische Dissipation, die auftretende Störungen und Fehler herausdämpft. Angewandt auf eine allgemeine Differenzialgleichung erster Ordnung lautet das Verfahren:

$$\frac{y_{n+1} - y_n}{\Delta t} = \frac{1}{3}\frac{y_n - y_{n-1}}{\Delta t} + \frac{2}{3} f(y_{n+1}, t_{n+1}). \tag{2.70}$$

Der Schreibweise von Förster (2007) folgend, lassen sich beide vorgestellten Verfahren in einer einheitlichen Form aufschreiben. Die Anwendung der Gleichungen (2.69) oder (2.70) auf die semidiskrete Impulsbilanz und Kontinuitätsgleichung (2.58) führt auf die diskrete Form des stabilisierten ALE-Finite-Element-Verfahrens:

$$\begin{aligned} \mathbf{M}_{\mathrm{M}}(\mathbf{u}_{n+1})\mathbf{u}_{n+1} + \delta\left(\mathbf{K}_{\mathrm{M}}(\mathbf{u}_{n+1})\mathbf{u}_{n+1} + \mathbf{N}_{\mathrm{M}}(\mathbf{u}_{n+1}) + \mathbf{G}_{\mathrm{M}}(\mathbf{u}_{n+1})\mathbf{p}_{n+1}\right) &= \mathbf{f}_{\mathrm{M}} \\ \mathbf{M}_{\mathrm{C}}\mathbf{u}_{n+1} + \delta\left(\mathbf{K}_{\mathrm{C}}(\mathbf{u}_{n+1})\mathbf{u}_{n+1} + \mathbf{C}\mathbf{p}_{n+1}\right) &= \mathbf{f}_{\mathrm{C}}. \end{aligned} \tag{2.71}$$

Hierbei bezeichnet δ einen Skalar, der vom verwendeten Zeitintegrations-Verfahren abhängt:

$$\delta_\theta = \theta\Delta t; \qquad\qquad \delta_{\mathrm{BDF2}} = \frac{2}{3}\Delta t. \tag{2.72}$$

Die rechte Seite enthält neben den Lastanteilen $\mathbf{r}_{\mathrm{M,b}}$, $\mathbf{r}_{\mathrm{M,h}}$ und $\mathbf{r}_{\mathrm{C}}$ zusätzlich Geschichtsterme der Geschwindigkeit. Für das Einschritt-θ-Verfahren ist dies neben der Geschwindigkeit des letzten Zeitschritts $\mathbf{u}_n$ auch die Beschleunigung $\dot{\mathbf{u}}_n$:

$$\begin{aligned} \mathbf{f}_{\mathrm{M}} &= \delta_\theta(\mathbf{r}_{\mathrm{M,b}}(\mathbf{u}_{n+1}) + \mathbf{r}_{\mathrm{M,h}}) + \mathbf{M}_{\mathrm{M}}(\mathbf{u}_n)\mathbf{u}_n + (1-\theta)\Delta t\mathbf{M}_{\mathrm{M}}(\mathbf{u}_n)\dot{\mathbf{u}}_n \\ \mathbf{f}_{\mathrm{C}} &= \delta_\theta\mathbf{r}_{\mathrm{C}} + \mathbf{M}_{\mathrm{C}}(\mathbf{u}_n)\mathbf{u}_n + (1-\theta)\Delta t\mathbf{M}_{\mathrm{C}}(\mathbf{u}_n)\dot{\mathbf{u}}_n. \end{aligned} \tag{2.73}$$

Beim BDF2-Verfahren werden neben den Geschwindigkeiten des letzten Zeitschritts $\mathbf{u}_n$ auch die des vorletzten Schritts benötigt $\mathbf{u}_{n-1}$:

$$\begin{aligned} \mathbf{f}_{\mathrm{M}} &= \delta_{\mathrm{BDF2}}(\mathbf{r}_{\mathrm{M,b}}(\mathbf{u}_{n+1}) + \mathbf{r}_{\mathrm{M,h}}) + \frac{4}{3}\mathbf{M}_{\mathrm{M}}(\mathbf{u}_n)\mathbf{u}_n - \frac{1}{3}\mathbf{M}_{\mathrm{M}}(\mathbf{u}_{n-1})\mathbf{u}_{n-1} \\ \mathbf{f}_{\mathrm{C}} &= \delta_{\mathrm{BDF2}}\mathbf{r}_{\mathrm{C}} + \frac{4}{3}\mathbf{M}_{\mathrm{C}}(\mathbf{u}_n)\mathbf{u}_n - \frac{1}{3}\mathbf{M}_{\mathrm{C}}(\mathbf{u}_{n-1})\mathbf{u}_{n-1}. \end{aligned} \tag{2.74}$$

Aufgrund seiner Robustheit und Genauigkeit wird das BDF2-Verfahren in dieser Arbeit bevorzugt und in allen Beispielen verwendet.

2.2.5 Iteratives Lösungsverfahren

Analog zur Strukturdynamik wird das resultierende nichtlineare Gleichungssystem (2.71) in jedem Zeitschritt $[t_n; t_{n+1}]$ mit dem Newton-Raphson-Verfahren gelöst.

Hierbei wird die Linearisierung von allen Stabilisierungstermen berücksichtigt, allein die beiden Stabilisierungsparameter τ_{M} und τ_{C} sind nicht linearisiert. Für Problemstellungen, bei denen die Stabilisierungsparameter nahezu konstant bleiben, wird damit eine quadratische Konvergenz im Fluid-Löser erreicht.

2.2.6 Reduktion auf den Kopplungsrand

Für den Fall der Fluid-Struktur-Wechselwirkung entspricht der an die flexible Struktur grenzende Rand des Fluids auch dem Kopplungsrand Γ. Dieser Teil der Berandung kann, je nach Kopplungsalgorithmus, eine Untermenge des Neumann-Rands ($\Gamma \subset \Gamma_{\mathrm{h}}$) oder des Dirichlet-Rands sein ($\Gamma \subset \Gamma_{\mathrm{u}}$).

Im Folgenden wird, wie schon für die Struktur, ein Operator eingeführt, der die in den vorherigen Abschnitten beschriebene iterative Lösung der nichtlinearen Fluiddynamik zusammenfasst. Dazu werden die Geschwindigkeiten und Drücke in die Freiheitsgrade auf dem Kopplungsrand $(\bullet)_\Gamma$ und alle übrigen Freiheitsgrade $(\bullet)_{\mathrm{I}}$ aufgeteilt:

$$\mathbf{u} = \begin{pmatrix} \mathbf{u}_{\mathrm{I}} \\ \mathbf{u}_\Gamma \end{pmatrix}; \qquad \mathbf{p} = \begin{pmatrix} \mathbf{p}_{\mathrm{I}} \\ \mathbf{p}_\Gamma \end{pmatrix}. \tag{2.75}$$

Für die in dieser Arbeit vorgestellten Kopplungsalgorithmen ist der Kopplungsrand ein Teil des Dirichlet-Rands, d.h. es werden die Kräfte am Kopplungsrand gesucht, die sich aus einer Geschwindigkeits-Randbedingung auf diesem Rand ergeben. Zusätzlich muss die neue Lage des Fluidgebiets berücksichtigt werden. Die Lösung des Fluids auf dem neuen Gebiet mit den oben angegebenen Randbedingungen zusammen mit der Nachlaufrechnung zur Berechnung der Kräfte, die auf den Kopplungsrand wirken, wird in dem nichtlinearen Operator $\mathcal{F}$ zusammengefasst:

$$\mathbf{f}_{\Gamma,n+1} = \mathcal{F}_{n+1}(\mathbf{u}_{\Gamma,n+1}, \mathbf{d}_{\mathrm{F},n+1}). \tag{2.76}$$

Hierin bezeichnet $\mathbf{f}_{\Gamma,n+1}$ die Kopplungskräfte, die vom Fluid auf die Struktur wirken, und $\mathbf{d}_{\mathrm{F},n+1}$ die Verschiebungen aller Knoten im Fluidgebiet. Die Ermittlung der neuen Positionen aller Knoten im Fluidgebiet wird im folgenden Abschnitt 2.3 thematisiert.

2.3 Netzdynamik

Wie im vorherigen Abschnitt bereits beschrieben, wird zur Simulation von Strömungen auf zeitveränderlichen Gebieten ein ALE-Ansatz verwendet. Für die numerische Umsetzung ist es erforderlich, die Netzverschiebung, die durch die Bewegung der Struktur am Kopplungsrand initiiert wird, für das gesamte ALE-Gebiet zu bestimmen. Sowohl die Netzqualität als auch die Netzgeschwindigkeit haben einen wesentlichen Einfluss auf die Qualität der Ergebnisse der Strömungssimulation auf einem bewegten Netz. Daher werden folgende Anforderungen an den Netzbewegungsalgorithmus gestellt:

- Vermeidung von extremen Elementverzerrungen
- glatte Verteilung der Netzgeschwindigkeit
- Erhalt von Grenzschichtnetzen und anderen Netzcharakteristika
- keine Benutzerinteraktion.

Generell kann zwischen Netzbewegungsalgorithmen unterschieden werden, die speziell für ein bestimmtes Problem konzipiert wurden, und solchen Ansätzen, die für beliebige Probleme eingesetzt werden können. Problemspezifische Ansätze können zum Beispiel einfache Interpolationen der Randverschiebungen entlang der Normalen des Kopplungsrands sein oder diskontinuierliche Ansätze, die bei großen aber vorherbestimmten Deformationen verwendet werden. Beispiele hierfür sind rotierende Bauteile, wie Turbinen und Helikopter-Rotoren, oder rein translatorische bewegte Körper, wie Fahrzeuge. Eine ausführliche Übersicht über verschiedene Methoden zur Netzbewegung wird in Wall (1999) gegeben.

In dieser Arbeit wird hauptsächlich ein sogenannter Pseudo-Struktur-Ansatz verwendet. Dabei wird das Finite-Element-Netz als eine kontinuierliche linear-elastische Pseudo-Struktur betrachtet. Die Steifigkeit des Netzes kann lokal variiert werden, um jeweils problemspezifisch die Netzqualität zu verbessern. Die Bestimmung der neuen Netzposition erfordert in jedem Zeitschritt die Lösung eines zusätzlichen Randwertproblems der Form

$$\mathbf{K}_{\mathrm{N}}\mathbf{d}_{\mathrm{N},n+1} = \mathbf{f}_{\mathrm{N},n+1}. \tag{2.77}$$

Dabei werden Dirichlet-Randbedingungen $\hat{\mathbf{d}}_{n+1}$ auf dem kompletten Rand Γ_N des Netzgebiets Ω_N vorgegeben, woraus die einzigen Anteile des Lastvektors $\mathbf{f}_\mathrm{N}$ entstehen. Auf dem Kopplungsrand Γ wird die Position des benetzen Strukturrands $\mathbf{d}_{\Gamma,n+1}$ vorgeschrieben, am restlichen Rand soll sich das Netz nicht verschieben.

$$\hat{\mathbf{d}}_{n+1} = \mathbf{d}_{\Gamma,n+1} \qquad \text{auf } \Gamma \qquad\qquad \hat{\mathbf{d}}_{n+1} = \mathbf{0} \qquad \text{auf } \Gamma_\mathrm{N} \setminus \Gamma \tag{2.78}$$

Die verformte Lage der Pseudo-Struktur entspricht dann in jedem Zeitschritt der Position des Fluidnetzes:

$$\mathbf{d}_{\mathrm{F},n+1} = \mathbf{d}_{\mathrm{N},n+1}. \tag{2.79}$$

Aufgrund der linearen Formulierung ist die Steifigkeitsmatrix der Pseudo-Struktur im Verlauf der gesamten Berechnung unveränderlich, so dass sie nur ein einziges Mal faktorisiert werden muss und die Lösung in jedem Zeitschritt dann durch einfaches Rückwärtseinsetzen bestimmt werden kann.

Zur vereinfachten Darstellung der Kopplungsalgorithmen wird auch die Bestimmung der neuen Position des Fluidnetzes in einer Operatorschreibweise zusammengefasst. In jedem Zeitschritt berechnet der lineare Operator $\mathcal{N}$ aus der Position der Struktur am Kopplungsrand $\mathbf{d}_{\Gamma,n+1}$ die Lage aller Knoten des Fluidnetzes $\mathbf{d}_{\mathrm{F},n+1}$:

$$\mathbf{d}_{\mathrm{F},n+1} = \mathcal{N}_{n+1}(\mathbf{d}_{\Gamma,n+1}). \tag{2.80}$$

3

Strömungssimulation durch Hochleistungsrechnen

Hochleistungsrechnen („high performance computing“, HPC) bezeichnet den Bereich des computergestützten Rechnens, bei dem eine hohe Rechenleistung oder Speicherkapazität erforderlich ist. Eine exakte Abgrenzung des Hochleistungsrechnens von anderen Bereichen des computergestützten Rechnens durch dauerhafte formale Kriterien ist unter anderem aufgrund der schnellen Entwicklung der Rechentechnik nicht möglich. Allgemein anerkannt ist, dass Rechenanwendungen in den Bereich des Hochleistungsrechnens fallen, deren Berechnung auf einfachen Arbeitsplatzrechnern aufgrund ihrer Komplexität oder ihres Umfangs nicht möglich ist (Wikipedia 2008).

Hochleistungsrechnen ist auf verschiedenen Rechnerarchitekturen möglich. Ein gemeinsames Merkmal dieser Rechner ist, dass sie (zum Teil massiv) parallel arbeiten, d.h. die zu bearbeitende Aufgabe wird aufgeteilt und die Teilaufgaben zeitgleich auf mehreren Prozessoren gelöst. Dabei gibt es hauptsächlich zwei unterschiedliche Konstruktionsansätze. Zum Einen gibt es Cluster, die aus einer großen Anzahl von vernetzten Computern bestehen. Sie sind untereinander über ein schnelles Netzwerk verbunden, müssen aber nicht unbedingt an einem Ort stehen (verteiltes Rechnen). In jüngerer Zeit werden sie immer häufiger aus preiswerten, seriengefertigten PC-Komponenten zusammengesetzt.

Zum Anderen gibt es parallele Hochleistungsrechner, die meistens als Vektorrechner konzipiert sind. Sie erreichen ihre hohe Rechenleistung durch maximale Ausnutzung der verfügbaren Technik mithilfe von teuren Spezialkomponenten. Im Vergleich zu Clustern benötigen sie weniger Prozessoren, um eine entsprechende Leistung zu erzielen. Parallele Vektorrechner gelten in vielen Bereichen als die schnellsten verfügbaren Computer (Löhner und Galle 2002).

Vor allem im wissenschaftlichen Rechnen gewinnt das Hochleistungsrechnen zunehmend an Bedeutung als Hilfsmittel zur Berechnung, Modellierung und Simulation komplexer Systeme und zur Verarbeitung riesiger Messdatenmengen. Derartige Anwendungen finden sich heute in praktisch allen Bereichen der Natur- und Ingenieurwissenschaften. Typische Anwendungen sind die Meteorologie und Klimatologie, Astro- und Teilchenphysik, Strömungsmechanik sowie die Systembiologie, Genetik und Quantenchemie.

Besonders beim „computational steering“ sind Hochleistungsrechner unabdingbar. Ziel ist es numerische Simulationen interaktiv zu steuern, d.h. die Ergebnisse direkt z.B. in einer „virtual reality“ zu betrachten und über Veränderungen an Randbedingungen den Fortgang der Simulation zu beeinflussen. Dabei ist es erforderlich, dass die Reaktion der Simulation auf neue Randbedingungen nahezu in Echtzeit geschieht. Ein Beispiel für „computational steering“ ist eine interaktive Simulation zur Beurteilung des thermischen Behaglichkeitsempfindens in Innenräumen (Wenisch u. a. 2007).

Auch in der Industrie ist das Hochleistungsrechnen mittlerweile weit verbreitet. Die Anwendungsgebiete (Wettervorhersage, Crashtestsimulation, Strömungssimulation im Flugzeugbau) entsprechen weitestgehend denen des wissenschaftlichen Rechnens. Auch die Generierung von Animationsfilmen bedarf des Hochleistungsrechnens.

Die numerische Simulation von Strömungen kann eine kostengünstige Alternative oder Ergänzung zu aufwändigen Windkanalversuchen sein. Dafür ist es erforderlich, dass realitätsnahe Berechnungen von Strömungen möglich sind. Viele Anwendungsbereiche der numerischen Strömungssimulation erfordern aber aufgrund der vorhandenen Skalenunterschiede und der transienten Natur von Strömungen Berechnungen mit sehr vielen Freiheitsgraden und Zeitschritten. Die Größenordnung dieser Probleme sprengt noch in vielen Fällen die heute verfügbaren Rechenkapazitäten und erfordert daher auch besondere Beachtung bei der Implementierung der Algorithmen.

Beim Hochleistungsrechnen sind zwei Komponenten von Bedeutung: Neben der notwendigen modernen Computerhardware, die die erforderliche Rechenleistung einschließlich Speicher zur Verfügung stellt, sind die verwendeten Algorithmen ebenso wichtig. Die Algorithmen müssen einerseits in der Lage sein, komplexe, realitätsnahe Probleme genau zu simulieren, andererseits auch so effizient sein, dass die vorhandene Hardware möglichst gut genutzt und die Simulation in angemessener Zeit beendet wird.

Hierbei gibt es eine Wechselwirkung zwischen dem Hochleistungsrechnen und der numerischen Strömungssimulation. Während die modernen Hochleistungsrechner die realitätsnahe Simulation von Strömungen erst ermöglicht haben, treiben die immer höheren Anforderungen, die der Ingenieur an die Simulation stellt, die Entwicklung von neuen Hochleistungsrechentechniken (sowohl Algorithmen als auch Hardware) voran (Tezduyar u. a. 1996). Die Anwender erwarten heute genaue und verlässliche Ergebnisse

mit kurzen Rechenzeiten (Zheng und Liou 2003). Jedoch wird dies erst durch den Einsatz von Hochleistungsrechnern und vektorisierten und parallelisierten Berechnungsprogrammen bei numerischen Simulationen von komplexen Systemen und Strukturen, wie im Bereich der Fluid-Struktur-Wechselwirkung, ermöglicht (Katz u. a. 2000).

Im folgenden Abschnitt werden zunächst einige Begriffe aus dem Hochleistungsrechnen erläutert und die im Rahmen dieser Arbeit hauptsächlich verwendeten Vektorprozessoren vorgestellt. Im Anschluss daran werden aus drei Bereichen der numerischen Strömungssimulation Methoden vorgestellt, die es ermöglichen, realitätsnahe, sehr große Simulationen durchzuführen. Da für die in dieser Arbeit betrachteten Probleme das Fluidfeld am rechenaufwändigsten ist, wird dieses hier beispielhaft betrachtet. Die vorgestellten Techniken lassen sich aber ohne Schwierigkeiten auf andere Problemklassen übertragen.

3.1 Grundlagen des Hochleistungsrechnens

3.1.1 Rechengeschwindigkeit messen

Die Geschwindigkeit von Prozessoren oder auch Computersystemen wird in Gleitkommaoperationen pro Sekunde („floating point operations per second“ (FLOPS)) gemessen. Eine Gleitkommaoperation entspricht dabei z.B. einer Addition oder Multiplikation. Da manche Prozessoren für eine Gleitkommaoperation mehrere Taktzyklen benötigen, andere aber mehrere Gleitkommaoperationen in einem Takt ausführen können, eignet sich die Angabe der FLOPS zum Vergleich von Prozessorgeschwindigkeiten besser als die Taktfrequenz.

Weiterhin muss zwischen der theoretisch erreichbaren Maximalleistung eines Rechners („peak performance“) und der tatsächlich durch eine spezielle Anwendung verwendeten Leistung („sustained performance“) unterschieden werden. Die „peak performance“ kann in der Praxis nicht vollständig erreicht werden, da das Lesen und Schreiben der Daten aus dem Speicher Taktzyklen benötigt, in denen keine Gleitkommaoperationen ausgeführt werden können. Die größer werdende Diskrepanz zwischen „peak“ und „sustained performance“ ist ein bekanntes Problem im Hochleistungsrechnen (Oliker u. a. 2003; Turek 1999) und hängt sowohl von der Rechnerarchitektur als auch von der Anwendung ab.

Auf Vektorsystemen sind Auslastungen von 30 % und mehr die Regel, bei speziell für Vektorprozessoren optimierten Programmen können auch deutlich mehr als 70 % der „peak performance“ erreicht werden. Cache-basierte Prozessoren zeichnen sich dagegen

durch eine üblicherweise deutlich niedrigere Auslastung aus. Auf diesen Systemen sind Auslastungen von nur 1-5 % keine Seltenheit (Behr u. a. 2000).

3.1.2 Vektorprozessoren

Vektorprozessoren (auch Array-Prozessoren genannt) sind in der Lage eine Berechnung mit vielen Daten in einem Vektor nahezu gleichzeitig auszuführen. In der Vergangenheit wurden sie hauptsächlich für Hochleistungsrechner eingesetzt. Sehr frühe, prominente Vertreter dieser Vektorrechner waren in den 1970er Jahren die Cray-Supercomputer. Lange Zeit führten Vektorrechner die TOP500-Liste der schnellsten Computer weltweit an, so auch bis Juni 2004 der „Earth Simulator“ in Japan.

Wenn viele gleichartige Daten auf dieselbe Weise bearbeitet werden sollen (beispielsweise bei Matrizenoperationen) sind Vektorprozessoren reinen Skalar-Prozessoren, die jeden Wert einzeln nacheinander bearbeiten, weit überlegen. Dies trifft bei vielen Diskretisierungsverfahren zu, bei denen für jeden Punkt oder jedes Element dieselben Operationen ausgeführt werden müssen.

Wegen der Vorteile, die sich durch die gleichzeitige Ausführung einer Rechenoperation auf mehreren Daten ergeben („single instruction, multiple data“; SIMD) wurden auch in Standardprozessoren seit den 1990er Jahren Erweiterungen integriert, die diese Art von Berechnungen beschleunigen. Eine der ersten SIMD-Erweiterungen war MMX für die x86-Architektur von Intel. All diese Erweiterungen der Standardprozessoren orientieren sich an der Unterstützung von Echtzeit-3-D-Graphik-Anwendungen. Gerade aufwändige 3-D-Spiele verlangen sehr viele Berechnungen mit großen Datenmengen (Matrizenoperationen auf 3-D-Koordinaten, Anti-Aliasing der Bildschirmausgabe).

Der SX-8-Prozessor

Im Rahmen dieser Arbeit wurde für alle Anwendungen aus dem Bereich des Hochleistungsrechnens ein paralleler Vektorcomputer SX-8 von NEC genutzt, der am Höchstleistungsrechenzentrum (HLRS) in Stuttgart zur Verfügung steht. Manche der vorgestellten Ergebnisse wurden bereits auf dem Vorgängermodell, einem SX-6+-System erzielt, das sich aber in der grundlegenden Funktionsweise nicht von der SX-8-Plattform unterscheidet.

Ein Knoten der SX-8-Architektur besteht aus 8 CPUs („central processing unit“), die auf einen gemeinsamen Hauptspeicher zugreifen. Daher können die acht CPUs eines Knotens sowohl im „shared memory“- als auch im „distributed memory“-Betrieb genutzt werden. Jede einzelne SX-8-CPU besteht aus einer Skalar-Einheit mit einer Leistung von

1.	Vergleich der Exponenten	1,354e5 - 7,258e4
2.	Shift der Mantisse	1,354e5 - 0,7258e5
3.	Addition der Mantissen	0,6282e5
4.	Auswahl des Exponenten und Normalisierung	6,282e4

Abbildung 3.1: Segmentierung der Subtraktion von zwei Gleitkommazahlen.

2 GFLOPS (Giga-FLOPS) und vier Vektor-Einheiten mit einer Gesamtleistung von 16 GFLOPS. Jede Vektor-Einheit kann mit einer Bandbreite von 16 GB/s Daten aus dem Hauptspeicher in acht Vektor-Register der Länge 256 laden und von dort in vier verschiedenen 4-fach Vektor-Pipelines verarbeiten. Die erste dieser Pipelines führt logische Operationen aus, die zweite Multiplikationen und die dritte Additionen und Shift-Operationen. Eine Besonderheit ist die vierte Pipeline, die Divisionen und Quadratwurzeln berechnen kann (Berger 2005).

Pipeline-Architektur

Pipeline-Architekturen teilen die Abarbeitung von Maschinenbefehlen in mehrere, aufeinanderfolgende Teilaufgaben auf (Segmentierung), die jeweils in einem Takt verarbeitet werden. Dadurch werden für die Bearbeitung eines Befehls zwar mehr Takte benötigt, da die Teilaufgaben aber einfacher und dadurch schneller zu bearbeiten sind, kann im Gegenzug die Taktfrequenz erhöht werden. So kann zum Beispiel die Addition oder Subtraktion von zwei Gleitkommazahlen, wie in Abbildung 3.1 gezeigt, in vier Schritte aufgeteilt werden (Galle und Schoenemeyer 2005). Im ersten Schritt werden die Exponenten der zu summierenden Zahlen verglichen und durch einen Shift der Mantisse wird im zweiten Schritt ein einheitlicher Exponent gewählt. Nun können die Mantissen addiert werden und im letzten Schritt der passende Exponent für das Ergebnis verwendet werden.

Diese Teilaufgaben werden nun im Idealfall für aufeinanderfolgende Befehle, jeweils um eine Teilaufgabenstufe versetzt, gleichzeitig ausgeführt. Dadurch können für das Beispiel der Addition/Subtraktion bis zu vier Befehle parallel in Bearbeitung sein, so dass in jedem Takt ein Befehl fertig gestellt wird (Abbildung 3.2). Zu Beginn des „pipelinings“ dauert es einige Takte (3) bevor der erste Befehl bearbeitet ist. Diese Zeit wird als Latenz bezeichnet. „Pipelining“ ist nur möglich, wenn die Ausführung eines Befehls nicht von dem Ergebnis eines vorherigen Befehls abhängig ist, d.h. wenn Datenparallelität vorliegt.

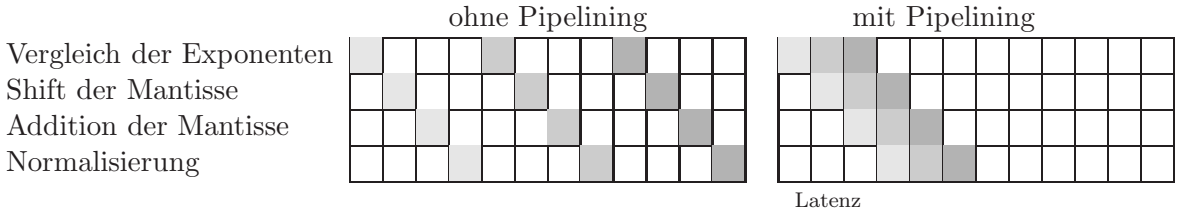


Abbildung 3.2: Pipelining der Subtraktion von zwei Gleitkommazahlen.

3.2 Netzerzeugung in zwei Stufen

3.2.1 Problemstellung

Die Erzeugung eines Netzes für eine Finite-Element-Simulation ist ein sehr zeitaufwändiger und fehleranfälliger Prozess, für den es bis heute noch keine für alle Anwendungsfälle zufriedenstellende Verfahren gibt. Besonders bei der Vernetzung von dreidimensionalen, komplexen Gebieten, wie sie bei der Berechnung von realitätsnahen Aufgabenstellungen auftreten, ist es nicht einfach, zulässige und für die Simulation brauchbare Gitter zu generieren.

Neben den Algorithmen stellt auch die Rechenzeit und der Speicherbedarf, den vorhandene Vernetzer benötigen, bei großen Aufgabenstellungen eine Herausforderung dar. So kann es erforderlich sein, bereits die Erzeugung eines Netzes auf einem parallelen Hochleistungsrechner durchzuführen. Wird auch die Arbeitszeit, die zur Vorbereitung des Netzes erforderlich ist, berücksichtigt, beansprucht in manchen Fällen die Erzeugung des Netzes bis zu 2/3 des gesamten Zeitaufwands einer Simulation (Zheng und Liou 2003).

Aber auch die Weitergabe der Netzdaten vom Präprozessor/Vernetzer zum Löser und von dort die Weitergabe der Ergebnisse zum Postprozessor kann ab einer gewissen Problemgröße unerwartete Schwierigkeiten mit sich bringen. Insbesondere die Ergebnisse einer über viele Zeitschritte laufenden Strömungssimulation umfassen schnell mehrere Gigabyte an Daten. Die Handhabung, Archivierung und speziell die Visualisierung dieser Daten sind extrem zeitaufwändig und beanspruchen große Hardware-Ressourcen. Häufig werden die Ergebnisse aber nicht in derselben Feinheit benötigt, wie sie für die Simulation erforderlich war. Bei der Visualisierung der Ergebnisse am Bildschirm oder im Druck können ohnehin oft nur deutlich weniger Punkte dargestellt werden als in der Berechnung verwendet wurden. Durch den Einsatz des in den folgenden Abschnitten beschriebenen Verfahrens einer Netzerzeugung in zwei Stufen können diese Schwierigkeiten umgangen werden.

Bereits in den Anfängen der Finite-Element-Methode wurden Verfahren entwickelt, bei denen die Erzeugung des Rechengitters in zwei Stufen erfolgte. Die sogenannten „mapping“-Methoden wurden auch als halb-automatische Vernetzer bezeichnet, da im ersten Schritt häufig manuell, später auch automatisch, das Rechengebiet in einfache Teilgebiete unterteilt wurde. Auf diese Teilgebiete konnte dann ein regelmäßiges Netz projiziert werden. Zu den ersten Verfahren dieser Art gehören die „isoparametric mapping method“ von Zienkiewicz und Phillips (1971) und die „transfinite mapping method“ von Gordon und Hall (1973).

Die manuelle Aufteilung des Gebiets in einfache Teilgebiete wird bei dem hier vorgestellten Ansatz durch das automatische Erzeugen eines relativ groben Netzes im Präprozessor ersetzt. Dieses grobe Netz wird im eigentlichen Löser dann nach Belieben verfeinert. Dies kann auch mit relativ wenig Kommunikation zwischen den Rechenprozessen parallel geschehen. Während die Simulation auf dem feinen Netz geschieht, werden die Ergebnisse auf das grobe Gitter projiziert und nur für dieses gespeichert. Die Visualisierung der Ergebnisse erfolgt somit wieder auf dem groben Netz.

Im Folgenden werden zunächst die Vorteile und Einschränkungen des im Rahmen dieser Arbeit entwickelten Verfahrens zur Netzerzeugung in zwei Stufen diskutiert, bevor die Einzelheiten der Umsetzung beschrieben werden.

3.2.2 Vorteile

Neben den bereits erwähnten Vorteilen in Bezug auf die Datenmengen, die zwischen Präprozessor, Löser und Postprozessor ausgetauscht werden müssen, ergeben sich eine Reihe von weiteren Vorteilen aus der Verwendung von zwei unterschiedlich feinen Netzen. So ist einerseits der Rechenaufwand und damit auch der Zeitaufwand, um ein feines Netz direkt mit einem Freivernetzer zu erstellen, deutlich höher, als der Aufwand ein grobes Netz frei zu vernetzen und dann gleichmäßig zu unterteilen.

Andererseits kann das grobe Gitter auch bei der Erzeugung eines zulässigen feinen Netzes hilfreich sein. An dem gröberen Netz ist es bedeutend einfacher Fehler zu finden. So kann ein grobes Gitter leichter optisch auf Fehler wie z.B. Löcher untersucht werden oder die Elementdichte-Verteilung überprüft werden. Außerdem ist es mit dem groben Gitter möglich, bereits am Arbeitsplatzrechner einige Zeitschritte der Simulation zu rechnen. Dadurch können falsch orientierte oder anderweitig irreguläre Elemente gefunden werden und durch Visualisierung der (physikalisch falschen) Ergebnisse die Randbedingungen überprüft werden. Durch eine spätere Verfeinerung der Elemente werden z.B. die Orientierung der Elemente und die Randbedingungen nicht mehr beeinflusst, so dass die Simulation mit dem feinen Netz auf einem Hochleistungsrechner problemlos anlaufen kann.

Zusätzlich ist es möglich aus einem einmal validierten groben Netz verschiedene feine Netze für die Simulation zu erstellen. So können aus einem Grob-Gitter-Element beliebig viele Fein-Gitter-Elemente erstellt (h-Adaption) oder aus jedem Grob-Gitter-Element ein Fein-Gitter-Element mit beliebiger Polynomordnung der Ansatzfunktionen generiert werden (p-Adaption). Natürlich sind auch verschiedene Kombinationen möglich, d.h. aus einem Grob-Gitter-Element können viele Fein-Gitter-Elemente jedes Polynomgrads erstellt werden. Dadurch ist es zum Beispiel möglich ohne viel Aufwand Konvergenzstudien für sowohl die h- als auch die p-Adaption zu erstellen. Außerdem bietet sich dieses Verfahren an, um verschieden feine Netze für eine Mehrskalen-Berechnung z.B. mit der hp-d-Methode (Rank 1992, Krause und Rank 2003) zu erstellen.

Durch eine sukzessive Anwendung der Netzverfeinerung können Netzhierarchien erzeugt werden, die sich als Grundlage von geometrischen Mehrgitter-Verfahren eignen. Unter Verwendung eines Verfahrens zur Behandlung von hängenden Knoten ist es auch möglich das grobe Netz nur in manchen Bereichen stärker zu verfeinern.

3.2.3 Einschränkungen

Der zusätzliche Aufwand, der durch die Vernetzung in zwei Stufen entsteht, ist der entscheidende Nachteil dieses Verfahrens. Während die zusätzlich benötigte Rechenzeit zum Erstellen des feinen Netzes und Transferieren von Daten von einem Netz auf das andere durch die deutlich schnellere Generierung des groben Gitters im Präprozessor kompensiert wird, entsteht ein Mehrbedarf an Arbeitsspeicher. Auch der einmalige Aufwand einer korrekten, effizienten Implementierung muss bei einer Beurteilung der Methode berücksichtigt werden.

Ein großer Nachteil der ursprünglichen „mapping“-Methoden war, wie Zienkiewicz u. a. (2005) schreiben, der geometrische Fehler, der durch die Approximation von gekrümmten Rändern durch das grobe Netz entsteht und bei einer Verfeinerung auf der Basis dieses Netzes auch bei feineren Diskretisierungen nicht reduziert wird.

Diese Problematik lässt sich vermeiden, wenn an den Rändern des Gebiets nicht nur das grobe Netz als Ausgangspunkt der Verfeinerung benutzt wird, sondern zusätzlich die exakte Geometrie des Rechengebiets. Die Umsetzung dieses Ansatzes wird in den folgenden Abschnitten erläutert.

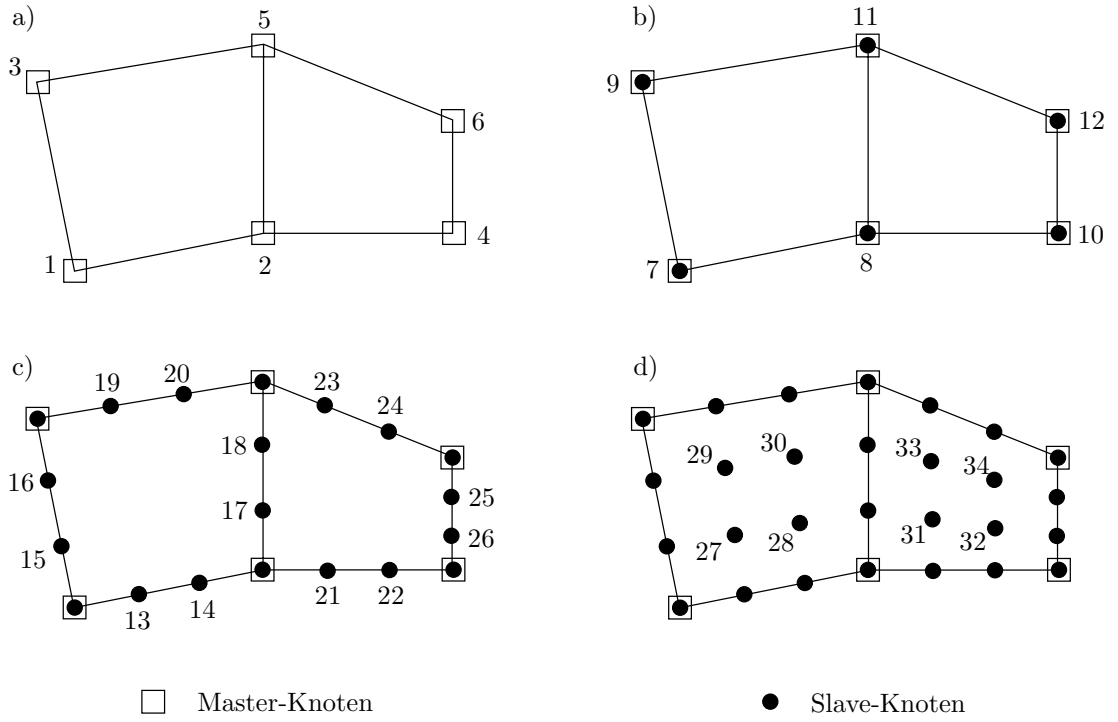


Abbildung 3.3: Element-Unterteilung: Sukzessive Erzeugung der Slave-Knoten.

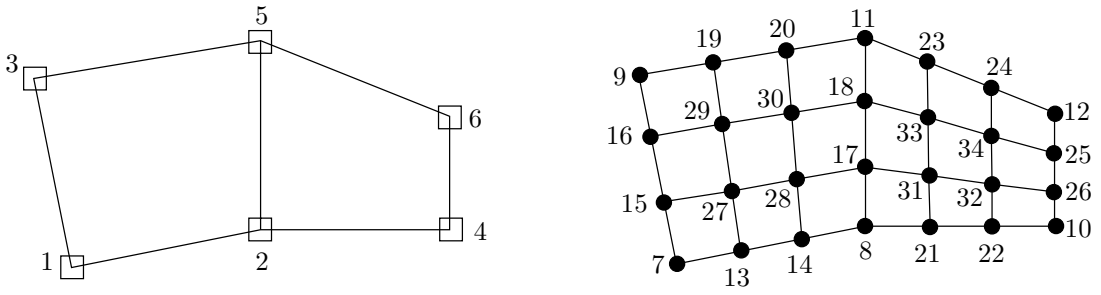
3.2.4 Umsetzung

Im Folgenden wird die grobe Diskretisierung als „Master“ und das feine Netz als „Slave“ gekennzeichnet. Der Faktor für die Netzverfeinerung in jede Koordinatenrichtung wird mit *subdiv* bezeichnet.

Als Beispiel wird hier die Verfeinerung von 4-knotigen Scheibenelementen (Abbildung 3.3 (a)) mit einem Verfeinerungsfaktor $subdiv = 3$ näher beschrieben. Für andere Master-Elemente (Dreiecke, Hexaeder oder Tetraeder) erfolgt das Vorgehen analog.

Um bei der Netzverfeinerung sicher zu stellen, dass bei benachbarten Master-Elementen auch die Slave-Elemente die Knoten auf ihrer gemeinsamen Kante teilen und keine doppelten Knoten erzeugt werden, wurde ein Verfahren implementiert, bei dem für alle topologischen Objekte sukzessive vom 0-D bis zum 3-D zunächst die neuen Knoten erzeugt werden und im Anschluss daran aus diesen Knoten die Slave-Elemente gebildet werden.

Im Einzelnen bedeutet dies, dass zunächst die Master-Knoten auf die neue Diskretisierung kopiert werden. Dabei bleiben die Koordinaten erhalten, die Knotennummern werden um die Anzahl der Master-Knoten erhöht (Abbildung 3.3 (b)). Für die Master-



Master-Diskretisierung Slave-Diskretisierung

Abbildung 3.4: Master- und Slave-Diskretisierung nach der Element-Unterteilung.

Knoten wird ein Verweis auf den jeweiligen Slave-Knoten und umgekehrt gespeichert, so dass beim späteren Übertragen von Ergebnissen keine Suche erforderlich ist.

Danach werden auf allen Elementkanten $(subdiv - 1)$ neue Knoten erzeugt. Liegt eine Elementkante auf der Berandung des Gebiets, werden die Slave-Knoten auf der echten Berandung des Gebiets erzeugt, ansonsten werden sie gleichmäßig auf der geraden Verbindung der beiden Endknoten verteilt (Knoten 13–26 in Abbildung 3.3 (c)).

Im nächsten Schritt werden die Slave-Knoten auf den Elementflächen erzeugt. Auch hier wird, wenn erforderlich, die exakte Geometrie des Gebiets berücksichtigt. Details hierzu sind im folgenden Abschnitt zu finden. Bei der Positionierung der $(subdiv - 1)^2$ Slave-Knoten wird die Geometrie des Master-Elements berücksichtigt (Abbildung 3.3 (d)). Für dreidimensionale Problemstellungen werden im Anschluss auch noch die Slave-Knoten im Inneren der Master-Elemente erzeugt.

Nachdem alle Slave-Knoten erzeugt wurden, werden für jedes Master-Element $subdiv^2$ (bzw. $subdiv^3$) Slave-Elemente generiert. Dabei wird die Orientierung des Master-Elements berücksichtigt, um auf der Slave-Diskretisierung nur reguläre Elemente zu erzeugen. Sind die Knoten des linken Master-Elements in Abbildung 3.4 in der Reihenfolge 1-2-5-3 angeordnet, werden die Slave-Elemente in derselben Orientierung erzeugt, z.B. 7-13-27-15 oder 13-14-28-27. Die Elementeigenschaften, wie Anzahl der Integrationspunkte oder die Elementformulierung, werden dabei vom Master-Element an die Slave-Elemente weitergegeben. Die Slave-Elemente speichern zudem einen Verweis auf ihr Master-Element, der bei späteren „mappings“ von Ergebnissen verwendet werden kann.

Neben der bisher beschriebenen Verfeinerung, bei der die Elementform erhalten bleibt, ist es auch möglich, die Topologie der Elemente oder ihren Grad der Ansatzfunktionen zu verändern. So kann zum Beispiel auf der Basis derselben Slave-Knoten entweder ein verfeinertes Vierecks-Netz oder auch ein Dreiecks-Netz erzeugt werden (Abbildung 3.5 (a)). Sollen höherwertige Ansatzfunktionen verwendet werden, so muss nur bei der Erzeugung der Slave-Knoten darauf geachtet werden, dass für „Serendipity“-Elemente keine Mit-

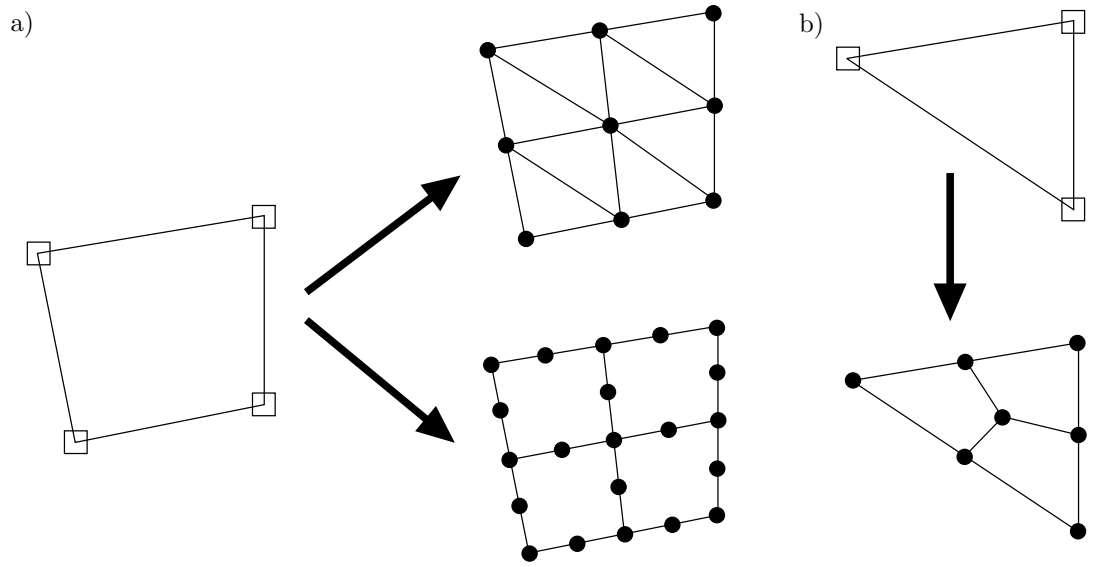


Abbildung 3.5: Element-Unterteilung: Veränderung der Elementtopologie und des Grads der Ansatzfunktionen.

telknoten erzeugt werden dürfen (Abbildung 3.5 (a)). Der übrige Verfeinerungsprozess verläuft analog zu der obigen Beschreibung.

Wird als Master-Diskretisierung ein Dreiecks-Netz gewählt, kann dieses entweder zu Dreiecks-Elementen verfeinert werden, oder aber es werden aus jedem Dreiecks-Element drei Vierecks-Elemente erzeugt, wie in Abbildung 3.5 (b) gezeigt ist. Dabei ist zu beachten, dass die resultierenden Vierecks-Elemente besonders bei spitzwinkligen Dreiecken stark verzerrt sind (Rank u. a. 1993) und dann meistens schlechtere Ergebnisse liefern als unverzerrte Elemente.

Wie aus Abbildung 3.4 zu erkennen ist, bestehen die Slave-Elemente aus Knoten mit zum Teil sehr verschiedenen Knotennummern, z.B. das Slave-Element 7-13-27-15. Dies resultiert aus der Reihenfolge, in der die Slave-Knoten erzeugt werden. Werden die Freiheitsgrade den Knoten entsprechend der Knotennummerierung zugewiesen, führt dies auf eine sehr große Bandbreite der globalen Steifigkeitsmatrix. Für Verfahren (z.B. Sparse-Matrix-Formate und lineare Löser), bei denen die Bandbreite einen Einfluss auf die Rechengeschwindigkeit hat (z.B. direkte Löser), sollte daher die Verteilung der Knotennummern bzw. der Freiheitsgrade nach Erzeugung der Slave-Diskretisierung mithilfe eines Bandbreiten-Optimierers angepasst werden.

3.2.5 Berücksichtigung der exakten Geometrie

Bei der Erzeugung von Knoten auf Kanten bzw. Flächen, die der Berandung des Gebiets entsprechen, werden diese nicht auf der gradlinigen Verbindung der beiden Eckknoten des Master-Elements positioniert, sondern auf dem exakten Rand des Gebiets. Dazu ist es erforderlich, die Beschreibung der Geometrie des Gebiets vom Präprozessor zum Löser, in dem die Unterteilung der Elemente stattfindet, zu übermitteln. Dies geschieht mithilfe derselben Datei, in der auch die grobe Diskretisierung an den Löser übergeben wird. Der Präprozessor GiD (GiD 2008) schreibt dafür Informationen über alle verwendeten geometrischen Objekte in diese Datei.

Auf der untersten Ebene der Hierarchie der geometrischen Objekte sind dies Punkte, die z.B. an Ecken des Gebiets liegen und durch eine fortlaufende Nummer und ihre Koordinaten beschrieben sind. Die Berandung des Berechnungsgebiets wird im 2-D durch Linien beschrieben, die mindestens durch die Nummern ihrer beiden Endpunkte definiert werden. Handelt es sich nicht um eine gerade Linie, wird sie durch einen NURBS beschrieben. Dies erlaubt eine einheitliche Darstellung von sowohl analytisch beschreibbaren Kurven, wie z.B. Kegelschnitten, als auch Freiformkurven. Die Definition von NURBS wird in den folgenden Abschnitten erläutert. Flächen, als nächste Stufe, stellen im 3-D die Berandung des Gebiets dar und werden von GiD zunächst durch die umrandenden Linien beschrieben. Nicht ebene Flächen werden des Weiteren durch NURBS-Flächen beschrieben, die aus der Multiplikation von zwei NURBS-Kurven folgen. Auch die Beschreibung der Volumina, aus denen das Berechnungsgebiet besteht, wird von GiD an den Löser weitergegeben, wird aber bei der Positionierung der neuen Knoten nicht benötigt, da nur auf dem Rand eine exakte Positionierung der Knoten erforderlich ist.

NURBS

In den 1950er Jahren wurden besonders im Automobil- und Schiffsbau für die fehlerfreie Reproduzierbarkeit technischer Bauteile mathematisch exakte Beschreibungen von Freiformflächen benötigt. So begann unter anderem der Ingenieur Pierre Étienne Bézier[1], der zu dieser Zeit bei der Firma Renault in Frankreich arbeitete, mit der Entwicklung der nach ihm benannten Bézierkurve. Unabhängig von Bézier arbeitete Paul de Casteljau[2], angestellt bei Citroën, zur gleichen Zeit ebenfalls an diesem mathematischen Problem. Nach ihm wurde ein Algorithmus benannt, der für die numerische Verarbeitung parametrischer Flächen eingesetzt wird.

[1]Pierre Étienne Bézier, * 1. September 1910, Paris, † 25. November 1999, französischer Ingenieur.
[2]Paul de Faget de Casteljau, * 19. November 1930 in Besançon, französischer Physiker und Mathematiker.

Da die Definition von „non-uniform rational B-splines“ (NURBS) auf der Definition von Bézierkurven und B-Splines basiert, wird hier die Beschreibung von NURBS, Hartmann (2000) folgend, mit der Definition von Bézierkurven begonnen.

Bézierkurven Eine Bézierkurve $C(s)$ vom Grad n ist definiert durch eine parametrische Beschreibung in s:

$$C(s) = \sum_{i=0}^{n} B_{i,n}(s) P_i; \qquad 0 \leq s \leq 1. \tag{3.1}$$

Hierbei bezeichnet P_i die Kontrollpunkte. $B_{i,n}$ sind Bernstein[3]-Polynome n-ten Grads:

$$B_{i,n}(s) = \frac{n!}{i!(n-i)!} s^i (1-s)^{n-i}. \tag{3.2}$$

Bernsteinpolynome sind für alle $s \in [0;1]$ positiv, bilden eine Zerlegung der Eins („partition of unity“), aber erfüllen nur an den Intervallgrenzen die Interpolationseigenschaft.

Da in der Praxis sehr häufig kubische Bézierkurven, die mindestens erforderlich sind, um C^1-Kontinuität zwischen benachbarten Kurven zu erreichen, verwendet werden, werden die dafür notwendigen kubischen Bernstein-Polynome hier beispielhaft gezeigt (Abbildung 3.6).

$$\begin{aligned} B_{0,3}(s) &= 1 - 3s + 3s^2 - s^3 & \qquad B_{0,1}(s) &= 3(s - 2s^2 + s^3) \\ B_{2,3}(s) &= 3(s^2 - s^3) & \qquad B_{3,3}(s) &= s^3 \end{aligned} \tag{3.3}$$

Die Kontrollpunkte P_0 und P_n bilden den Anfangs- und Endpunkt der Kurve. Die Tangenten in diesen beiden Punkten entsprechen dem jeweiligen Segment des Kontrollpunkt-Polygons (Abbildung 3.7):

$$C'(0) = \overrightarrow{P_0 P_1}; \qquad C'(1) = \overrightarrow{P_{n-1} P_n}. \tag{3.4}$$

Ein Nachteil der Approximation mit Bézierkurven ist der globale Einfluss aller Kontrollpunkte auf die gesamte Kurve.

B-Spline B-Splines stellen eine Verallgemeinerung der Bézierkurven dar und können auch als eine Aneinanderkettung mehrerer Bézierkurven mit entsprechender Kontinuität verstanden werden. Durch die Konstruktion spezieller Ansatzfunktionen, die rekursiv

[3]Sergei Natanowitsch Bernstein, * 5. März 1880 in Odessa; † 26. Oktober 1968 in Moskau, russischer Mathematiker.

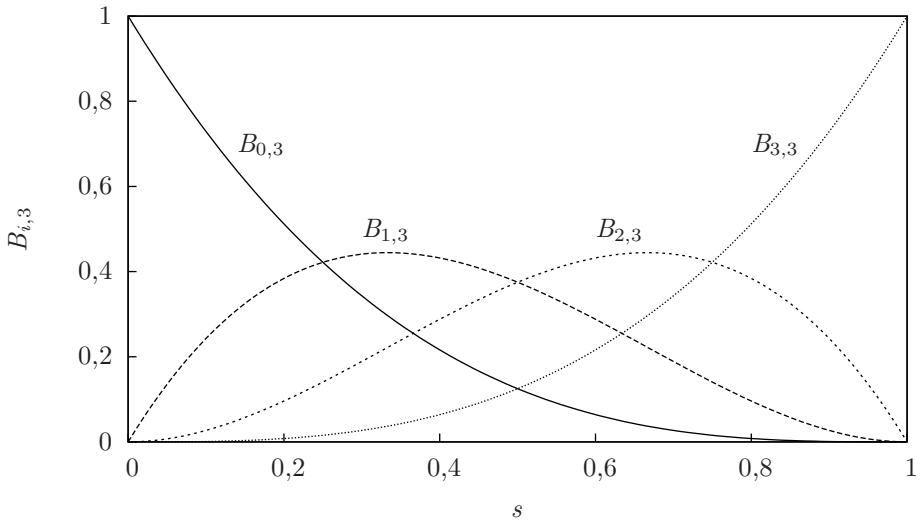


Abbildung 3.6: Kubische Bernstein-Polynome.

definiert werden, bleibt der Einfluss eines Kontrollpunkts auf die Kurve auf einen lokalen Bereich beschränkt.

Ein B-Spline des Grads n wird analog zur Bézierkurve auch in parametrischer Darstellung als Multiplikation der B-Spline-Basisfunktionen $N_{i,n}$ mit den m Kontrollpunkten P_i definiert:

$$C(s) = \sum_{i=0}^{m} N_{i,n}(s) P_i. \tag{3.5}$$

Die B-Spline-Basisfunktionen werden dabei nicht über das gesamte Intervall $[0;1]$ definiert, sondern über eine Folge von geordneten reellen Zahlen, dem sogenannten Knotenvektor $\mathbf{S}_\mathrm{B}$:

$$\mathbf{S}_\mathrm{B} = \left\{ s_0 \leq s_1 \leq \ldots \leq s_{m+n+1} \right\}. \tag{3.6}$$

Die Basisfunktionen vom Grad n sind gegeben durch die Rekursionsformel von de Boor/ Cox/Mansfield (Hartmann 2000):

$$\begin{aligned} N_{i,0}(s) &= \begin{cases} 1 & s_i \leq s \leq s_{i+1} \\ 0 & \text{sonst} \end{cases} \\ N_{i,n}(s) &= \frac{s - s_i}{s_{i+n} - s_i} N_{i,n-1}(s) + \frac{s_{i+n+1} - s}{s_{i+n+1} - s_{i+1}} N_{i+1,n-1}(s). \end{aligned} \tag{3.7}$$

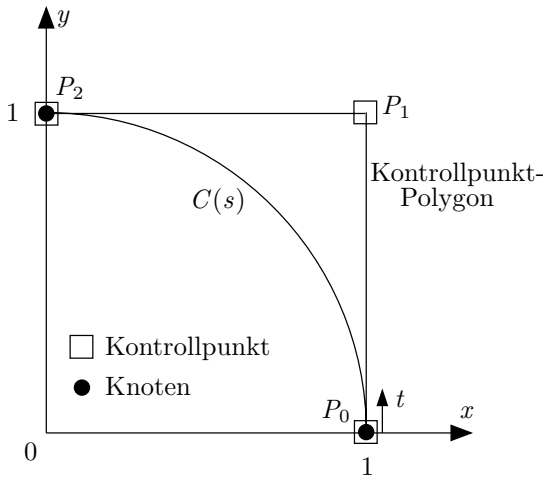


Abbildung 3.7: Parametrische Beschreibung einer Kurve $C(s)$ mit den erforderlichen Kontrollpunkten, Knoten und dem Kontrollpunkt-Polygon.

Bei der Auswertung werden nur Summanden berücksichtigt, deren Nenner ungleich null ist.

Sind die Elemente des Knotenvektors äquidistant, wird von einem „uniform B-spline“ gesprochen. Die Basisfunktionen sehen in diesem Fall alle gleich aus und sind nur entlang der Koordinate s verschoben.

Haben zwei oder mehr benachbarte Knoten denselben Wert, wird dadurch die Wertigkeit w dieses Knotens erhöht. Die Kontinuität an dieser Stelle wird dadurch auf C^{n-w} reduziert. Entspricht die Wertigkeit dem Polynomgrad ($w = n$), so ergeben sich als B-Spline-Basisfunktionen die Bernstein-Polynome. Hierdurch zeigt sich, dass Bézierkurven ein Sonderfall der B-Splines sind.

NURBS Eine Verallgemeinerung der B-Splines stellen die NURBS dar. Mit diesen ist es im Gegensatz zu den B-Splines auch möglich reguläre Geometrien, wie Kegelschnitte, exakt abzubilden. Dafür ist es erforderlich rationale Funktionen als Basisfunktionen einzuführen.

Ein NURBS des Grads n wird somit definiert als:

$$C(s) = \sum_{i=0}^{m} R_{i,n}(s) P_i. \tag{3.8}$$

Hierbei bezeichnet P_i wiederum die m Kontrollpunkte. $R_{i,n}$ sind in diesem Fall die rationalen B-Spline-Basisfunktionen. Diese errechnen sich aus den bereits bekannten

Basisfunktionen und zu den Kontrollpunkten gehörenden Gewichten w_i:

$$R_{i,n}(s) = \frac{N_{i,n}(s)w_i}{\sum_{j=0}^{m} N_{j,n}(s)w_j}. \tag{3.9}$$

Werden alle Gewichte w_i zu eins gesetzt, führt dies auf den entsprechenden B-Spline. Durch Gewichte größer als eins wird die Kurve zu dem Kontrollpunkt hingezogen, $w_i < 1$ stößt die Kurve dagegen ab.

Zusammenfassend kann gesagt werden, dass NURBS eine einheitliche mathematische Darstellung für sowohl analytische Standardformen als auch Freiformflächen bieten, die durch numerisch stabile und präzise Algorithmen schnell ausgewertet werden kann und verhältnismäßig wenig Speicherplatz benötigt.

Beispiel: Viertelkreis Am Beispiel eines Viertelkreises mit Radius eins wird die Beschreibung von Kurven durch NURBS gezeigt (Abbildung 3.7). Zur exakten Darstellung eines Kreises reichen NURBS 2-ten Grads aus. Benötigt werden drei Kontrollpunkte P_i mit den dazugehörenden Gewichten

$$\begin{array}{lllllllll} P_0 & = & (1;0) & & P_1 & = & (1;1) & & P_2 & = & (0;1) \\ w_0 & = & 1{,}0 & & w_1 & = & \sqrt{2}/2 & & w_2 & = & 1{,}0 \end{array} \tag{3.10}$$

und 6 Knoten:

$$\mathbf{S}_\mathrm{B} = \{0;0;0;1;1;1\}. \tag{3.11}$$

Nach Gleichung (3.7) ergeben sich die drei B-Spline-Basisfunktionen zu:

$$N_{0,2} = (1-s)^2; \qquad N_{1,2} = 2s(1-s); \qquad N_{2,2} = s^2. \tag{3.12}$$

Diese Funktionen, die in Abbildung 3.8 zu sehen sind, erfüllen aufgrund der dreifachen Wertigkeit an den Endknoten die Interpolationseigenschaft.

Einsetzen dieser Basisfunktionen und der Gewichte w_i in Gleichung (3.9) führt auf die rationalen Basisfunktionen. Mit diesen und den Kontrollpunkten kann nun die parametrische Darstellung des Viertelkreises bestimmt werden:

$$x(s) = \frac{(1-s)^2 + \sqrt{2}s(1-s)}{(1-s)^2 + \sqrt{2}s(1-s) + s^2}; \quad y(s) = \frac{\sqrt{2}s(1-s) + s^2}{(1-s)^2 + \sqrt{2}s(1-s) + s^2}. \tag{3.13}$$

Durch die Überprüfung von $x(s)^2 + y(s)^2 = 1$ lässt sich zeigen, dass durch diese parametrische Darstellung ein Viertelkreis exakt beschrieben wird.

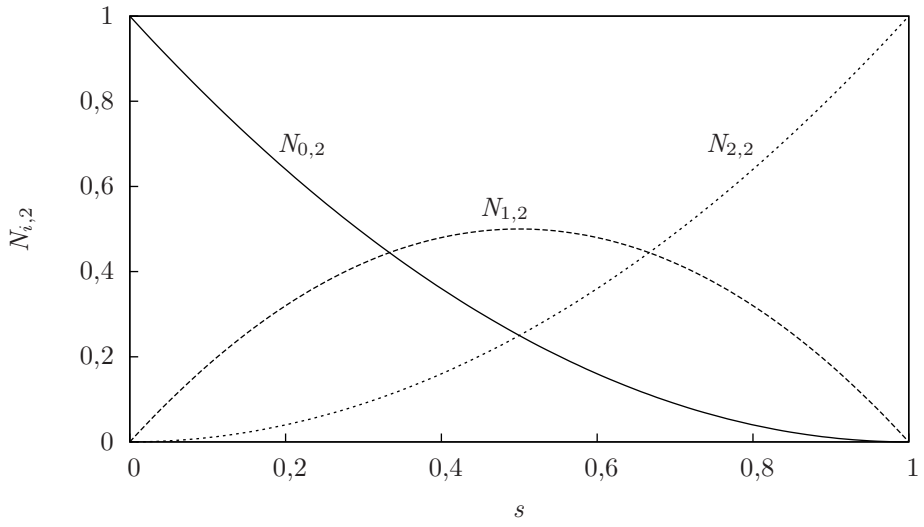


Abbildung 3.8: Quadratische B-Spline-Basisfunktionen zur Beschreibung eines Viertelkreises.

Positionierung der Knoten

Um neue Knoten in einem Master-Element auf einem gegebenen NURBS positionieren zu können, muss zunächst die Lage der Eckknoten des Master-Elements im Parameter s des NURBS bestimmt werden. Da die parametrische Beschreibung des NURBS im Allgemeinen nicht (eindeutig) invertierbar ist, wird dazu ein einfacher Suchalgorithmus, das Bisektionsverfahren, verwendet.

Ausgehend von einem Intervall, dass den gesamten NURBS beinhaltet, also die Grenzen $s_0 = 0$ und $s_1 = 1$ hat, werden die kartesischen Koordinaten der Intervallgrenzen bestimmt und geprüft, welcher dieser beiden Punkte weiter vom gesuchten Eckknoten des Master-Elements entfernt liegt. Diese Intervallgrenze wird daraufhin in die Mitte des ursprünglichen Intervalls verschoben und die Suche beginnt von Neuem. Abgebrochen wird die Suche, sobald der gesuchte Eckknoten näher als eine vorgegebene Toleranz an einer Intervallgrenze liegt. Die Genauigkeit, mit der die Lage der Eckknoten auf dem NURBS bestimmt wird, muss nicht besonders hoch sein, da durch eine ungenaue Positionierung nur die Gleichmäßigkeit der Verteilung der Slave-Knoten auf der Kante des Master-Elements beeinflusst wird, nicht aber die Genauigkeit, mit der die Geometrie approximiert wird.

Ist die Lage der Eckknoten des Master-Elements auf dem NURBS bestimmt, können die Slave-Knoten dazwischen, entlang der NURBS-Koordinate s gleichmäßig verteilt, positioniert werden. Aus den so errechneten Parameterwerten können aus der Definition des NURBS (3.8) die kartesischen Koordinaten der neuen Knoten bestimmt werden.

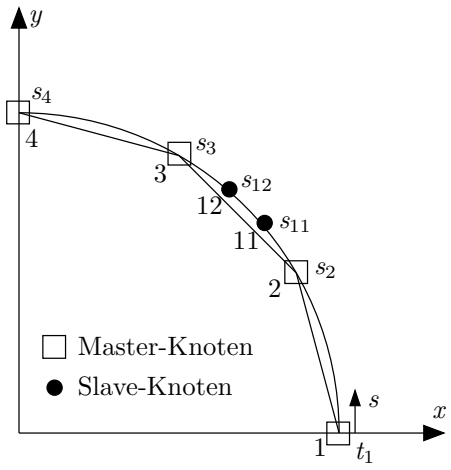


Abbildung 3.9: Element-Unterteilung eines linearen Linienelements auf einem Viertelkreis.

Abschließend wird die Positionierung von Knoten auf einem NURBS am Beispiel des Viertelkreises demonstriert.

Beispiel: Viertelkreis Ein Viertelkreis, der gemäß Gleichung (3.13) durch eine NURBS-Parametrisierung beschrieben ist, wurde durch drei lineare Master-Elemente vernetzt (Abbildung 3.9). Das mittlere dieser Elemente soll nun mit dem Faktor $subdiv = 3$ verfeinert werden.

Die Endknoten dieses Elements haben die Koordinaten:

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} 0{,}866 \\ 0{,}5 \end{pmatrix}; \qquad \begin{pmatrix} x_3 \\ y_3 \end{pmatrix} = \begin{pmatrix} 0{,}5 \\ 0{,}866 \end{pmatrix}. \tag{3.14}$$

Mit dem Bisektionsverfahren werden hieraus die entsprechenden Parameterwerte s bestimmt:

$$s_2 = 0{,}3411; \qquad s_3 = 0{,}6589. \tag{3.15}$$

Nun können die Positionen der beiden neuen Knoten gleichmäßig zwischen diesen Werten verteilt werden

$$s_{11} = s_2 + 1 \cdot \frac{s_3 - s_2}{3} = 0{,}447033; \qquad s_{12} = s_2 + 2 \cdot \frac{s_3 - s_2}{3} = 0{,}552967 \tag{3.16}$$

und daraus mit Gleichung (3.13) die kartesischen Koordinaten der Slave-Knoten ermittelt werden:

$$\begin{pmatrix} x_{11} \\ y_{11} \end{pmatrix} = \begin{pmatrix} 0{,}76632 \\ 0{,}64245 \end{pmatrix}; \qquad \begin{pmatrix} x_{12} \\ y_{12} \end{pmatrix} = \begin{pmatrix} 0{,}64245 \\ 0{,}76632 \end{pmatrix}. \tag{3.17}$$

3.3 Vektorisierung

Um auf einer bestimmten Hardware-Architektur eine hohe Effizienz zu erreichen, ist es im Allgemeinen von großem Vorteil, die speziellen Eigenschaften der gewählten Architektur, z.B. Vektorregister oder Cache-Hierarchien, bei der Programmierung zu berücksichtigen. Im Bereich des HPC ist es zwingend erforderlich und daher generell üblich prozessorangepasste Software zu verwenden (Turek 1998), d.h. besonders bei der Implementierung der rechenintensiven Programmteile werden Datenstrukturen und Algorithmen auf die Hardware abgestimmt.

Da es am Höchstleistungsrechenzentrum Stuttgart (HLRS) eine lange Tradition von Vektorrechnern gibt und auch der aktuelle Hochleistungsrechner ein Vektorrechner (SX-8 Homepage 2008) ist, wird hier gezeigt, wie ein wissenschaftliches Programm an die Vektor-Architektur angepasst, d.h. vektorisiert werden kann. Zunächst werden allgemeine Regeln genannt, die beim Programmieren beachtet werden müssen, um vektorisierenden Programmcode zu erzeugen. Abschließend wird dann am Beispiel der Berechnung der Elementmatrizen des stabilisierten Fluidelements gezeigt, wie diese Regeln in einem Finite-Element-Programm umgesetzt werden können und welchen Einfluss dies auf die Rechengeschwindigkeit hat.

3.3.1 Regeln für die Vektorisierung

Ausgehend von der Beschreibung eines Vektorprozessors und der Nutzung der Pipelines im Abschnitt 3.1.2 lassen sich zwei grundlegende Forderungen für vektorisierenden Programmcode aufstellen:

- hoher Anteil an Vektor-Operationen, um die leistungsfähigeren Vektor-Einheiten des Prozessors optimal zu nutzen
- große (optimale) Vektorlänge, um die Vektor-Register effizient zu nutzen

$\Rightarrow$ gleiche Operationen müssen simultan für viele unabhängige Datensätze ausgeführt werden.

Ein hoher Anteil an Vektor-Operationen kann erreicht werden, indem sichergestellt wird, dass alle Schleifen, bei verschachtelten Schleifen nur die innerste, vektorisiert abgearbeitet werden können. Dies ist in der Regel möglich, wenn Datenparallelität vorliegt und die Schleife weder Funktionsaufrufe, noch bedingte Anweisungen (z.B. „if“) oder In- und Output enthält. Da in der Programmiersprache C im Normalfall auf jede Variable mit beliebig vielen Zeigervariablen verwiesen werden kann, ist es für den Compiler in vielen Fällen nicht möglich zweifelsfrei festzustellen, ob in einer Schleife Datenparallelität vorliegt.

Die verwendete Vektorlänge hängt direkt mit der Anzahl der Iterationen einer Schleife zusammen. Um eine optimale Vektorlänge (256 auf der NEC SX-8-Plattform) zu erreichen, muss die Anzahl der Iterationen mindestens der Größe der Vektorregister entsprechen. Besser sind jedoch Schleifen, die ein Vielfaches dieser Zahl an Iterationen haben.

Außerdem ist es sinnvoll, wenn der Schleifenrumpf (die zu wiederholenden Anweisungen) nicht zu kurz ist. Der Mehraufwand („overhead“), der bei jeder Iteration durch das Hochzählen der Schleifenvariablen und das Prüfen der Abbruchbedingung entsteht, verhindert bei Schleifen mit nur sehr wenigen Anweisungen eine hohe Rechengeschwindigkeit. Zusätzlich ermöglicht eine große Anzahl an Anweisungen in einer Schleife auch die gleichzeitige Auslastung von allen vorhandenen Pipelines und damit eine höhere Leistung.

Auch der Zugriff auf Arrays in Schleifen hat einen großen Einfluss auf die Geschwindigkeit. Zum Beispiel erschwert eine indirekte Adressierung die Vektorisierung oder verhindert sie zum Teil vollständig. Entscheidend für die effiziente Nutzung der Vektorregister ist auch die Schrittweite, mit der Array-Elemente verwendet werden. Optimal ist ein „stride one“, eine Schrittweite von eins, d.h. die Schleifenvariable wird in der äußersten Dimension eines Arrays verwendet. Alternativ können auch ungerade Schrittweiten verwendet werden. Vorsicht ist geboten bei großen Schrittweiten, die auch noch eine Zweierpotenz darstellen. Diese tritt beispielsweise auf, wenn die Zählvariable der Schleife in der inneren Dimension eines Arrays verwendet wird und die äußere Dimension 512 ist. In diesem Fall können „memory bank conflicts“ das Laden der Daten in die Vektorregister verzögern.

Um die Vektorisierung eines existierenden Codes zu erhöhen, stehen drei verschiedene Ansätze zur Verfügung (Oliker u. a. 2003):

- Compiler Optionen
- Compiler Direktiven
- Code Modifikationen.

Diese drei Varianten unterscheiden sich im Aufwand, den der Entwickler in die Vektorisierung investieren muss, aber dafür auch umso deutlicher in ihrem Effekt.

Der einfachste Weg, die Vektorisierung eines Codes zu beeinflussen, ist der Einsatz von Compiler Optionen. Beim Aufruf des Compiler werden ihm Anweisungen übergeben, welche Vektor-Optimierungen er auf den gesamten Code anwenden soll. Dies kann zum Beispiel „expansion“, „unrolling“, „division“ oder „reordering“ von Schleifen sein oder das automatische „inlining“ von Funktionen. Compiler Optionen haben meist nur einen sehr geringen Einfluss auf die Vektorisierung des Codes und können bei einer zu starken Optimierung auch die Funktionsweise des Programms verändern.

Eine bessere Steuerungsmöglichkeit bieten hingegen Compiler Direktiven. Dies sind spezielle Anweisungen, die direkt in den Code geschrieben werden um den Compiler zu beeinflussen. Sie können zum Beispiel verwendet werden, um dem Compiler mitzuteilen, dass eine bestimmte Funktion immer automatisch „inlined“ werden soll, oder dass in einer Schleife alle Variablen datenparallel sind, obwohl Zeigervariablen verwendet werden. Durch den gezielteren Einsatz kann mit Compiler Direktiven in den meisten Fällen eine größere Leistungssteigerung erreicht werden als mit Compiler Optionen. Jedoch gibt es auch viele Fälle, bei denen der Grund, der die Vektorisierung verhindert, nur durch eine Restrukturierung des Codes beseitigt werden kann.

Bei einer Umstrukturierung des Programmcodes können die folgenden Maßnahmen ergriffen werden, um die Vektorisierbarkeit zu erhöhen (Chen u. a. 2003; Oliker u. a. 2003):

„loop fusion“ oder „division“
: Durch das Zusammenfügen oder Trennen von Schleifen können zu kurze Schleifen verlängert werden oder nicht vektorisierbare Abschnitte aus Schleifen entfernt werden.

„loop expansion“
: Schleifen mit nur wenigen Iterationen (2,3,4,...) können ausgeschrieben werden und so innerhalb von vektorisierbaren Schleifen stehen.

„loop unrolling“
: Beim „loop unrolling“ wird der Schleifenrumpf für mehrere (z.B. 5) Iterationen ausgeschrieben und im Gegenzug die Zählvariable bei jeder Iteration um 5 inkrementiert. Dadurch stehen mehr Anweisungen in einer Schleife (Ausnutzung aller vorhandenen Pipelines) und der Overhead für die Ausführung der Schleife wird verringert.

„loop interchange“
: Durch das Vertauschen der Reihenfolge von geschachtelten Schleifen kann entweder

die Schleife mit den meisten Iterationen als innerste Schleife verwendet werden oder in der innersten Schleife ein „stride one“ erzeugt werden.

Temporäre Variablen
In manchen Fällen ist es möglich durch den Einsatz von temporären Variablen Datenabhängigkeiten in Schleifen zu durchbrechen.

„Inlinen“ von Funktionen
Kurze Funktionen können am Ort des Funktionsaufrufs ausgeschrieben werden, um eine Vektorisierung zu ermöglichen. Das „Inlinen“ wird häufig, durch Compiler Direktiven gesteuert, vom Compiler automatisch ausgeführt.

Viele wissenschaftliche Programme, wie z.B. Finite-Element-Programme, erlauben eine Vektorisierung des Codes, da in ihnen eine natürliche Datenparallelität enthalten ist (Oliker u. a. 2003).

Im Folgenden wird ein Konzept vorgestellt, das im Rahmen dieser Arbeit entwickelt wurde (Neumann u. a. 2006; Förster u. a. 2006a) und mit dem bei geringem Implementierungsaufwand ein hoher Vektorisierungsgrad für die Berechnung der Elementmatrizen in einem Finite-Element-Programm erreicht werden kann.

3.3.2 Vektorisierungskonzept für die Berechnung von Elementmatrizen

Die Berechnung der Element(steifigkeits)-Matrizen ist neben dem Lösen des linearen Gleichungssystems der rechenaufwändigste Teil eines Finite-Element-Programms. Insbesondere komplexere Elemente, wie z.B. stabilisierte Fluid-Elemente oder „locking“-freie Schalenelemente, erfordern für jedes Element eine große Anzahl an Operationen. In manchen Fällen kann dies deutlich mehr als die Hälfte der gesamten Rechenzeit ausmachen.

Die Voraussetzungen, um bei der Berechnung der Elementmatrizen einen hohen Vektorisierungsgrad zu erreichen, sind offensichtlich: Für eine große Menge an unabhängigen Daten (für alle Elemente) müssen dieselben Operationen ausgeführt werden, um die Elementmatrizen zu berechnen. Bei der Implementierung werden die Schleifen aber häufig in einer Reihenfolge angeordnet, die eine Vektorisierung nicht zulässt.

Das Struktogramm in Abbildung 3.10 veranschaulicht diese Struktur, wobei für alle Schleifen die Zahl in Klammern eine typische Anzahl an Schleifendurchläufen angibt. Dabei werden alle Elemente einzeln hintereinander abgearbeitet. Die Schleife über alle Elemente ist somit die äußerste Schleife und kann nicht vektorisiert werden. Für jedes Element wird dann die numerische Integration mit einer Schleife über alle Integrations-

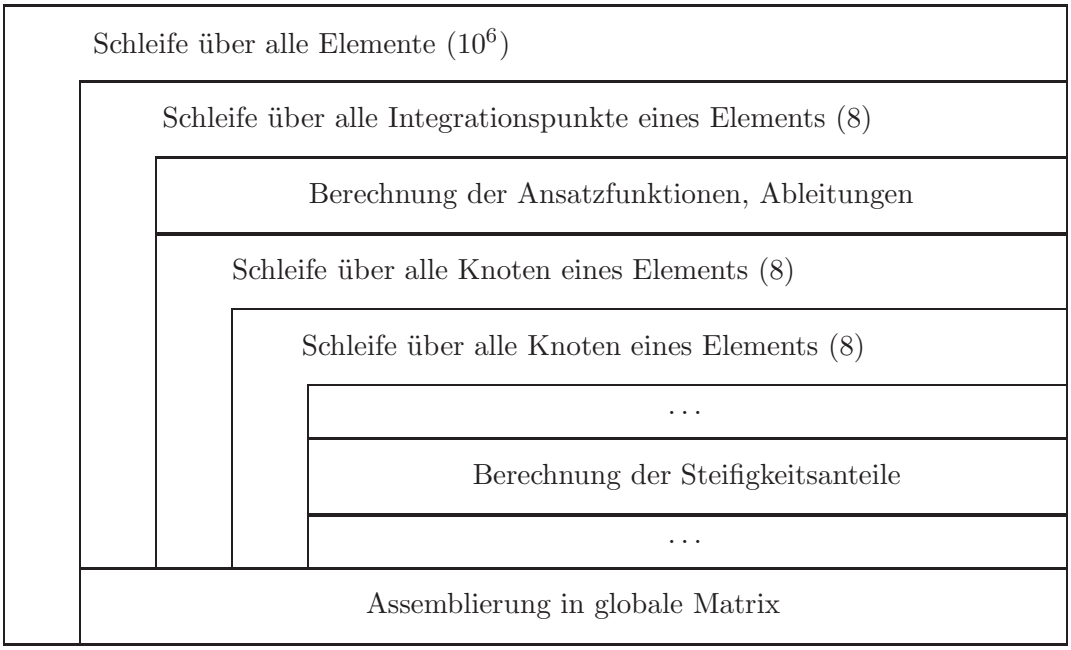


Abbildung 3.10: Struktogramm zur typischen Berechnung der Elementmatrizen.

punkte des Elements ausgeführt. Als innerste Schleifen ergeben sich häufig zwei Schleifen über alle Knoten des Elements, in denen die Anteile zur Elementmatrix für jeden Block berechnet werden. Diese Schleifen haben üblicherweise relativ wenige Iterationen und bieten daher kaum Potential für eine Vektorisierung.

Durch Vertauschen der Schleifen („loop interchange“) kann die Schleife über alle Elemente, die (abhängig von der Problemgröße) die meisten Iterationen hat, als innerste Schleife verwendet werden. Dabei muss sichergestellt werden, dass alle Elemente, die in dieser Schleife abgearbeitet werden, dieselben Eigenschaften haben, d.h. dass identische Operationen erforderlich sind, um die Elementmatrizen zu berechnen. Wichtig sind hierbei neben der Elementformulierung und dem Materialgesetz insbesondere die Anzahl der Knoten und Integrationspunkte. Außerdem ist es nun erforderlich Zwischenergebnisse, wie z.B. die Werte der Ansatzfunktionen am aktuellen Integrationspunkt, für alle Elemente der innersten Schleife abzuspeichern.

Um den erforderlichen Speicheraufwand begrenzt zu halten und auch unterschiedliche Elementtypen behandeln zu können, ist es sinnvoll nicht alle Elemente einer Simulation in die innerste Schleife zu überführen. Stattdessen können gleichartige Elemente in Gruppen zusammengefasst werden, deren Größe frei gewählt werden kann. Diese Gruppen werden in einer äußeren Schleife nacheinander abgearbeitet, wobei die Schleife über alle Elemente einer Gruppe immer als innerste Schleife auftritt (Abbildung 3.11).

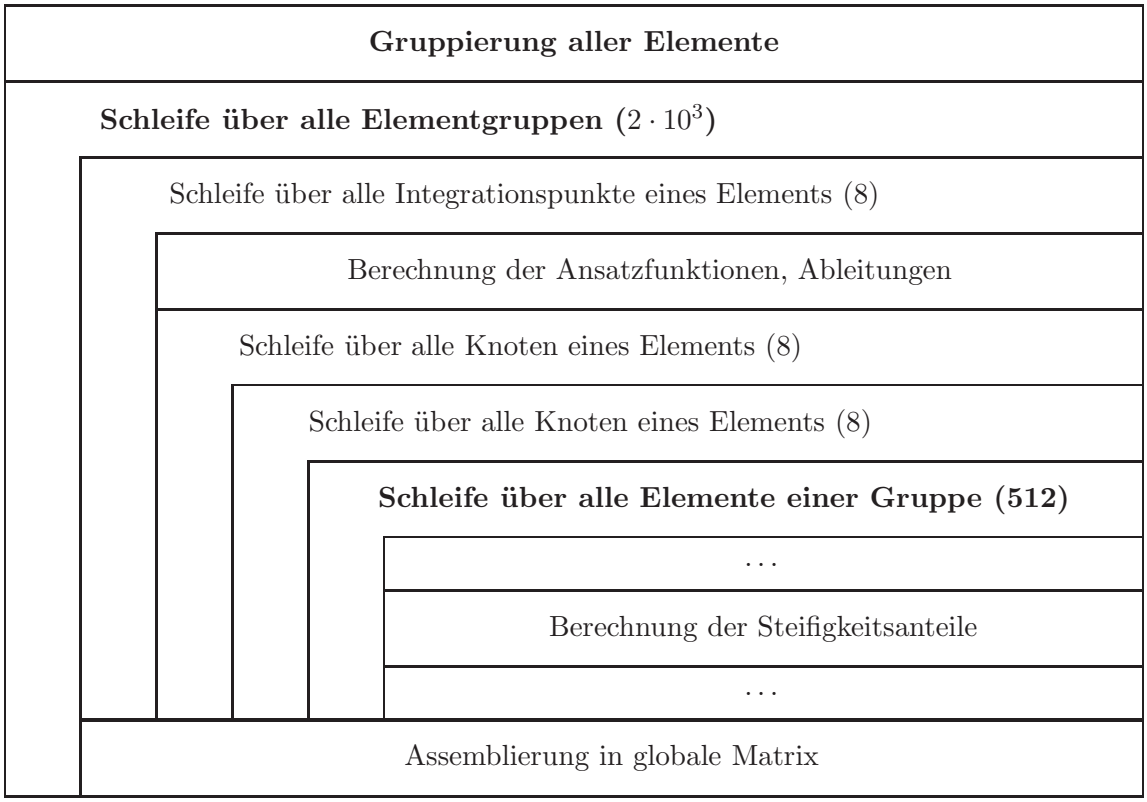


Abbildung 3.11: Struktogramm zur vektorisierten Berechnung der Elementmatrizen.

Die notwendigen Veränderungen am Code werden am Beispiel einer Unterfunktion, der Berechnung der Jacobi[4]-Matrix, verdeutlicht. Wird jedes Element, wie in der ursprünglichen Version (Abbildung 3.12), einzeln abgearbeitet, sind zur Berechnung der Jacobi-Matrix eines linearen, 8-knotigen Hexaeder-Elements die Ableitungen der acht Ansatzfunktionen in die drei Koordinatenrichtungen `deriv[3][8]` und die Koordinaten der acht Knoten `ele_coord[3][8]` erforderlich. Die berechnete Jacobi-Matrix, eine 3×3-Matrix, wird in `jaco[3][3]` gespeichert und so an die aufrufende Funktion zurückgegeben. Zur Berechnung sind nun drei Schleifen erforderlich: Mit den beiden Schleifen über `i` und `j` werden nacheinander die neun Einträge der Jacobi-Matrix abgearbeitet, während die Schleife über `l` die Anteile der acht Knoten zu jedem Eintrag aufaddiert. Mit acht Iterationen bietet hierbei die innerste Schleife kein Potential für eine effiziente Ausführung auf einem Vektorprozessor.

Wird die Jacobi-Matrix nun für eine Gruppe von z.B. 256 Elementen gleichzeitig berechnet, sind nur wenige Veränderungen am Code erforderlich (siehe Abbildung 3.13). Zum Einen werden die Knotenkoordinaten von allen 256 Elementen benötigt und es müssen 256 Jacobi-Matrizen gespeichert werden. Dazu erhalten die beiden Arrays `ele_coord`

[4]Carl Gustav Jacob Jacobi, * 10. Dezember 1804 in Potsdam, † 18. Februar 1851 in Berlin, deutscher Mathematiker.

```
f3_jaco ( deriv[3][8], ele_coord[3][8], jaco[3][3] )
{
  for (i=0; i<3; i++) {
    for (j=0; j<3; j++) {
      for (l=0; l<8; l++) {
       jaco[i][j] += deriv[i][l] * ele_coord[j][l];
  } } }
  return;
}
```

Abbildung 3.12: Pseudo-Code für eine Funktion zur Berechnung der Jacobi-Matrix: nicht vektorisierende Version.

```
f3f_jaco ( deriv[3][8], ele_coord[3][8][256], jaco[3][3][256] )
{
  for (i=0; i<3; i++) {
    for (j=0; j<3; j++) {
      for (l=0; l<8; l++) {
        for (k=0; k<256; k++) {
        jaco[i][j][k] += deriv[i][l] * ele_coord[j][l][k];
        }
  } } }
  return;
}
```

Abbildung 3.13: Pseudo-Code für eine Funktion zur Berechnung der Jacobi-Matrix: vektorisierende Version.

und `jaco` eine dritte Dimension der Länge 256. Da die Ableitungen der Ansatzfunktionen in natürlichen Koordinaten für alle Elemente gleich sind, muss `deriv` nicht verändert werden. Zum Anderen muss zusätzlich als innerste Iteration die Schleife über alle 256 Elemente einer Gruppe ergänzt werden (grauer Kasten). Diese Schleife kann vektorisiert abgearbeitet werden: Sie hat eine ausreichende Länge (in diesem Beispiel 256 Iterationen), ist datenparallel und verwendet einen „stride one“ für die relevanten Arrays. Um eine optimale Leistung zu erreichen, müsste der Schleifenrumpf noch mehr Anweisungen enthalten. Dies ist für viele andere Unterfunktionen bei der Berechnung der Elementmatrizen automatisch der Fall.

3.3.3 Einflüsse auf die Effizienz

Neben der richtigen Anordnung der Schleifen haben auch die Wahl der Programmiersprache, die Behandlung mehrdimensionaler Arrays und in diesem Fall die Größe der

Elementgruppen einen erheblichen Einfluss auf die Effizienz des Codes. Im Folgenden wird der Effekt der genannten Faktoren auf die Rechengeschwindigkeit untersucht.

Programmiersprache und mehrdimensionale Arrays

Obwohl allgemein bekannt ist, dass die Programmiersprache einen erheblichen Einfluss auf die Effizienz eines Programms haben kann, wird dieser Aspekt bei der Wahl der Programmiersprache häufig anderen Überlegungen, wie z.B. Programmierkomfort und verfügbare Bibliotheken, untergeordnet. Trotz großer Bemühungen und Fortschritten bei anderen Programmiersprachen (Veldhuizen 1997; Veldhuizen und Jernigan 1997), gilt Fortran auch heute noch als beste Wahl für einen effizienten wissenschaftlichen Code (Pohl u. a. 2004). Insbesondere das sehr allgemeine Zeiger-Konzept in C und die Objekte in C++ erschweren die Vektorisierung oder verhindern sie sogar vollständig (Oliker u. a. 2003). Da zwei verschiedene Zeiger auf dieselbe Variable zeigen können, ist es für den Compiler nur schwer möglich datenparallele Schleifen zu identifizieren. Abhilfe schaffen können in diesen Fällen Compiler Optionen oder das Schlüsselwort `restrict`. Durch letzteres kann dem Compiler mitgeteilt werden, dass es keine weiteren Zeiger auf eine Variable gibt. Da dies ein neueres Schlüsselwort im C-Standard ist, wird es noch nicht von allen Compilern unterstützt oder erzeugt einen ungewollten Effekt.

Im Folgenden werden sechs verschiedene Implementierungen der Berechnung der Elementmatrizen von 8-knotigen, stabilisierten Fluid-Elementen verglichen. Diese unterscheiden sich in der

- **Programmiersprache:** C oder Fortran
- **Verwendung von Arrays:** mehrdimensional oder nur eindimensional
- **Verwendung des Schlüsselworts** `restrict`.

Bei der Verwendung von ausschließlich eindimensionalen Arrays muss die Indizierung bei mehrdimensionalen Daten manuell erfolgen. Dies bedeutet einen deutlich höheren Programmieraufwand und ist außerdem auch fehleranfälliger. Bis auf die erste Implementierung verwenden alle Varianten die im vorangegangenen Abschnitt beschriebene vektorisierbare Berechnung der Elementmatrizen. Die genauen Eigenschaften der sechs untersuchten Implementierungen können dem oberen Teil der Tabelle 3.1 entnommen werden. Der untere Teil der Tabelle zeigt die an der ursprünglichen Implementierung (Abbildung 3.10) normierten Rechenzeiten für eine repräsentative Unterfunktion zur Berechnung von Matrixanteilen.

Zunächst werden nur die Ergebnisse für den Vektorprozessor, in diesem Fall den SX-6+-Prozessor, betrachtet. Alle fünf vektorisierenden Varianten („Vektor 1“ bis „Vektor

	Original	Vektor 1	Vektor 2	Vektor 3	Vektor 4	Vektor 5
Algorithmus	3.10	3.11	3.11	3.11	3.11	3.11
Sprache	C	C	C	C	C	Fortran
Arrays	mehrdim.	mehrdim.	mehrdim.	eindim.	eindim.	mehrdim.
Schlüsselwort			restrict		restrict	
SX-6+[3]	1,000	0,024	0,024	0,016	0,013	0,011
Itanium2[4]	1,000	1,495	1,236	0,742	0,207	0,105
Pentium4[5]	1,000	2,289	1,606	1,272	1,563	0,523

Tabelle 3.1: Eigenschaften der sechs Implementierungen und deren relative Zeit für die Berechnung repräsentativer Matrixanteile auf drei verschiedenen Hardware-Architekturen.

5“) benötigen hier nur weniger als 3 % der Rechenzeit der ursprünglichen Implementierung. Dies zeigt sehr eindrücklich, wie groß der Effizienzgewinn sein kann, wenn die grundlegenden Regeln der Vektorisierung berücksichtigt werden. Weiterhin wird deutlich, dass die Verwendung von eindimensionalen Arrays teilweise fast doppelt so schnell ist als der Einsatz von mehrdimensionalen Feldern. Das Schlüsselwort `restrict` zeigt nur bei eindimensionalen Arrays eine Auswirkung. Die schnellste Implementierung auf dem Vektorprozessor ist die Variante „Vektor 5“ in Fortran.

Werden die beiden anderen Hardware-Architekturen betrachtet, fällt zunächst auf, dass die reine Umstrukturierung des Codes („Vektor 1“ im Vergleich zu „Original“) einen negativen Effekt auf die Effizienz hat. Dies kann daran liegen, dass die Datenmenge, die in den neu geschaffenen, sehr langen inneren Schleifen benötigt wird, die Größe des Caches übersteigt und daher viele Zugriffe auf den Hauptspeicher erforderlich sind. Wie schon der Vektorprozessor, zeigen auch die beiden anderen Prozessoren bei der Verwendung von eindimensionalen Arrays eine fast doppelt so hohe Rechengeschwindigkeit. Jedoch hat das Schlüsselwort `restrict` nicht immer einen positiven Effekt. Auf dem Pentium-Prozessor reduziert es beim Einsatz für eindimensionale Arrays die Geschwindigkeit merklich.

Das wichtigste Ergebnis dieses Vergleichs ist die auf allen Plattformen überragende Leistung der Fortran-Implementierung. Auf den beiden Nicht-Vektor-Prozessoren ist diese Variante fast doppelt so schnell wie die jeweils zweitschnellste Implementierung.

[3] NEC SX-6+, 565 MHz; NEC C++/SX Compiler, Version 1.0 Rev. 063; NEC FORTRAN/SX Compiler, Version 2.0 Rev. 305.
[4] Hewlett Packard Itanium2, 1.3 GHz; HP aC++/ANSI C Compiler, Rev. C.05.50; HP F90 Compiler, v2.7.
[5] Intel Pentium4, 2.6 GHz; Intel C++ Compiler, Version 8.0; Intel Fortran Compiler, Version 8.0.

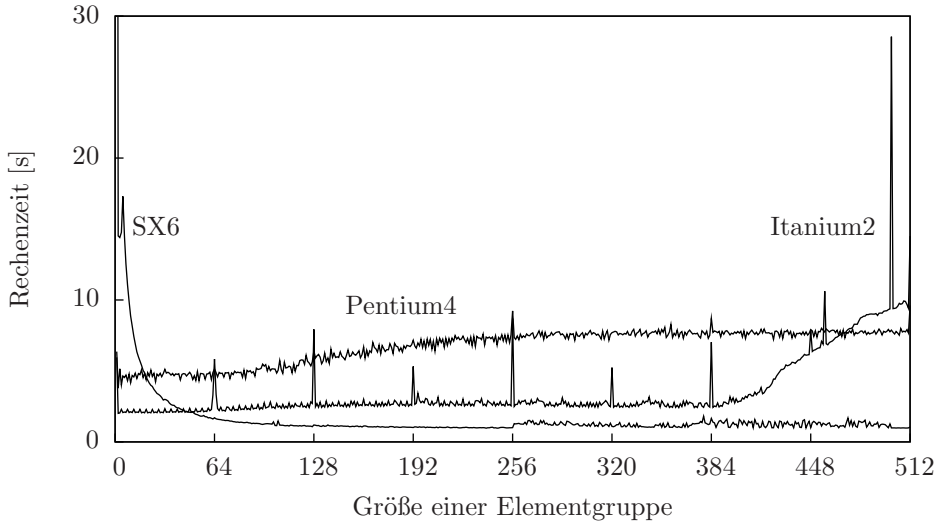


Abbildung 3.14: Zeit für die Berechnung repräsentativer Matrixanteile für unterschiedliche Größen der Elementgruppen.

Fortran wird also zu Recht als die beste Wahl für einen effizienten wissenschaftlichen Code bezeichnet (Pohl u. a. 2004).

Da die Implementierung in Fortran („Vektor 5“) sich auf allen getesteten Architekturen als die schnellste Variante herausgestellt hat, wurde in dieser Arbeit die komplette Berechnung der Elementmatrizen für das 3-dimensionale, stabilisierte Fluid-Element in dieser Form implementiert. Dabei wurde bewusst auf eine Verwendung von eindimensionalen Arrays in Fortran abgesehen, um den Programmcode übersichtlich und leichter wartbar zu halten. Auf die leichte Steigerung der Effizienz, die dadurch noch möglich wäre, wird zugunsten des Programmierkomforts verzichtet. Für die folgenden Untersuchungen und Beispiele wurde ausschließlich diese Implementierung verwendet.

Größe der Elementgruppen

Neben der Programmiersprache hat auch die Größe der Elementgruppen einen deutlichen Einfluss auf die Effizienz. Die optimale Größe der Gruppen und damit die Anzahl der Iterationen der innersten Schleife ist abhängig von der gewählten Hardware. Um die jeweils richtige Größe für die drei verwendeten Architekturen zu ermitteln, wurden die Rechenzeiten für eine repräsentative Unterfunktion zur Berechnung von Matrixanteilen für alle drei Prozessoren für Gruppen von einem Element bis 512 Elemente gemessen. Die Ergebnisse sind in Abbildung 3.14 dargestellt.

Die Kurve für den Pentium4-Prozessor zeigt einen typischen Verlauf für einen Cache-basierten Prozessor. Für kleine Elementgruppen, d.h. solange die Daten noch in den Cache

passen, zeigt er eine relativ konstante Leistung. Werden die Elementgruppen dann größer, muss häufiger auf den Hauptspeicher zugegriffen werden und die Rechenzeit steigt an. Auch der Itanium2-Prozessor zeigt diesen Anstieg der Rechenzeit, der auf ein Überlaufen des Caches deutet; hier jedoch deutlich später als beim Pentium4. Zusätzlich sind beim Itanium2 deutliche Rechenzeitspitzen für Vielfache von 64 zu erkennen. Diese könnten durch „memory bank conflicts“ verursacht werden. Durch die Wahl einer anderen Gruppengröße oder das künstliche Vergrößern der verwendeten Arrays um eins („array padding“) kann verhindert werden, dass zu schnell auf Speicherbereiche zugegriffen wird, die sich nach dem letzten Zugriff noch nicht regeneriert haben. Für beide Cache-basierte Prozessoren wird die größte Rechengeschwindigkeit mit 10 bis 20 Elementen je Gruppe erreicht.

Der Vektorprozessor SX-6+ zeigt einen deutlich anderen Verlauf. Die Rechenzeit sinkt für größer werdende Elementgruppen bis 256 Elemente je Gruppe erreicht sind und damit die Vektorregister optimal gefüllt werden. Für größere Gruppen bleibt die Rechengeschwindigkeit nahezu konstant, zeigt aber für Vielfache von 256 die besten Werte. Für das folgende Beispiel wurde eine Größe der Elementgruppen von 512 gewählt.

Numerisches Beispiel

Abschließend wird der Effekt der Vektorisierung an einer vollständigen Finite-Element-Simulation demonstriert. Gewählt wurde hierfür die Beltrami[5]-Strömung in einem Einheitswürfel. Details zu den Rand- und Anfangsbedingungen sind in Ethier und Steinman (1994) zu finden. Der Einheitswürfel wurde mit 32.768 stabilisierten, 8-knotigen Hexaeder-Elementen diskretisiert und die Strömung für 32 Zeitschritte berechnet.

In Abbildung 3.15 ist die Gesamtzeit für die Simulation, aufgeteilt in ihre Anteile für die Berechnung der Elementmatrizen, den Löser des linearen Gleichungssystems und sonstige Teile der Berechnung (z.B. Initialisierung und In- und Output), für zwei unterschiedliche Implementierungen dargestellt. Bei der ursprünglichen, nicht vektorisierbaren Implementierung („Original“) werden ungefähr 2/3 der Rechenzeit für die Berechnung der Elementmatrizen benötigt. Durch die in diesem Kapitel beschriebene Vektorisierung der Berechnung der Elementmatrizen („Vektor 5“) kann dieser Anteil um den Faktor 24 reduziert werden.

Diese deutliche Verbesserung ist auch in der vom Programm genutzten Leistung des Prozessors erkennbar. Tabelle 3.2 zeigt den Anteil der von beiden Implementierungen genutzten theoretischen Maximalleistungen der drei Prozessoren. Die linken beiden Spalten

[5]Eugenio Beltrami, * 16. November 1835 in Cremona, † 18. Februar 1900 in Rom, italienischer Mathematiker.

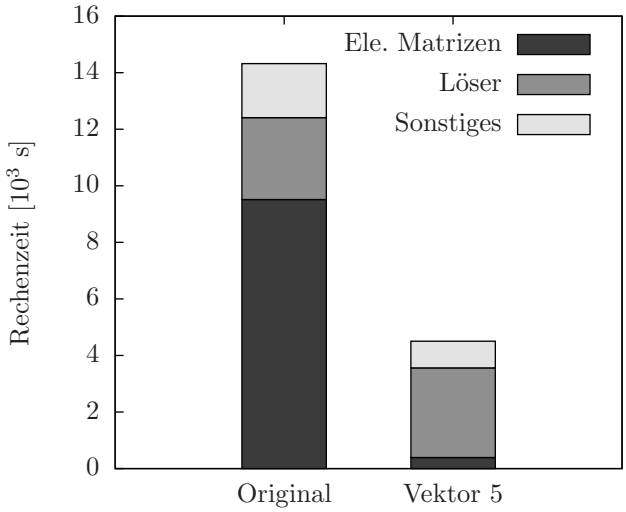


Abbildung 3.15: Anteile an der Rechenzeit für die Simulation von 32 Zeitschritten der Beltrami-Strömung auf dem Vektorprozessor SX-6+.

zeigen dabei die Werte für die bereits vorher in diesem Abschnitt verwendete repräsentative Unterfunktion zur Berechnung von Matrixanteilen, wohingegen die rechten Spalten die gesamte Berechnung der Elementmatrizen berücksichtigen. Die Werte hierfür sind grundsätzlich schlechter, da sie auch Teile des Codes berücksichtigen, die z.B. aufgrund von bedingten Anweisungen nicht vektorisiert werden können (z.B. die Bestimmung der Stabilisierungs-Parameter).

Die ursprüngliche Implementierung, die nicht für eine spezielle Architektur entwickelt worden war, zeigt nur eine sehr geringe Effizienz von unter einem Prozent auf dem Vektorprozessor. Auch auf den anderen beiden Prozessoren nutzt der Code höchstens 12,5 % der zur Verfügung stehenden Leistung. Die speziell für den Vektorprozessor entwickelte Implementierung der Berechnung der Elementmatrizen („Vektor 5“) erreicht in einer Unterfunktion, in der praktisch alle Anweisungen vektorisiert abgearbeitet werden können, eine hervorragende Ausnutzung von über 70 %. Aber auch für die gesamte Be-

	repr. Matrixanteile		ges. Elementberech.	
	Original	Vektor 5	Original	Vektor 5
SX-6+	0,83	71,07	0,95	29,55
Itanium2	6,59	59,71	8,68	35,01
Pentium4	10,31	23,98	12,52	20,16

Tabelle 3.2: Effizienz der ursprünglichen (Original) und der vektorisierten Implementierung (Vektor 5) in Prozent der theoretisch erreichbaren Maximalleistung.

rechnung der Elementmatrizen folgt noch eine akzeptable „sustained performance“ von ca. 30 %. Diese hohe Effizienz ergibt sich aus einer durchschnittlichen Vektorlänge von nahezu 256 und einem Anteil der Vektoroperationen von über 99,5 %.

Aber auch für den Itanium2- und Pentium4-Prozessor, die nicht die Zielarchitektur der Code-Optimierung waren, war es möglich die Effizienz der gesamten Berechnung der Elementmatrizen um einen Faktor von bis zu vier zu steigern.

4

Iterative Methoden zur Lösung linearer Gleichungssysteme

Die Lösung von sehr großen, schwach besetzten, linearen Gleichungssystemen ist einer der rechenaufwändigsten Teile einer Finite-Element-Simulation. Besonders bei sehr großen, instationären Problemen, wie sie in der CFD auftreten, kann die Effizienz des Lösers über die Lösbarkeit eines Problems entscheiden. Das Angebot an Lösungsverfahren ist umfangreich und die benötigte Rechenzeit hängt sehr stark vom gewählten Iterationsverfahren und Vorkonditionierer ab.

In den folgenden Abschnitten werden zunächst verschiedene Iterationsverfahren zur Lösung von linearen Gleichungssystemen und entsprechende Vorkonditionierer vorgestellt. Anhand eines exemplarischen CFD-Problems werden im Anschluss die Effizienz dieser Lösungsverfahren verglichen und der Einfluss verschiedener Parameter auf das Löserverhalten untersucht.

4.1 Einführung in iterative Löser

Das erste Iterationsverfahren für lineare Gleichungssysteme wurde von Carl Friedrich Gauß vor über 180 Jahren beschrieben. Um Gleichungssysteme zu lösen, die für eine Gauß-Elimination zu groß waren, entwickelte er 1822 eine Variante des heute bekannten Gauß-Seidel-Verfahrens. Auch Carl Gustav Jacobi veröffentlichte 1845 ein sehr ähnliches Verfahren. Erst über 100 Jahre später, als die klassischen Verfahren zum Lösen von immer größeren Gleichungssystemen auf elektronischen Rechnern zu langsam waren, wurden relaxierte Varianten dieser Verfahren entwickelt, die den Iterationsprozess wesentlich beschleunigten (Hackbusch 1993). Heute werden das Gauß-Seidel- und Jacobi-

Verfahren hauptsächlich als Vorkonditionierer in Kombination mit modernen, effektiveren Iterationsverfahren eingesetzt.

Wie schon bei Gauß, ist auch heute noch der Rechenaufwand ein Argument für den Einsatz von iterativen Verfahren im Vergleich zu direkten Lösern. Für vollbesetzte Matrizen der Größe $n \times n$ benötigen direkte Verfahren im Allgemeinen $O(n^3)$ Operationen. Iterative Methoden erfordern in jedem Schritt die Berechnung des Residuums des Systems und damit nur $O(n^2)$ Operationen. Ist die Anzahl der benötigten Iterationen also unabhängig von n oder skaliert sublinear mit n, sind iterative Methoden konkurrenzfähig zu direkten Verfahren (Plato 2000; Quarteroni u. a. 2002). Außerdem können direkte Verfahren die besondere Gestalt der aus Diskretisierungsverfahren stammenden Gleichungssysteme nicht immer optimal ausnutzen, wodurch dichter besetzte Zwischenmatrizen („fill-in“) entstehen können, die einerseits den verfügbaren Speicherplatz überschreiten und andererseits zu inakzeptablen Rechenzeiten führen (Quarteroni u. a. 2002).

Da die hier betrachteten Gleichungssysteme aus einer Diskretisierung der Problemstellung entstanden sind, entspricht die exakte Lösung des Gleichungssystems auch nur einer Approximation der gesuchten Lösung der ursprünglichen Problemstellung. Eine Näherungslösung des Gleichungssystems, wie sie iterative Verfahren bestimmen, mit einem Fehler in der Größenordnung des Diskretisierungsfehlers ist somit auch ausreichend. (Meister 1999)

4.2 Klassifizierung von Iterationsverfahren

Die Lösung $\mathbf{x}$ des linearen Gleichungssystems

$$\mathbf{A}\mathbf{x} = \mathbf{b} \qquad \mathbf{A} \in \mathbb{R}^{n \times n};\ \mathbf{x},\mathbf{b} \in \mathbb{R}^n \tag{4.1}$$

mit der gegebenen regulären Matrix $\mathbf{A}$ und der rechten Seite $\mathbf{b}$ soll mit einem iterativen Verfahren gefunden werden.

Ein allgemeines Iterationsverfahren, das aus einem gegebenen Startwert $\mathbf{x}^{(0)}$ die Iterierten $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots$ erzeugt, kann durch die Abbildung $\Phi^{(k)}$ definiert werden:

$$\begin{aligned} &\mathbf{x}^{(k+1)} := \Phi^{(k+1)}(\mathbf{x}^{(k)}, \mathbf{x}^{(k-1)}, \mathbf{x}^{(k-2)}, \ldots, \mathbf{x}^{(k-m)}, \mathbf{A}, \mathbf{b}), \quad k \geq m \\ &\Phi^{(k+1)} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n. \end{aligned} \tag{4.2}$$

Der Fixpunkt $\mathbf{x}^*$ des Iterationsverfahrens $\Phi^{(k)}$ ist definiert als

$$\mathbf{x}^* = \Phi^{(k)}(\mathbf{x}^*, \mathbf{A}, \mathbf{b}). \tag{4.3}$$

Ist für alle $\mathbf{b} \in \mathbb{R}^n$ jede Lösung $\mathbf{x}$ des Gleichungssystems (4.1) Fixpunkt des Iterationsverfahrens $\Phi^{(k)}$, so heißt dieses konsistent. Alle in diesem Kapitel vorgestellten Iterationsverfahren erfüllen diese Definition der Konsistenz.

Iterationsverfahren können u.a. nach den folgenden drei Kriterien kategorisiert werden (Quarteroni u. a. 2002):

- Ein Iterationsverfahren heißt **stationär**, wenn die Abbildung $\Phi^{(k)}$ unabhängig vom Iterationsschritt k ist, andernfalls wird es als **instationär** bezeichnet.
- Hängt $\Phi^{(k)}$ höchstens linear von $\mathbf{x}^{(0)}, \ldots, \mathbf{x}^{(k)}$ ab, so wird die Methode **linear**, andernfalls **nichtlinear** genannt.
- Die Anzahl von Schritten, von denen die aktuelle Iteration abhängt, heißt **Ordnung** der Methode.

4.3 Lineare Iterationsverfahren

Im Folgenden werden zunächst stationäre, lineare Iterationsverfahren erster Ordnung näher beschrieben, zu denen auch die beiden in der Einleitung erwähnten klassischen Methoden gehören. Für diese Verfahren lässt sich die Iteration Φ auch wie folgt darstellen:

$$\Phi(\mathbf{x}^{(k)}, \mathbf{A}, \mathbf{b}) = \mathbf{M}\mathbf{x}^{(k)} + \mathbf{C}\mathbf{b}. \tag{4.4}$$

Die Matrix $\mathbf{M}$ wird dabei als Iterationsmatrix bezeichnet. Die dazugehörige Iterationsvorschrift ergibt sich aus (4.2) und wird auch als erste Normalform des Verfahrens bezeichnet:

$$\mathbf{x}^{(k+1)} := \mathbf{M}\mathbf{x}^{(k)} + \mathbf{C}\mathbf{b}. \tag{4.5}$$

Für konsistente Verfahren lässt sich durch die Existenz des Fixpunkts ein Zusammenhang zwischen den Matrizen $\mathbf{M}$ und $\mathbf{C}$ aufstellen (Hackbusch 1993):

$$\mathbf{M} + \mathbf{C}\mathbf{A} = \mathbf{1}. \tag{4.6}$$

Diese Beziehung führt auf die zweite Normalform einer linearen und konsistenten Iteration:

$$\mathbf{x}^{(k+1)} := \mathbf{x}^{(k)} - \mathbf{C}(\mathbf{A}\mathbf{x}^{(k)} - \mathbf{b}). \tag{4.7}$$

Aus dieser Schreibweise wird deutlich, dass die Anwendung der Matrix $\mathbf{C}$ auf den aktuellen Defekt, das Residuum der Gleichung (4.1), zur Korrektur in jedem Iterationsschritt führt. Diese Korrektormatrix $\mathbf{C}$ kann als eine leicht zu invertierende Approximation der Matrix $\mathbf{A}$ verstanden werden und stellt somit auch eine mögliche Vorkonditionierungsmatrix für andere Iterationsverfahren dar.

Für ein lineares, konsistentes Iterationsverfahren mit der Iterationsmatrix $\mathbf{M}$ konvergiert die Folge $\{\mathbf{x}^{(k)}\}$ für jede beliebige Wahl von $\mathbf{x}^{(0)}$ zur Lösung des Gleichungsystems (4.1) genau dann, wenn für den Spektralradius der Iterationsmatrix $\rho(\mathbf{M}) < 1$ gilt (Quarteroni u. a. 2002). Eine hinreichende Bedingung für die Konvergenz einer Iteration ist die Abschätzung der Iterationsmatrix $\mathbf{M}$ in einer Matrixnorm $||\cdot||$.

$$||\mathbf{M}|| < 1 \tag{4.8}$$

Für konsistente Verfahren lässt sich über die sogenannte Kontraktionszahl $\zeta := ||\mathbf{M}||$ der Fehler in jeder Iterierten $\mathbf{e}^{(k)} = \mathbf{x}^{(k)} - \mathbf{x}^*$ abschätzen:

$$||\mathbf{e}^{(k+1)}|| \leq ||\mathbf{M}||\ ||\mathbf{e}^{(k)}||; \qquad ||\mathbf{e}^{(k)}|| \leq ||\mathbf{M}||^k\ ||\mathbf{e}^{(0)}||. \tag{4.9}$$

Durch diese Ungleichung $||\mathbf{e}^{(k+1)}|| \leq \zeta||\mathbf{e}^{(k)}||$ mit $\zeta :\leq 1$ wird lineare Konvergenz beschrieben. Schnellere Konvergenz ist nur mit nichtlinearen Iterationsverfahren zu erreichen (Hackbusch 1993).

Die Konstruktion konsistenter, linearer iterativer Methoden basiert üblicherweise auf der additiven Zerlegung der Matrix $\mathbf{A}$ in der Form:

$$\mathbf{A} = \mathbf{D}_\mathrm{A} - \mathbf{E}_\mathrm{A} - \mathbf{F}_\mathrm{A} \tag{4.10}$$

$\mathbf{D}_\mathrm{A}$: Diagonalmatrix, $\mathbf{D}_\mathrm{A} := \mathrm{diag}\{\mathbf{A}\}$

$\mathbf{E}_\mathrm{A}$: strikte untere Dreiecksmatrix

$\mathbf{F}_\mathrm{A}$: strikte obere Dreiecksmatrix.

Daher werden diese Verfahren auch als Splitting-Verfahren bezeichnet.

4.3.1 Jacobi-Verfahren

Das Jacobi- oder Gesamtschritt-Verfahren setzt nichtverschwindende Diagonalelemente $a_{ii} \neq 0,\ i = 1, \ldots, n$ der Matrix $\mathbf{A}$ voraus und nutzt die Inverse der somit regulären Diagonalmatrix $\mathbf{D}_\mathrm{A}$ als Korrektormatrix $\mathbf{C}$:

$$\mathbf{C}_\mathrm{J} = \mathbf{D}_\mathrm{A}^{-1}; \qquad \mathbf{M}_\mathrm{J} = \mathbf{D}_\mathrm{A}^{-1}(\mathbf{D}_\mathrm{A} - \mathbf{A}) = \mathbf{1} - \mathbf{D}_\mathrm{A}^{-1}\mathbf{A}. \tag{4.11}$$

In Komponentenschreibweise wird deutlich, dass die neue Iterierte $\mathbf{x}^{(k+1)}$ ausschließlich aus der alten Iterierten $\mathbf{x}^{(k)}$ ermittelt wird:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(k)} \right) \qquad i = 1, \ldots, n. \tag{4.12}$$

Dies bedeutet, dass jede Gleichung i von $i = 0$ bis $i = n$ einzeln nach der zugehörigen Komponente des Lösungsvektors x_i aufgelöst wird. Dabei werden für alle übrigen Komponenten des Lösungsvektors x_j die Werte aus der vorangegangenen Iteration (k) verwendet. Das Verfahren ist daher unabhängig von der gewählten Nummerierung der Unbekannten und kann sehr gut parallelisiert bzw. vektorisiert werden.

4.3.2 Gauß-Seidel-Verfahren

Beim Gauß-Seidel- oder Einzelschrittverfahren wird ebenfalls eine reguläre Diagonalmatrix $\mathbf{D}_\mathrm{A}$ vorausgesetzt, als Korrektormatrix werden aber zusätzlich noch die Einträge der unteren Dreiecksmatrix $\mathbf{E}_\mathrm{A}$ verwendet:

$$\mathbf{C}_\mathrm{GS} = (\mathbf{D}_\mathrm{A} - \mathbf{E}_\mathrm{A})^{-1}; \qquad \mathbf{M}_\mathrm{GS} = (\mathbf{D}_\mathrm{A} - \mathbf{E}_\mathrm{A})^{-1} \mathbf{F}_\mathrm{A}. \tag{4.13}$$

Zur Berechnung der i-ten Komponente der $(k+1)$-ten Iterierten werden dadurch neben den Komponenten der k-ten Iterierten auch die bereits bekannten ersten $i-1$ Komponenten der neuen Iterierten $\mathbf{x}^{(k+1)}$ verwendet.

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) \qquad i = 1, \ldots, n. \tag{4.14}$$

Im Gegensatz zum Jacobi-Verfahren werden hier beim Auflösen der i-ten Gleichung für die Komponenten des Lösungsvektors x_j oberhalb der aktuellen Gleichung $(j < i)$ bereits die in der aktuellen Iteration $(k+1)$ ermittelten Werte verwendet. Das Verfahren ist damit abhängig von der gewählten Nummerierung der Unbekannten und kann ohne besondere Techniken (z.B. „coloring“) nicht parallelisiert werden. Verglichen mit dem Jacobi-Verfahren wird mit $(\mathbf{D}_\mathrm{A} - \mathbf{E}_\mathrm{A})$ eine bessere Approximation von $\mathbf{A}$ als Korrektormatrix verwendet, wodurch ein kleinerer Spektralradius der Iterationsmatrix und damit eine schnellere Konvergenz vorausgesetzt werden kann.

4.3.3 Symmetrisches Gauß-Seidel-Verfahren

Setzt ein iteratives Verfahren die Symmetrie und positive Definitheit der Matrix voraus, so sollte dies auch für ein vorkonditioniertes System der Fall sein. Daher wird, besonders in seiner Anwendung als Vorkonditionierer, eine symmetrische Variante (SGS) des eigentlich unsymmetrischen Gauß-Seidel-Verfahrens verwendet. Zunächst wird dazu das rückwärts durchgeführte Gauß-Seidel-Verfahren (oder Gauß-Seidel-Rückwärtsverfahren) eingeführt, das sich durch Vertauschen der beiden Dreiecksmatrizen $\mathbf{E}_\mathrm{A}$ und $\mathbf{F}_\mathrm{A}$ ergibt:

$$\mathbf{C}_\mathrm{RGS} = (\mathbf{D}_\mathrm{A} - \mathbf{F}_\mathrm{A})^{-1}; \qquad \mathbf{M}_\mathrm{RGS} = (\mathbf{D}_\mathrm{A} - \mathbf{F}_\mathrm{A})^{-1}\mathbf{E}_\mathrm{A}. \tag{4.15}$$

In Komponentenschreibweise entspricht dies der Berechnung der Komponenten von $\mathbf{x}^{(k+1)}$ von unten nach oben, d.h. beginnend mit $x_n^{(k+1)}$.

Aus der Kombination der beiden Varianten des Gauß-Seidel-Verfahrens in einem Iterationsschritt folgt ein symmetrisches Verfahren mit der folgenden Korrektor- und Iterationsmatrix:

$$\begin{aligned} \mathbf{C}_\mathrm{SGS} &= (\mathbf{D}_\mathrm{A} - \mathbf{F}_\mathrm{A})^{-1}\mathbf{D}_\mathrm{A}(\mathbf{D}_\mathrm{A} - \mathbf{E}_\mathrm{A})^{-1} \\ \mathbf{M}_\mathrm{SGS} &= (\mathbf{D}_\mathrm{A} - \mathbf{F}_\mathrm{A})^{-1}\mathbf{E}_\mathrm{A}(\mathbf{D}_\mathrm{A} - \mathbf{E}_\mathrm{A})^{-1}\mathbf{F}_\mathrm{A}. \end{aligned} \tag{4.16}$$

4.3.4 Gauß-Seidel-Relaxationsverfahren

Für die drei bisher beschriebenen Verfahren lässt sich auch jeweils eine relaxierte Variante aufschreiben. Dabei wird die neue Iterierte $\mathbf{x}^{(k+1)}$ aus einer Linearkombination der alten Lösung $\mathbf{x}^{(k)}$ und dem Ergebnis der Iterationsvorschrift $\tilde{\mathbf{x}}^{(k+1)}$ gebildet:

$$\tilde{\mathbf{x}}^{(k+1)} = \mathbf{M}\mathbf{x}^{(k)} + \mathbf{C}\mathbf{b}; \qquad \mathbf{x}^{(k+1)} = \omega\tilde{\mathbf{x}}^{(k+1)} + (1-\omega)\mathbf{x}^{(k)}. \tag{4.17}$$

Ziel der Relaxation mit dem Parameter ω ist eine Verbesserung der Konvergenzgeschwindigkeit durch eine Minimierung des Spektralradius der Iterationsmatrix $\mathbf{M}$.

An dieser Stelle wird nur das Gauß-Seidel-Relaxationsverfahren gezeigt, die relaxierten Varianten des Jacobi- und symmetrischen Gauß-Seidel-Verfahrens lassen sich analog herleiten. In Komponentenschreibweise führt das Gauß-Seidel-Relaxationsverfahren, das in der englischsprachigen Literatur auch unter dem Namen SOR („successive overrela-

xation“) bekannt ist, auf:

$$x_i^{(k+1)} = \underbrace{\frac{\omega}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right)}_{\omega \hat{x}_i^{(k+1)}} + (1-\omega) x_i^{(k)}. \tag{4.18}$$

Die Korrektor- und Iterationsmatrix in relaxierter Form lauten somit:

$$\mathbf{C}_{\mathrm{SOR}} = \omega(\mathbf{D}_{\mathrm{A}} - \omega \mathbf{E}_{\mathrm{A}})^{-1}; \quad \mathbf{M}_{\mathrm{SOR}} = (\mathbf{D}_{\mathrm{A}} - \omega \mathbf{E}_{\mathrm{A}})^{-1} \left\{ (1-\omega)\mathbf{D}_{\mathrm{A}} + \omega \mathbf{F}_{\mathrm{A}} \right\}. \tag{4.19}$$

Die Bestimmung des optimalen ω ist hierbei nicht immer möglich, da ω nichtlinear in $\mathbf{M}_{\mathrm{SOR}}$ eingeht. Nur für bestimmte Klassen von Matrizen kann das optimale ω, welches $\rho(\mathbf{M}_{\mathrm{SOR}})$ minimiert, explizit bestimmt werden. Für allgemeine Matrizen kann nur gesagt werden, dass das SOR-Verfahren höchstens für einen Relaxationsparameter $\omega \in [0; 2]$ konvergiert.

Sind die Matrix $\mathbf{A}$ konsistent geordnet, die Eigenwerte der Matrix $\mathbf{M}_{\mathrm{J}}$ reell und der Spektralradius der Iterationsmatrix der Jacobi-Iteration kleiner eins ($\rho := \rho(\mathbf{M}_{\mathrm{J}}) < 1$), konvergiert das Gauß-Seidel-Relaxationsverfahren für alle $\omega \in [0; 2]$ und der Spektralradius der Iterationsmatrix $\mathbf{M}_{\mathrm{SOR}}$ wird minimal für (Meister 1999)

$$\omega_{\mathrm{opt}} = \frac{2}{1 + \sqrt{1 - \rho^2}}. \tag{4.20}$$

Bei der Anwendung des Gauß-Seidel-Relaxationsverfahrens auf Diskretisierungen von Randwertaufgaben ist im Allgemeinen $\omega > 1$ zu wählen, das heißt zu überrelaxieren (Knabner und Angermann 2000). Dies erklärt auch die englische Bezeichnung des Verfahrens.

4.4 Projektionsmethoden und Krylow-Unterraum-Verfahren

Projektionsmethoden berechnen eine Folge von Näherungslösungen $\mathbf{x}^{(k)}$ für ein lineares Gleichungssytem (4.1) unter Verwendung von K_{k}, eines k-dimensionalen Unterraums des $\mathbb{R}^n$,

$$\mathbf{x}^{(k)} \in \left\{ \mathbf{x}^{(0)} + \mathbf{y}, \;\; \mathbf{y} \in K_{\mathrm{k}} \right\} \tag{4.21}$$

unter Berücksichtigung der Orthogonalitätsbedingung:

$$\left(\mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}\right) \perp L_\mathrm{k}. \tag{4.22}$$

Hierbei bezeichnet L_k einen weiteren k-dimensionalen Unterraum des $\mathbb{R}^n$. Gilt $K_\mathrm{k} = L_\mathrm{k}$ so steht der Residuenvektor $\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}$ senkrecht auf K_k und es wird von einer orthogonalen Projektionsmethode gesprochen. Die Bedingung (4.22) wird in diesem Fall als Galerkin-Bedingung bezeichnet. Für $K_\mathrm{k} \neq L_\mathrm{k}$ liegt eine schiefe Projektionsmethode vor und (4.22) heißt dann Petrow-Galerkin-Bedingung.

Ein Krylow[1]-Unterraum-Verfahren ist eine Projektionsmethode zur Lösung eines linearen Gleichungssystems, bei der K_k den Krylow-Unterraum

$$K_\mathrm{k} = K_\mathrm{k}(\mathbf{A}, \mathbf{r}^{(0)}) = \mathrm{span}\left\{\mathbf{r}^{(0)}, \mathbf{A}\mathbf{r}^{(0)}, \ldots, \mathbf{A}^{k-1}\mathbf{r}^{(0)}\right\} \tag{4.23}$$

darstellt. Zwei der bekanntesten Verfahren sind das Verfahren der konjugierten Gradienten („conjugate gradients“, CG), entwickelt von Hestenes und Stiefel (1952), und die von Saad und Schulz (1986) hergeleitete „generalized minimal residual method“ (GMRES). Die große Nachfrage nach effizienten, schnellen und robusten iterativen Gleichungslösern hatte einen maßgeblichen Einfluss auf die Forschung und die dadurch erzielten Fortschritte im Bereich der Krylow-Unterraum-Verfahren. Neben den im Folgenden vorgestellten drei Verfahren existieren noch unzählige weitere Methoden oder Varianten, die hauptsächlich in den letzen 10 Jahren des 20. Jahrhunderts entwickelt wurden. Einen historischen Überblick über die Entwicklungen im Bereich der iterativen Lösungsverfahren für lineare Gleichungssysteme im 20. Jahrhundert bietet der Aufsatz von Saad und van der Vorst (2000). Die wichtigsten dieser Verfahren sind in dem Buch von Meister (1999) beschrieben, in dem auch die Zusammenhänge zwischen den einzelnen Verfahren veranschaulicht werden.

4.4.1 CG-Verfahren

Das CG-Verfahren setzt eine symmetrische, positiv definite Matrix $\mathbf{A}$ voraus und ist daher nicht geeignet zur Lösung von Gleichungssystemen, die aus der in dieser Arbeit beschriebenen Diskretisierung der instationären Navier-Stokes-Gleichungen mit stabilisierten Finiten Elementen resultieren. Es wird an dieser Stelle aber als ursprüngliches Krylow-Unterraum-Verfahren und damit als Grundlage für alle weiteren Verfahren kurz beschrieben.

[1] Alexei Nikolajewitsch Krylow, * 3. August 1863 in Wisjaga, † 26. Oktober 1945 in Leningrad, russischer Schiffsbau-Ingenieur und Mathematiker.

Die Herleitung basiert auf der Umwandlung des linearen Gleichungssystems (4.1) in eine Minimierung der Funktion

$$\begin{aligned} F &: \mathbb{R}^n \to \mathbb{R} \\ &\mathbf{x} \to F(\mathbf{x}) := \frac{1}{2} \left(\mathbf{A}\mathbf{x}, \mathbf{x}\right)_2 - \left(\mathbf{b}, \mathbf{x}\right)_2 . \end{aligned} \tag{4.24}$$

Für symmetrische, positiv definite Matrizen entspricht die Lösung $\mathbf{x}^*$ des Minimierungsproblems

$$\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathbb{R}^n} F(\mathbf{x}) \tag{4.25}$$

der Lösung des ursprünglichen Gleichungssystems (4.1).

Die Funktion $F(\mathbf{x})$ wird nun in jedem Iterationsschritt, ausgehend von der letzten Iterierten $\mathbf{x}^{(k)}$, sukzessive entlang einer speziellen Richtung $\mathbf{p}^{(k)}$ minimiert:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda^{(k)} \mathbf{p}^{(k)}. \tag{4.26}$$

Zunächst wird die Suchrichtung $\mathbf{p}^{(k)}$ als bekannt vorausgesetzt und die zugehörige optimale Schrittweite $\lambda^{(k)}$ bestimmt. Die Extremwertbestimmung der Funktion $F(\mathbf{x})$ entlang der Suchrichtung $\mathbf{p}^{(k)}$ führt für symmetrische, positiv definite Matrizen $\mathbf{A}$ auf den optimalen Wert für $\lambda^{(k)}$:

$$\lambda_{\text{opt}}^{(k)} = \frac{\left(\mathbf{r}^{(k)}, \mathbf{p}^{(k)}\right)_2}{\left(\mathbf{A}\mathbf{p}^{(k)}, \mathbf{p}^{(k)}\right)_2}. \tag{4.27}$$

Wird als Suchrichtung $\mathbf{p}^{(k)}$ der aktuelle Gradient der Funktion $F(\mathbf{x})$ gewählt

$$\mathbf{p}^{(k)} = \nabla F(\mathbf{x}^{(k)}) = \mathbf{A}\mathbf{x}^{(k)} - \mathbf{b} = -\mathbf{r}^{(k)}, \tag{4.28}$$

ergibt sich das Verfahren des steilsten Abstiegs, das auch Gradienten-Verfahren genannt wird. Dies stellt in jedem Schritt eine orthogonale Projektionsmethode mit

$$K = L = \text{span}\{\mathbf{r}^{(k-1)}\} \tag{4.29}$$

dar. Die jeweilige Näherungslösung $\mathbf{x}^{(k)}$ ist stets nur bezüglich $\mathbf{r}^{(k-1)}$ optimal. Dies resultiert in einer relativ schlechten Konvergenz. Das Gradienten-Verfahren wird daher hier als Lösungsverfahren nicht weiter betrachtet, aber im Kapitel zur Fluid-Struktur-Kopplung (Abschnitt 5.6.1) wieder aufgegriffen.

Das Verfahren der konjugierten Gradienten erweitert nun die Optimalität der Näherungslösung $\mathbf{x}^{(k)}$ auf den gesamten Unterraum $U_{\text{k}} = \text{span}\{\mathbf{p}^{(0)}, \ldots, \mathbf{p}^{(k-1)}\}$ mit linear un-

abhängigen Suchrichtungen $\mathbf{p}^{(0)}, \ldots, \mathbf{p}^{(k-1)}$. Dadurch wird spätestens nach n Iterationsschritten die Lösung des Gleichungssystems gefunden

$$\mathbf{x}^{(n)} = \mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}. \tag{4.30}$$

Das CG-Verfahren kann somit auch als direkter Löser interpretiert werden. Für große Matrizen wird eine hinreichend genaue Lösung in der Regel bereits mit deutlich weniger als n Iterationen gefunden.

Die Suchrichtungen des CG-Verfahrens werden sukzessive aus den Residuenvektoren $\mathbf{r}^{(0)}$, $\ldots$, $\mathbf{r}^{(k)}$ ermittelt:

$$\mathbf{p}^{(0)} = \mathbf{r}^{(0)}; \qquad \mathbf{p}^{(k)} = \mathbf{r}^{(k)} + \sum_{j=0}^{k-1} \alpha_j \mathbf{p}^{(j)}. \tag{4.31}$$

Durch die Berücksichtigung der bereits verwendeten Suchrichtungen $\mathbf{p}^{(0)}, \ldots, \mathbf{p}^{(k-1)}$ stehen nun k Freiwerte α_j zur Verfügung, mit denen die Konjugiertheit der Suchrichtungen gewährleistet werden kann:

$$0 = \left(\mathbf{A}\mathbf{p}^{(k)}, \mathbf{p}^{(i)}\right)_2 = \left(\mathbf{A}\mathbf{r}^{(k)}, \mathbf{p}^{(i)}\right)_2 + \sum_{j=0}^{k-1} \alpha_j \left(\mathbf{A}\mathbf{p}^{(j)}, \mathbf{p}^{(i)}\right)_2 \quad \text{für } i = 0, \ldots, k-1. \tag{4.32}$$

Durch Erfüllung dieser Bedingung folgt die benötigte Vorschrift zur Berechnung der Koeffizienten α_j:

$$\alpha_j = -\frac{\left(\mathbf{A}\mathbf{r}^{(k)}, \mathbf{p}^{(j)}\right)_2}{\left(\mathbf{A}\mathbf{p}^{(j)}, \mathbf{p}^{(j)}\right)_2}. \tag{4.33}$$

Das CG-Verfahren stellt eine orthogonale Krylow-Unterraum-Methode dar, bei der die Iterierten $\mathbf{x}^{(k)}$ sich aus dem Anfangswert $\mathbf{x}^{(0)}$ und dem Krylow-Unterraum K_k zusammensetzen:

$$\mathbf{x}^{(k)} \in \mathbf{x}^{(0)} + \underbrace{\operatorname{span}\left\{\mathbf{p}^{(0)}, \ldots, \mathbf{p}^{(k-1)}\right\}}_{= \operatorname{span}\{\mathbf{r}^{(0)}, \ldots, \mathbf{A}^{k-1}\mathbf{r}^{(0)}\} = K_\mathrm{k}}. \tag{4.34}$$

In dieser Formulierung hat das CG-Verfahren noch den Nachteil, dass zur Bestimmung einer neuen Suchrichtung alle bisher verwendeten Suchrichtungen benötigt werden. Im ungünstigsten Fall wird daher der Speicherplatz einer vollbesetzten $n \times n$-Matrix benötigt, was bei sehr großen, schwachbesetzten Matrizen ineffizient und unpraktikabel ist.

Es kann aber zusätzlich gezeigt werden, dass das Residuum $\mathbf{r}^{(k)}$ bereits zu allen $\mathbf{p}^{(j)}$ mit $0 \leq j < k-1$ konjugiert ist (siehe z.B. MEISTER (1999) oder QUARTERONI U. A. (2002)). Deshalb kann jede neue Suchrichtung allein aus dem aktuellen Residuum und der letzten bekannten Suchrichtung bestimmt werden:

$$\mathbf{p}^{(k)} = \mathbf{r}^{(k)} - \frac{\left(\mathbf{A}\mathbf{r}^{(k)}, \mathbf{p}^{(k-1)}\right)_2}{\left(\mathbf{A}\mathbf{p}^{(k-1)}, \mathbf{p}^{(k-1)}\right)_2}\mathbf{p}^{(k-1)}. \tag{4.35}$$

Wie bereits erwähnt, findet das CG-Verfahren nach höchstens n Schritten die exakte Lösung des linearen Gleichungssystems. Diese Aussage ist aber nur für eine exakte Arithmetik gültig. Kumultative Rundungsfehler können bei einer endlichen Arithmetik die Konjugiertheit der Suchrichtungen und damit das Auffinden der Lösung nach n Schritten verhindern. In einem solchen Fall kann der Iterationsprozess mit dem letzten bekannten Residuum neu gestartet werden und es folgt ein zyklisches CG-Verfahren oder CG-Verfahren mit Neustart (QUARTERONI U. A. 2002).

Der Fehler $\mathbf{e}^{(k)} = \mathbf{x}^{(k)} - \mathbf{x}^*$ für jede Iterierte kann für das Verfahren der konjugierten Gradienten aus der spektralen Konditionszahl κ der Matrix $\mathbf{A}$ abgeschätzt werden (KNABNER UND ANGERMANN 2000):

$$||\mathbf{e}^{(k)}|| \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k ||\mathbf{e}^{(0)}||. \tag{4.36}$$

Die spektrale Konditionszahl κ lässt sich mit der Spektralnorm $||\bullet||_2$ oder dem maximalen und minimalen Eigenwert ($\sigma_{\mathrm{n}}(\mathbf{A})$ und $\sigma_1(\mathbf{A})$) der Matrix $\mathbf{A}$ darstellen:

$$\kappa = ||\mathbf{A}||_2 \cdot ||\mathbf{A}^{-1}||_2 = \frac{\sigma_1(\mathbf{A})}{\sigma_{\mathrm{n}}(\mathbf{A})}. \tag{4.37}$$

Für symmetrische, positiv definite Matrizen gilt zusätzlich $\kappa \geq 1$. Es wird deutlich, dass die Konditionszahl der Matrix $\mathbf{A}$ eine entscheidende Bedeutung für die Konvergenzgüte hat. Das CG-Verfahren arbeitet dann effizient, wenn die Konditionszahl nicht allzu groß ist oder aber durch geeignete Maßnahmen, wie zum Beispiel Vorkonditionierung, ausreichend reduziert werden kann.

4.4.2 BiCGSTAB-Verfahren

Das BiCGSTAB-Verfahren („stabilized bi-conjugate gradient“, Stabilisiertes Gradientenverfahren biorthogonaler Richtungen) stellt den aktuellen state-of-the-art einer Reihe von Verfahren dar, die das CG-Verfahren auf unsymmetrische Matrizen erweitern. Das erste Verfahren in dieser Entwicklung, das BiCG-Verfahren („bi-conjugate gradi-

ent“), wurde ursprünglich bereits 1952 von Lanczos (1952) vorgestellt. Das Verfahren stellt eine auf dem Bi-Lanczos[2]-Algorithmus basierende Krylow-Unterraum-Methode dar, wobei die Orthogonalität durch eine Petrow-Galerkin-Bedingung $L_\mathrm{k} = K_\mathrm{k}^\mathrm{T} = \mathrm{span}\{\mathbf{r}^{(0)}, \mathbf{A}^\mathrm{T}\mathbf{r}^{(0)}, \ldots, (\mathbf{A}^\mathrm{T})^{k-1}\mathbf{r}^{(0)}\}$ gegeben ist. Neben den erforderlichen Multiplikationen mit der Transponierten der Systemmatrix weist die Methode noch zwei weitere erhebliche Nachteile auf. Einerseits kann das Verfahren bei einer regulären Matrix abbrechen, ohne dass die exakte Lösung gefunden wurde, andererseits sind die Iterierten, im Gegensatz z.B. zum GMRES-Verfahren, durch keine Minimierungseigenschaft charakterisiert, wodurch sich Oszillationen im Konvergenzverhalten zeigen können. Für symmetrische, positiv definite Matrizen stimmt das BiCG-Verfahren mit dem CG-Verfahren überein.

Das CGS-Verfahren („conjugate gradient squared“) von Sonneveld (1989) stellt eine Weiterentwicklung des BiCG-Verfahrens dar, bei dem die Notwendigkeit der Multiplikation mit $\mathbf{A}^\mathrm{T}$ entfällt. Außerdem weist es aufgrund der Quadrierung des Polynoms zur Definition des Residuenvektors ein schnelleres Konvergenzverhalten bei gleichbleibendem Rechenaufwand auf. Es können sich aber weiterhin, wie beim BiCG-Verfahren, Oszillationen im Residuenverlauf zeigen.

Ein wesentlich glatteres Konvergenzverhalten zeigt dagegen die BiCGSTAB-Methode (van der Vorst 1992), die unterschiedliche Polynome für die Definition der Suchrichtungen und der Residuenvektoren verwendet und über einen zusätzlichen Freiheitsgrad die Minimierung des Residuums ermöglicht.

Für eine BiCGSTAB-Iteration sind zwei Matrix-Vektor-Produkte, die für dünnbesetzte Matrizen einen Rechenaufwand der Ordnung $\mathrm{O}(nnz)$ bedeuten, und sechs Skalarprodukte mit einem Aufwand der Ordnung $\mathrm{O}(n)$ erforderlich. n bezeichnet hierbei die Dimension der Matrix und nnz die Anzahl der Einträge, die ungleich null sind. Zusätzlich sind noch zwei Vorkonditionierungsschritte erforderlich, deren Aufwand vom gewählten Vorkonditionierer abhängt.

4.4.3 GMRES-Verfahren

Wie bereits schon das BiCGSTAB-Verfahren ist auch die „generalized minimal residual method“ (GMRES) von Saad und Schulz (1986) für allgemeine lineare Gleichungssysteme geeignet. Die Regularität der Matrix reicht als theoretisch hinreichende Bedingung für die Konvergenz und Stabilität des Verfahrens aus.

[2]Cornelius Lanczos, * 2. Februar 1893 in Székesfehérvár, † 25. Juni 1974 in Budapest, ungarischer Mathematiker und Physiker.

Die GMRES-Methode ist eine Krylow-Unterraum-Methode, bei der die Projektion durch eine Petrow-Galerkin-Bedingung mit $L_\mathrm{k} = \mathbf{A}K_\mathrm{k}$ gegeben ist und das Verfahren somit eine schiefe Projektionsmethode darstellt. Dadurch ergibt sich die spezielle Eigenschaft, dass das Residuum in jedem Iterationsschritt $|\mathbf{r}^{(k)}| = |\mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}|$ minimal im Vergleich zu allen anderen möglichen $\mathbf{x} \in \mathbf{x}^{(0)} + K_\mathrm{k}$ ist. In der Theorie ist somit das Residuum nicht nur monoton fallend, sondern sogar optimal monoton fallend.

Analog zum CG-Algorithmus kann das GMRES-Verfahren formal auch als direkter Löser interpretiert werden. Nach n Iterationen spannt der Krylow-Unterraum K_n den gesamten Raum $\mathbb{R}^n$ auf und die exakte Lösung des Gleichungssystems ist gefunden. Aufgrund des benötigten Speicherplatzes ist die Verwendung des GMRES in dieser Form aber nicht praktikabel.

Auch in der iterativen Anwendung weist die Methode zwei grundlegende Nachteile auf: Einerseits wächst der Rechenaufwand zur Bestimmung der Orthogonalbasis des Unterraums K_k linear mit der Dimension des Krylow-Unterraums und andererseits steigt auch der Speicherbedarf mit jeder Iteration, da alle Basisvektoren des Unterraums gespeichert werden müssen. Bei großen Gleichungssystemen übersteigt der Speicherbedarf häufig die vorhandenen Ressourcen. Daher wird in der Praxis oftmals ein GMRES-Verfahren mit Neustart verwendet, bei dem die maximale Größe des Krylow-Unterraums auf einen Wert m_max beschränkt wird. Ist bei Erreichen dieser Grenze noch nicht die erforderliche Genauigkeit der Lösung erreicht, wird der aktuelle Lösungsvektor als Startwert eines Neustarts verwendet, wobei die bisherigen Krylow-Vektoren verworfen werden und ein neuer Unterraum aufgebaut wird. Die Interpretation als direktes Verfahren geht in dieser Variante zwar verloren, das Residuum ist aber dennoch monoton fallend.

Der Rechenaufwand für eine GMRES-Iteration ist geringer als für eine BiCGSTAB-Iteration, hängt aber auch von der gewählten Größe des Krylow-Unterraums ab. Neben nur einem Matrix-Vektor-Produkt der Ordnung $\mathrm{O}(nnz)$ werden zwei Skalarprodukte der Ordnung $\mathrm{O}(n)$ und drei Skalarprodukte der Ordnung $m \cdot \mathrm{O}(n)$ benötigt. Dabei bezeichnet m die aktuelle Größe des Krylow-Unterraums. Zusätzlich ist nur ein Vorkonditionierungsschritt erforderlich.

Für allgemeine Matrizen ist das monotone Fallen des Residuums in euklidischer Norm die stärkste mögliche Konvergenzaussage. Im ungünstigsten Fall ist es aber möglich, dass die Residuen konstant bleiben und erst im allerletzten Schritt auf null fallen.

Ist die Matrix positiv definit, so kann die Konvergenz des Residuums durch den betragsmäßig kleinsten $\lambda_\mathrm{min}(\mathbf{A})$ und größten $\lambda_\mathrm{max}(\mathbf{A})$ Eigenwert der Matrix $\mathbf{A}$ abgeschätzt

werden (Meister 1999):

$$||\mathbf{r}^{(k)}||_2 \leq \left(1 - \frac{\lambda_{\min}^2\left(\frac{\mathbf{A}^{\mathrm{T}}+\mathbf{A}}{2}\right)}{\lambda_{\max}\left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)}\right)^{\frac{k}{2}} ||\mathbf{r}^{(0)}||_2. \tag{4.38}$$

Ist die Matrix zusätzlich noch symmetrisch, folgt eine Abschätzung unter Verwendung der Konditionszahl κ der Matrix $\mathbf{A}$, die, wie bereits beim CG-Verfahren, den Vorteil einer kleinen Konditionszahl verdeutlicht.

$$||\mathbf{r}^{(k)}||_2 \leq \left(\frac{\kappa^2 - 1}{\kappa^2}\right)^{\frac{k}{2}} ||\mathbf{r}^{(0)}||_2 \tag{4.39}$$

Diese Konvergenzaussagen gelten jedoch nur für die Residuen und geben damit keine Auskunft über den tatsächlichen Fehler $\mathbf{e}^{(k)}$, zu dem keine Aussagen bekannt sind.

4.5 Vorkonditionierung

Der Einfluss der Konditionszahl der Systemmatrix $\mathbf{A}$ auf das Konvergenzverhalten der beschriebenen Iterationsverfahren wurde in den vorherigen Abschnitten durch Konvergenzaussagen verdeutlicht. Durch äquivalente Umformulierung des Gleichungssystems kann die Konditionszahl der Matrix verringert werden, ohne dass die Lösung verändert wird. Diese Techniken werden als Vorkonditionierung bzw. Präkonditionierung bezeichnet und haben sich als effizientes Mittel zur Beschleunigung und Stabilisierung von Krylow-Unterraum-Verfahren erwiesen. Neben der Verwendung eines geeigneten Iterationsverfahrens ist die Wahl des Vorkonditionierers von entscheidender Bedeutung für die Konvergenzgeschwindigkeit und somit die Effizienz des Gesamtverfahrens.

Zur Minimierung der Konditionszahl der Systemmatrix $\mathbf{A}$ wird das Gleichungssystem (4.1) mithilfe der regulären Matrizen $\mathbf{P}_{\mathrm{L}}$ und $\mathbf{P}_{\mathrm{R}}$ in das folgende vorkonditionierte System umgewandelt:

$$\begin{aligned} \mathbf{P}_{\mathrm{L}}\mathbf{A}\mathbf{P}_{\mathrm{R}}\mathbf{x}^{\mathrm{P}} &= \mathbf{P}_{\mathrm{L}}\mathbf{b} \\ \mathbf{x} &= \mathbf{P}_{\mathrm{R}}\mathbf{x}^{\mathrm{P}}. \end{aligned} \tag{4.40}$$

$\mathbf{P}_{\mathrm{L}}$ heißt linker Vorkonditionierer und wenn $\mathbf{P}_{\mathrm{R}} = \mathbf{I}$ gilt, heißt das Gleichungssystem linksvorkonditioniert. Umgekehrt wird das System als rechtsvorkonditioniert bezeichnet, wenn $\mathbf{P}_{\mathrm{L}} = \mathbf{I}$ gilt. $\mathbf{P}_{\mathrm{R}}$ heißt rechter Vorkonditionierer. Ein links- und rechtsvorkonditioniertes System heißt beidseitig oder zentral vorkonditioniert.

Es gibt bisher wenige allgemeine theoretische Ergebnisse, die es erlauben einen optimalen Vorkonditionierer zu konstruieren. Im Allgemeinen wird davon ausgegangen, dass der Vorkonditionierer die Inverse der Matrix $\mathbf{A}$ möglichst genau approximieren sollte, damit die Eigenwerte von $\mathbf{P}_\mathrm{L}\mathbf{A}$ bzw. $\mathbf{A}\mathbf{P}_\mathrm{R}$ nahe bei eins liegen.

Vorkonditionierer lassen sich in zwei Hauptklassen, die algebraischen und die funktionalen Vorkonditionierer, aufteilen. Funktionale Verfahren nutzen im Gegensatz zu den algebraischen Methoden die Kenntnis der Problemstellung, um effiziente Vorkonditionierer zu konstruieren. Ein solches Verfahren ist zum Beispiel das „scaled director conditioning“ für Diskretisierungen von dünnwandigen Schalen, das die Besonderheiten in Dickenrichtung der Schale zur Vorkonditionierung nutzt (Gee 2004). Algebraische Methoden bilden den Vorkonditionierer unabhängig vom zu lösenden Problem rein aus der Systemmatrix $\mathbf{A}$. Im Weiteren werden verschiedene algebraische Vorkonditionierer vorgestellt, die sich zur Anwendung auf Problemstellungen der CFD eignen.

In der Praxis wird weiterhin zwischen expliziten und impliziten Vorkonditionierern unterschieden. Damit wird ausgedrückt, dass entweder die Vorkonditionierungsmatrix explizit zur Verfügung steht oder nur die Wirkungsweise des Vorkonditionierers bei einer Matrix-Vektor-Multiplikation bekannt ist. Die meisten der vorgestellten Verfahren werden in der Regel als implizite Vorkonditionierer implementiert.

4.5.1 Splitting-assoziierte Vorkonditionierer

Eine große Gruppe von Vorkonditionierern bilden die Splitting-Verfahren. Wie im Abschnitt 4.3 bereits erwähnt, stellt die Korrektormatrix $\mathbf{C}$ eine leicht zu invertierende Approximation der Systemmatrix $\mathbf{A}$ dar und eignet sich daher auch als Vorkonditionierungsmatrix.

Die splitting-assoziierten Vorkonditionierer lassen sich sowohl zur linken als auch zur rechten Vorkonditionierung verwenden. Bei der Implementierung kann ausgenutzt werden, dass die zu invertierenden Matrizen stets eine Diagonal- oder Dreiecksform aufweisen. Die Inversen müssen daher nicht explizit aufgestellt werden, sondern bei der Berechnung des Matrix-Vektor-Produkts kann die entsprechende Eliminationstechnik, wie Vorwärts- und Rückwärtseinsetzen, genutzt werden.

In Tabelle 4.1 sind die Vorkonditionierungsmatrizen der zuvor beschriebenen Splitting-Methoden noch einmal zusammengefasst.

Splitting-Methode	Vorkonditionierungs-Matrix
Jacobi-Verfahren	$\mathbf{P}_{\mathrm{J}} = \mathbf{D}_{\mathrm{A}}^{-1}$
Gauß-Seidel-Verfahren	$\mathbf{P}_{\mathrm{GS}} = (\mathbf{D}_{\mathrm{A}} - \mathbf{E}_{\mathrm{A}})^{-1}$
Symm. Gauß-Seidel-Verfahren	$\mathbf{P}_{\mathrm{SGS}} = (\mathbf{D}_{\mathrm{A}} - \mathbf{F}_{\mathrm{A}})^{-1}\mathbf{D}_{\mathrm{A}}(\mathbf{D}_{\mathrm{A}} - \mathbf{E}_{\mathrm{A}})^{-1}$
SOR-Verfahren	$\mathbf{P}_{\mathrm{SOR}} = \omega(\mathbf{D}_{\mathrm{A}} - \omega\mathbf{E}_{\mathrm{A}})^{-1}$

Tabelle 4.1: Splitting-assoziierte Vorkonditionierer.

4.5.2 Unvollständige LU-Zerlegung

Die LU-Zerlegung stellt ein direktes Verfahren zur Lösung von linearen Gleichungssystemen dar. Die Systemmatrix $\mathbf{A}$ wird multiplikativ in eine linke untere $\mathbf{L}_{\mathrm{A}}$ und eine rechte obere $\mathbf{U}_{\mathrm{A}}$ Dreiecksmatrix zerlegt. Durch Vorwärts- und Rückwärtseinsetzen kann das Gleichungssystem für beliebig viele rechte Seiten effizient gelöst werden. Da die beiden Dreiecksmatrizen in der Regel vollbesetzt sind, eignet sich das Verfahren nicht für große, schwachbesetzte Matrizen.

Die Idee der unvollständigen LU-Zerlegung („incomplete LU“, ILU) ist, eine Zerlegung der Form $\mathbf{A} = \tilde{\mathbf{L}}_{\mathrm{A}}\tilde{\mathbf{U}}_{\mathrm{A}} + \mathbf{F}_{\mathrm{A}}$ zu erstellen, wobei der Speicherbedarf der beiden Dreiecksmatrizen identisch zu dem der Matrix $\mathbf{A}$ sein soll. Die Vernachlässigung der Matrix $\mathbf{F}_{\mathrm{A}}$ führt auf eine leicht invertierbare Approximation der Matrix $\mathbf{A}$, die sich zur Vorkonditionierung des Gleichungssystems eignet.

$$\mathbf{P}_{\mathrm{ILU}} = (\tilde{\mathbf{L}}_{\mathrm{A}}\tilde{\mathbf{U}}_{\mathrm{A}})^{-1} \tag{4.41}$$

Auch hier wird die Vorkonditionierungsmatrix $\mathbf{P}_{\mathrm{ILU}}$ nicht explizit berechnet, sondern nur ihre Wirkung in einem Matrix-Vektor-Produkt durch Vorwärtseinsetzen mit $\tilde{\mathbf{L}}_{\mathrm{A}}$ und anschließendes Rückwärtseinsetzen mit $\tilde{\mathbf{U}}_{\mathrm{A}}$ bestimmt.

Eine Variante, die unvollständige LU-Zerlegung mit „fill-in“, erweist sich häufig bei Matrizen mit geringer Bandbreite als effektiv. Hier werden mehr Matrixeinträge in den Dreiecksmatrizen zugelassen als in der Matrix $\mathbf{A}$ vorhanden sind. Die Fehlermatrix $\mathbf{F}_{\mathrm{A}}$ enthält dadurch weniger Einträge und $\tilde{\mathbf{L}}_{\mathrm{A}}\tilde{\mathbf{U}}_{\mathrm{A}}$ bildet eine bessere Approximation der Systemmatrix. Bei der Variante ILU(p) bezeichnet p den Grad (Level) des „fill-in“ und $p = 0$ entspricht der normalen ILU-Zerlegung. Beim Level eins werden Matrixeinträge zugelassen, die der Besetzungsstruktur des Produkts der Matrizen $\tilde{\mathbf{L}}_{\mathrm{A}}$ und $\tilde{\mathbf{U}}_{\mathrm{A}}$ aus der ILU(0) entsprechen. Die höheren Level werden rekursiv in derselben Weise definiert. Die Bestimmung der Besetzungsstruktur der fertigen Zerlegung ILU(p) ist vorab möglich und ermöglicht eine einfache Implementierung. Bei der ILUT-Zerlegung (ILU mit

„threshold“) werden Einträge in der Zerlegung zugelassen, die eine gewisse Toleranz überschreiten. Dadurch wird gewährleistet, dass nur relevante Einträge in der Zerlegung berücksichtigt werden. Im Vergleich zum ILU(p) ist eine effiziente Implementierung hierbei schwieriger, da die Belegung der Zerlegung vorab nicht bekannt ist.

4.5.3 Block-Vorkonditionierer

Bisher wurden die Vorkonditionierer und Iterationsverfahren in der klassischen punktbasierten Schreibweise vorgestellt. Jede Gleichung, d.h. jede Zeile des Gleichungssystems, wird einzeln betrachtet, wie z.B. beim Jacobi-Verfahren in Komponentenschreibweise in Gleichung (4.12) deutlich wird. Es ist aber generell auch möglich diese Verfahren als Block-Variante zu betrachten. Dabei werden mehrere Gleichungen zusammengefasst, d.h. die Systemmatrix wird in Submatrizen, in Blöcke, aufgeteilt. Die Iterationsvorschrift arbeitet dann mit diesen Blöcken und nicht mehr mit einzelnen Gleichungen bzw. Matrixeinträgen.

Für das Block-Jacobi-Verfahren würde die Iterationsvorschrift wie folgt lauten (vgl. (4.12)):

$$\mathbf{x}_i^{(k+1)} = \mathbf{a}_{ii}^{-1} \left(\mathbf{b}_i - \sum_{\substack{j=1 \\ j \neq i}}^{n_\mathrm{b}} \mathbf{a}_{ij} \mathbf{x}_j^{(k)} \right) \qquad i = 1, \ldots, n_\mathrm{b}. \tag{4.42}$$

Dabei steht $\mathbf{a}_{ij}$ für einen Block, eine Submatrix der Systemmatrix $\mathbf{A}$, und $\mathbf{x}_i$ bzw. $\mathbf{b}_i$ für einen entsprechenden Abschnitt der Vektoren $\mathbf{x}$ und $\mathbf{b}$. n_b bezeichnet die Anzahl der Diagonalblöcke der Matrix $\mathbf{A}$.

Für Gleichungssysteme, die aus einer Diskretisierung mit Finiten Elementen resultieren, folgt eine naheliegende Einteilung in Blöcke aus der Zuordnung der Freiheitsgrade zu den Knoten. In den meisten Fällen hat jeder Knoten mehrere Freiheitsgrade, so dass es sinnvoll ist, jeweils alle Freiheitsgrade eines Knotens zu einem Block zusammenzufassen.

Wird die Blockstruktur bereits bei der Speicherung der Systemmatrix berücksichtigt, lässt sich durch die Verwendung von Block-Vorkonditionierern die indirekte Adressierung z.B. für eine ILU-Vorkonditionierung und die Matrix-Vektor-Produkte stark reduzieren. Werden z.B. jeweils vier Freiheitsgrade zu einem Block zusammengefasst, wie es in der in dieser Arbeit verwendeten Fluid-Formulierung im 3-D der Fall ist, ergeben sich Submatrizen mit jeweils 16 Einträgen. Statt vorher 16 ist nur noch eine indirekte Adressierung erforderlich, um diese Matrixeinträge im Matrix-Vektor-Produkt zu verwenden.

4.6 Vergleich der Effizienz

In der Literatur zur CFD finden sich zahlreiche Arbeiten über die Effizienz von Lösern und die damit zusammenhängende Wahl des Iterationsverfahrens und Vorkonditionierers. In vielen Fällen werden dabei bekannte Verfahren in unterschiedlichen Kombinationen und mit verschiedenen Parametern verwendet. Teilweise werden auch spezielle Verfahren vorgestellt, die nur für die verwendete Formulierung eingesetzt werden können. Ein Beispiel dafür sind die Arbeiten von Vuik u. a. (2000) oder Vuik und Saghir (2002), in denen eine Krylow-basierte Beschleunigung des SIMPLE(R)-Algorithmus, einer Druck-Korrektur-Methode zur Lösung der diskreten Navier-Stokes-Gleichungen, vorgestellt wird.

Dahl und Wille (1992) haben das BiCGSTAB-Verfahren mit Vorkonditionierern, die auf dem Jacobi-Verfahren, der Gauß-Seidel-Relaxationsmethode und der ILU-Zerlegung basieren, kombiniert. Bei der Lösung eines linearen Gleichungssystems, das aus der Diskretisierung der Navier-Stokes-Gleichungen mit einer gemischten Finite-Element-Formulierung resultiert, hat eine Vorkonditionierung mit einer ILU-Zerlegung des symmetrischen Anteils der Stokes-Matrix die besten Ergebnisse gezeigt.

Für die inkompressiblen Navier-Stokes-Gleichungen wurde von Vuik u. a. (2001) die Verwendung des Krylow-Unterraum-Verfahrens GCR („generalized conjugate residual“) in Verbindung mit einer Block-Gauß-Jacobi-Vorkonditionierung vorgeschlagen, während Marques und Pereira (2004) für kompressible Strömungen und einen GMRES-Löser die besten Ergebnisse mit einer Jacobi- und symmetrischen Gauß-Seidel-Vorkonditionierung erzielten.

Die Ergebnisse dieser Vergleiche zeigen deutlich, dass es keine eindeutige Antwort auf die Frage nach dem besten Löser für CFD-Problemstellungen gibt. Die Effizienz der einzelnen Verfahren hängt von vielen Parametern ab, z.B. den zugrunde liegenden Gleichungen (kompressibel, inkompressibel, Stokes, Navier-Stokes) und den Randbedingungen. Wird nicht nur die Konvergenzgeschwindigkeit, sondern auch die Rechenzeit verglichen, spielt zusätzlich neben der verwendeten Implementierung auch die Hardware-Architektur eine bedeutende Rolle. Eine wichtige Erkenntnis aus diesen Arbeiten ist, dass die Vorkonditionierung zur Lösung von Gleichungssystemen, die aus der Diskretisierung der Navier-Stokes-Gleichungen stammen, zwingend erforderlich ist (Souläimani u. a. 2002) und einen sehr großen Einfluss auf die Effizienz des Lösers hat.

Für die in dieser Arbeit verwendete Diskretisierung der inkompressiblen Navier-Stokes-Gleichungen mit stabilisierten Finiten Elementen wird anhand des folgenden Beispiels gezeigt, wie verschiedene Parameter die Effizienz des linearen Lösers beeinflussen. Variiert werden dabei sowohl direkte Löserparameter (z.B. Vorkonditionierer, Relaxati-

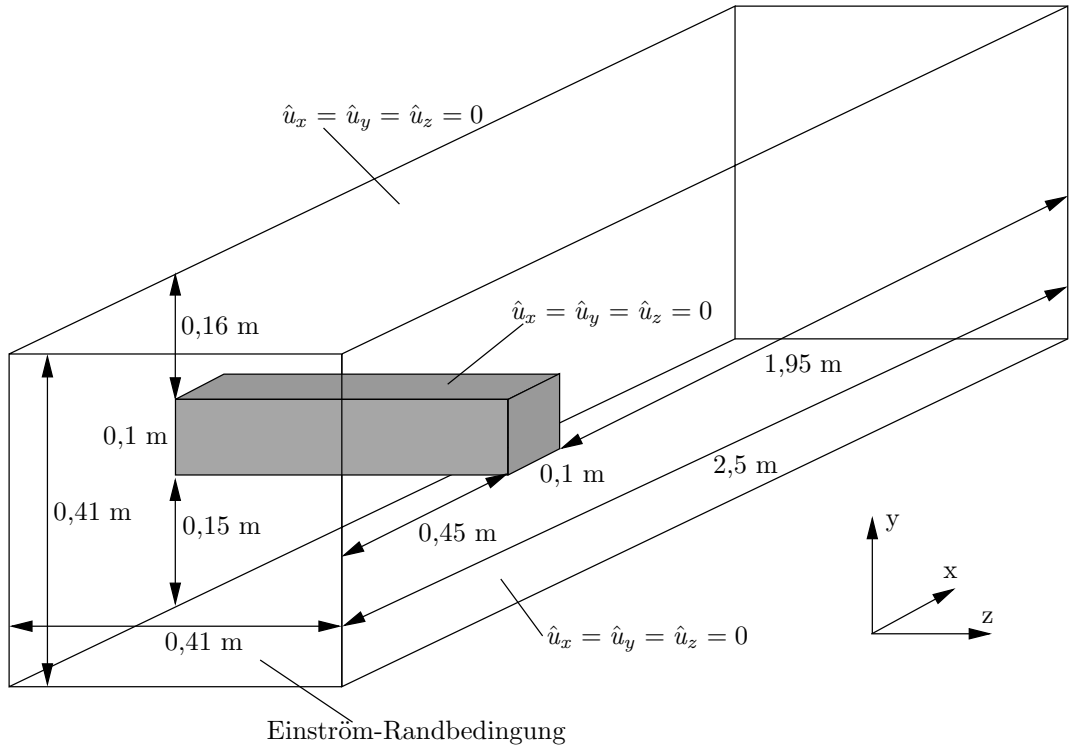


Abbildung 4.1: Geometrie und Randbedingungen eines umströmten Zylinders mit quadratischer Grundfläche.

onsparameter, Größe des Krylow-Unterraums) als auch Kenngrößen der Diskretisierung (z.B. Zeitschrittweite, Stabilisierung).

4.6.1 Problemstellung

Als Beispiel für diese Untersuchungen wurde eine reine Fluid-Berechnung, eine laminare, stationäre Strömung um einen Zylinder, gewählt. Die Aufgabenstellung stammt aus der Benchmarksammlung des DFG-Schwerpunktprogramms „Flow Simulation on High Performance Computers“ und wurde u.a. von Schäfer und Turek (1996) beschrieben. Es handelt sich um eine laminare Strömung in einem Kanal mit quadratischem Querschnitt mit Haft-Randbedingungen an den vier Längsseiten des Kanals. Im vorderen Bereich des Kanals wird dabei ein Zylinder mit quadratischer Grundfläche umströmt. Die genaue Geometrie mit allen Randbedingungen ist der Abbildung 4.1 zu entnehmen.

Die Einströmung ist durch ein in beide Richtungen parabelförmiges Profil gegeben, dessen Maximalwert $\hat{u}_{x,\max}$ linear in der Zeit gesteigert wird:

$$\hat{u}_x(0,y,z) = 16\hat{u}_{x,\max} yz(0{,}41 - y)(0{,}41 - z)/0{,}41^4; \qquad \hat{u}_y = \hat{u}_z = 0. \tag{4.43}$$

Die kinematische Viskosität beträgt $\nu_F = 10^{-3}$ m^2/s und die Dichte $\rho_F = 1{,}0$ kg/m^3.

Das Fluid wurde mit 33.570 linearen, stabilisierten Elementen diskretisiert. Dabei wird eine SUPG/PSPG-Stabilisierung mit den in Gleichungen (2.56) und (2.57) angegebenen Stabilisierungsparametern nach Förster (2007) und Codina (2002) verwendet. Nach der Berücksichtigung der Randbedingungen ergibt sich daraus ein lineares Gleichungssystem mit 151.040 Unbekannten. Die folgenden Untersuchungen wurden am 10. Zeitschritt ($\Delta t = 0{,}025$ s) durchgeführt, bei dem die Reynolds-Zahl einen Wert von etwa $\mathrm{Re} = 4$ annimmt.

4.6.2 „Block-based Linear Iterative Solver“ (BLIS)

Zum Vergleich der Lösereffizienz wird der lineare Löser BLIS („Block-based Linear Iterative Solver“) verwendet, der am Höchstleistungsrechenzentrum Stuttgart (HLRS) im Rahmen der Teraflop-Workbench speziell für den in Stuttgart verfügbaren Hochleistungs-Vektorrechner NEC SX-8 (SX-8 Homepage 2008) entwickelt wird. Die Teraflop-Workbench (Teraflop Workbench Homepage 2008) ist eine Kooperation zwischen dem HLRS, NEC und verschiedenen Universitätsinstituten, deren Ziel es ist zu zeigen, dass auf dem SX-8-Hochleistungscomputer mit verschiedenen Anwender-Codes eine „sustained performance“ von über 10^{12} FLOPS (Tera-FLOPS) erreicht werden kann. Neben der SX-8-Plattform werden auch Linux-Cluster zur Verfügung gestellt, um einen integrierten, effizienten Arbeitsablauf vom Präprozessor über die Simulation bis zu dem Postprozessor und der Visualisierung zu ermöglichen.

Durch die konsequente Verwendung von Blöcken beim Matrix-Vektor-Produkt und im Vorkonditionierer kann der Löser BLIS die indirekte Adressierung im Löserkern erheblich reduzieren (Tiyyagura und Küster 2007). Durch die Verwendung des JAD-Formats („jagged diagonal“) für dünnbesetzte Matrizen, das auf Pseudo-Diagonalen basiert, und „loop unrolling“ innerhalb der Blöcke, können zusätzlich relativ große Vektorlängen erreicht werden (Tiyyagura u. a. 2006). BLIS stellt neben den klassischen Krylow-Unterraum-Verfahren (CG, BiCGSTAB und GMRES) verschiedene blockbasierte Vorkonditionierer (Jacobi, symmetrischer Gauß-Seidel (SGS), ILU) zur Verfügung (Tiyyagura und von Scheven 2007). Bis auf das CG-Verfahren, das nur für symmetrische Matrizen geeignet ist, werden die beiden Krylow-Verfahren in Kombination mit den drei genannten Vorkonditionierern in den folgenden Untersuchungen verwendet.

Alle folgenden Ergebnisse bezeichnen immer die Rechenzeit bzw. die Anzahl der benötigten Iterationen des Krylow-Unterraum-Verfahrens für das Lösen des linearen Gleichungssystems der ersten nichtlinearen Iteration eines Zeitschritts. Sofern nicht anders angegeben, wurden die Berechnungen auf einem Knoten, d.h. auf acht Prozessoren des NEC SX-8-Vektorrechners am HLRS durchgeführt.

4.6.3 Einfluss der Löserparameter

Bei dieser ersten Reihe von Untersuchungen wird davon ausgegangen, dass das zu lösende Gleichungssystem unverändert bleibt. Variiert werden nur einzelne Parameter des linearen Lösers, wobei alle Kenngrößen, die nicht explizit genannt werden, konstant gehalten werden. Ziel der Untersuchungen ist zunächst, für jede Kombination aus Iterationsverfahren und Vorkonditionierer die jeweils besten Parameter zu finden. In einem zweiten Schritt werden dann diese „optimierten“ Kombinationen untereinander verglichen.

In den folgenden Diagrammen und Tabellen wird in Klammern für das GMRES-Verfahren die verwendete Größe des Krylow-Unterraums und für die splitting-assoziierten Vorkonditionierer die Anzahl der Vorkonditionierungs-Iterationen angegeben. Der Relaxationsparameter wird, wenn erforderlich, nach dem Krylow-Verfahren und Vorkonditionierer als dritter Löserparameter angegeben. So bezeichnet „GMRES(120), SGS(2), 0,9“ eine GMRES-Iteration mit Unterraumgröße 120 in Kombination mit zwei Iterationen symmetrischer Gauß-Seidel-Vorkonditionierung mit einem Relaxationsparameter $\omega = 0{,}9$.

Anzahl der Vorkonditionierer-Iterationen

Die erste Kenngröße, die einen sehr großen Einfluss auf die Effizienz des Lösers haben kann, ist die Anzahl der Iterationen eines splitting-assoziierten Vorkonditionierers. Je mehr Iterationen ausgeführt werden, desto besser wird die Approximation der inversen Systemmatrix durch den Vorkonditionierer, d.h. die Konditionierung des resultierenden Systems wird besser und das Krylow-Verfahren benötigt weniger Iterationen. Andererseits steigt aber auch der Rechenaufwand für den Vorkonditionierer nicht unerheblich. Dieses Verhalten lässt sich an den Messungen der Vergleichsrechnungen erkennen. Hierbei wurde unter Verwendung des BiCGSTAB-Verfahrens nur die Anzahl der Iterationen des symmetrischen Gauß-Seidel-Vorkonditionierers variiert. Abbildung 4.2 stellt als Säulendiagramm die Anzahl der benötigten Iterationen des Krylow-Verfahrens dar, während die Linie die Rechenzeit für das Lösen des Gleichungssystems angibt. Es zeigt sich deutlich, dass besonders für wenige SGS-Iterationen eine Erhöhung der Anzahl die Qualität des Vorkonditionierers erheblich verbessert und die Zahl der Krylow-Iterationen deutlich senkt. Da die Rechenzeit trotz abnehmender Anzahl an Krylow-Iterationen kontinuierlich steigt, ist zu erkennen, dass der Rechenaufwand des Vorkonditionierers dominant im Vergleich zum eigentlichen Iterationsverfahren ist.

Die Verläufe für die Anzahl der Krylow-Iterationen und die Rechenzeit sehen sowohl für die Kombination des SGS-Vorkonditionierers mit dem GMRES-Verfahren als auch für das BiCGSTAB-Verfahren mit einer Jacobi-Vorkonditionierung analog aus. Da die

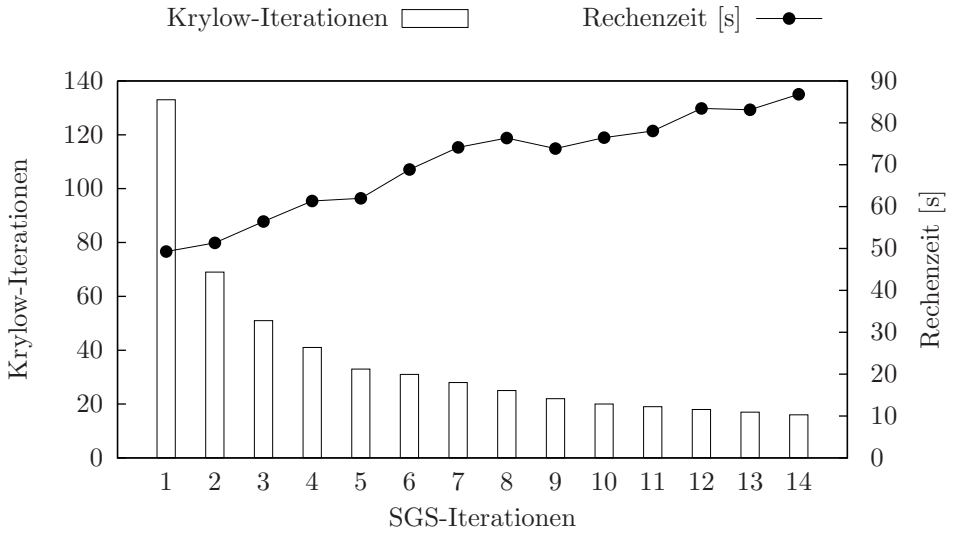


Abbildung 4.2: BiCGSTAB und SGS ($\omega = 0{,}9$): Einfluss der Anzahl der Vorkonditionierer-Iterationen auf Rechenzeit und Iterationszahl.

zweite Iteration des Vorkonditionierers den größten Effekt auf die Konvergenzgeschwindigkeit hat und die Rechenzeit im Vergleich zu nur einer Iteration annähernd gleich bleibt, werden für die drei bisher genannten Kombinationen aus Krylow-Verfahren und Vorkonditionierer in allen folgenden Rechnungen jeweils zwei Iterationen des Vorkonditionierers verwendet.

Das GMRES-Verfahren mit Jacobi-Vorkonditionierung zeigt ein abweichendes Verhalten. In den Verläufen für die Anzahl der Krylow-Iterationen und die Rechenzeit in

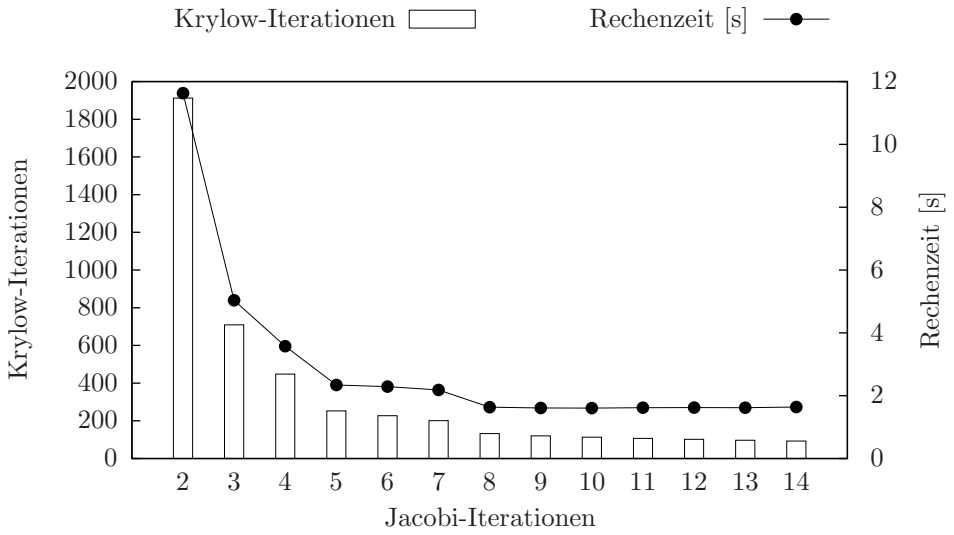


Abbildung 4.3: GMRES(120) und Jacobi ($\omega = 0{,}4$): Einfluss der Anzahl der Vorkonditionierer-Iterationen auf Rechenzeit und Iterationszahl.

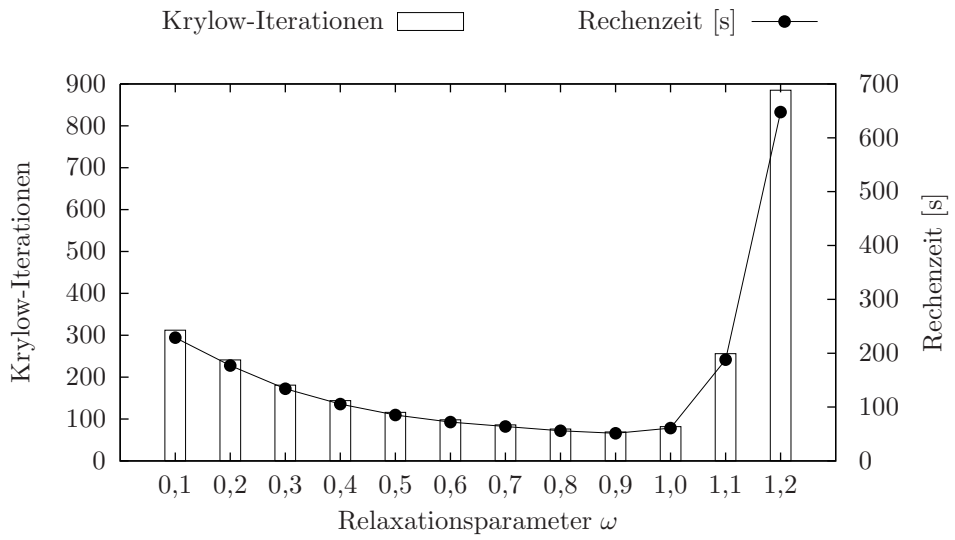


Abbildung 4.4: BiCGSTAB und SGS(2): Einfluss des Relaxationsparameters ω auf Rechenzeit und Iterationszahl.

Abbildung 4.3 wird deutlich, dass hier der Rechenaufwand für eine GMRES-Iteration dominant ist gegenüber den Jacobi-Iterationen. Die Rechenzeit verhält sich nahezu proportional zur Anzahl der Krylow-Iterationen. Erst für mehr als 10 Jacobi-Iterationen wird der Rechenaufwand des Vorkonditionierers deutlich: Die Gesamtrechenzeit steigt leicht an, während die Anzahl der GMRES-Iterationen weiterhin sinkt. Das Minimum der Rechenzeit wird für 10 Jacobi-Iterationen erreicht und so wird dieser Wert in allen folgenden Untersuchungen verwendet.

Relaxationsparameter

Wie bereits bei den Splitting-Verfahren hat der Relaxationsparameter ω bei den zugehörigen Vorkonditionierern einen erheblichen Einfluss auf die Effektivität und damit auf die Konvergenzgeschwindigkeit. Dabei bleibt der Rechenaufwand für den Vorkonditionierer für alle Werte von ω unverändert.

Abbildung 4.4 zeigt die Anzahl der benötigten Iterationen des BiCGSTAB-Verfahrens und die Rechenzeit für das Lösen des Gleichungssystems mit einem SGS-Vorkonditionierer bei unterschiedlichen Werten für ω. Der SGS-Vorkonditionierer erreicht bei einer Unterrelaxation für alle ω eine gute Konditionierung des Gleichungssystems. Aber bereits bei einer Überrelaxation mit $\omega = 1{,}2$ zeigt sich, dass die Konditionierung des resultierenden Gleichungssystems deutlich schlechter geworden ist und das BiCGSTAB-Verfahren mehr Iterationen benötigt. Für Werte $\omega > 1{,}2$ ist das System so schlecht konditioniert, dass das Krylow-Verfahren nicht mehr in der vorgegebenen maximalen

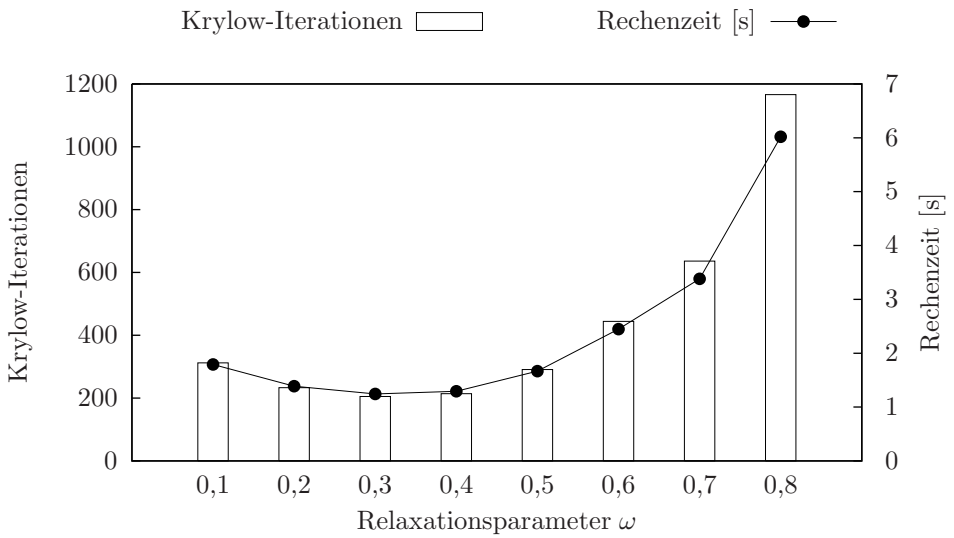


Abbildung 4.5: BiCGSTAB und Jacobi(2): Einfluss des Relaxationsparameters ω auf Rechenzeit und Iterationszahl.

Anzahl an Iterationen konvergiert. Die schnellste Lösung des Gleichungssystems wird in diesem Beispiel mit $\omega = 0{,}9$ erreicht.

Das Verhalten stimmt für beide untersuchten Krylow-Verfahren überein, so dass sowohl für das BiCGSTAB- als auch das GMRES-Verfahren in Kombination mit dem symmetrischen Gauß-Seidel-Vorkonditionierer in den vorangehenden und folgenden Rechnungen ein Wert $\omega = 0{,}9$ verwendet wird.

Mit einer Jacobi-Vorkonditionierung wird nur für relativ kleine Werte von ω die Konvergenz des Krylow-Verfahrens erreicht. Der vorkonditionierende Effekt des Jacobi-Verfahrens ist nicht so groß wie beim symmetrischen Gauß-Seidel-Verfahren, so dass in diesem Beispiel bei Verwendung einer BiCGSTAB-Iteration nur Relaxationsparameter $\omega \leq 0{,}8$ verwendet werden können (Abbildung 4.5). Dabei wird für eine Relaxation von $\omega = 0{,}3$ das Gleichungssystem am schnellsten gelöst. Das GMRES-Verfahren konvergiert sogar nur für Werte $\omega \leq 0{,}4$ und erreicht mit $\omega = 0{,}4$ die geringste Rechenzeit. Diese beiden Werte werden in allen anderen Vergleichen dieses Abschnitts als Relaxationsparameter für die Jacobi-Vorkonditionierung verwendet.

Die direkte Übereinstimmung der Kurven für die Rechenzeit und die benötigten Krylow-Iterationen bei veränderlichem ω (Abbildungen 4.4 und 4.5) zeigt deutlich, dass der Rechenaufwand für eine Krylow-Iteration unabhängig von der Wahl von ω ist und damit die Gesamtrechenzeit nur von der Anzahl der benötigten Krylow-Iterationen abhängt.

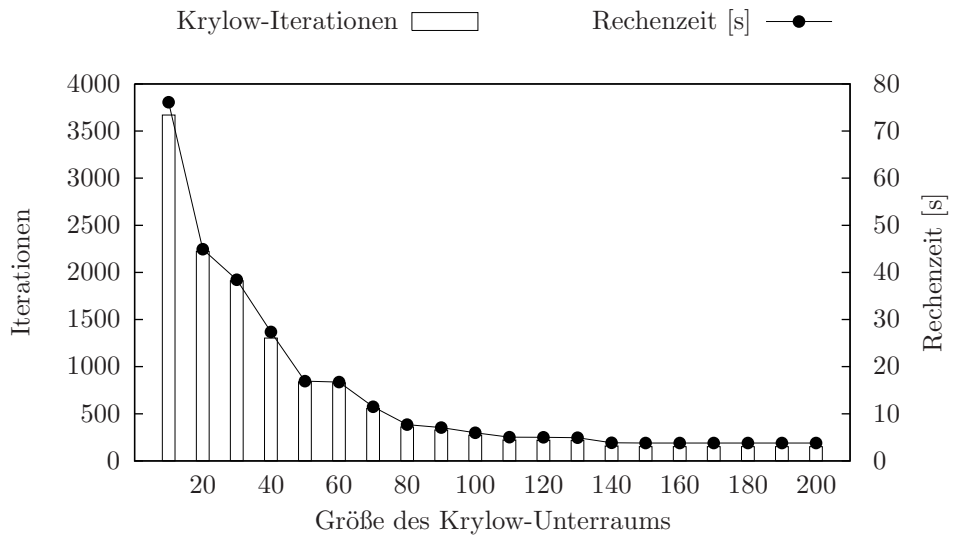


Abbildung 4.6: GMRES und BILU: Einfluss der Größe des Krylow-Unterraums auf Rechenzeit und Iterationszahl.

Größe des Krylow-Unterraums

Da das GMRES-Verfahren wie allgemein üblich in einer Variante mit Neustarts, d.h. mit einer beschränkten Größe des Krylow-Unterraums, verwendet wird, hat auch diese Größe einen Einfluss auf das Konvergenzverhalten. Jeder Neustart und damit der Verlust der bisher verwendeten Krylow-Vektoren verlangsamt die Konvergenz. Somit benötigt die Lösung eines Gleichungssystems, bei der mehrere Neustarts nötig sind, deutlich mehr Iterationen als eine Rechnung, bei der keine Neustarts erforderlich sind, d.h. bei der die Größe des verwendeten Unterraums die Anzahl der benötigten Iterationen übersteigt. Dagegen bedeutet ein großer Krylow-Unterraum einen erheblich größeren Speicherbedarf, so dass der verwendete Unterraum so klein wie möglich gewählt werden sollte.

Abbildung 4.6 zeigt einen typischen Verlauf für die Anzahl der GMRES-Iterationen bei unterschiedlichen Größen des Unterraums. Bei sehr kleinen Unterräumen werden durch die häufigen Neustarts extrem viele Iterationen bis zur Konvergenz benötigt. Leichte Vergrößerungen des Unterraums verursachen hier eine deutliche Verbesserung der Konvergenz. Nähert sich die Größe des Unterraums der Anzahl der benötigten Iterationen an, wird die Verringerung der Iterationsanzahl bei wachsendem Unterraum immer weniger. Sobald die Größe des Unterraums die Anzahl der benötigten Iterationen überschritten hat, bleibt diese Anzahl konstant. In dieser Untersuchung ist dies ab einer Krylow-Unterraumgröße von 160 der Fall. Für größere Unterräume bleibt die Anzahl der GMRES-Iterationen mit 151 konstant, da keine Neustarts mehr erforderlich sind.

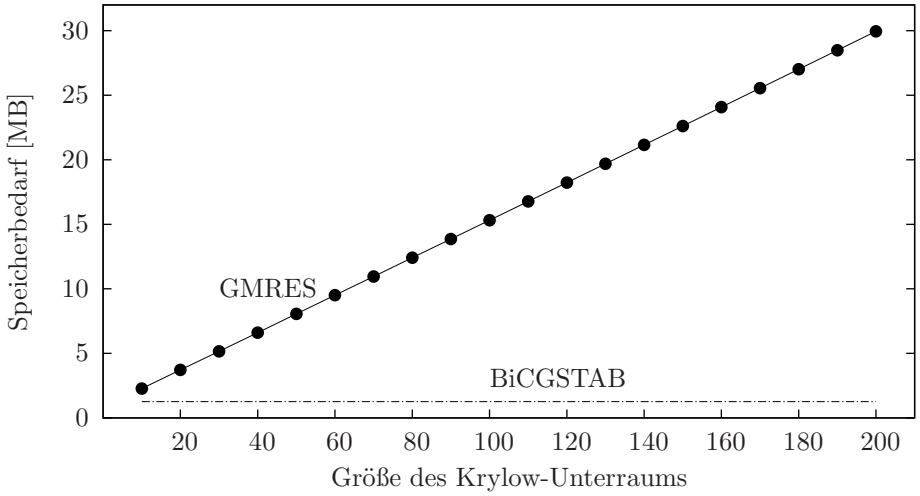


Abbildung 4.7: GMRES und BILU: Einfluss der Größe des Krylow-Unterraums auf den Speicherbedarf.

Der für die GMRES-Iteration erforderliche Speicher ist abhängig von der Größe der Systemmatrix n, der maximalen Größe des Krylow-Unterraums $m_{\max}$ sowie von der Anzahl an Freiheitsgraden n_{ext}, für die zwischen zwei Prozessen kommuniziert werden muss. Ohne den erforderlichen Arbeitsspeicher für die Systemmatrix selbst, die rechte Seite, den Lösungsvektor und den Vorkonditionierer berechnet sich der Speicherbedarf wie folgt:

$$\begin{aligned} SB_{\text{GMRES}} &= (m_{\max} + 1) \cdot (n + m_{\max} + 7) + 4 \cdot (n + n_{\text{ext}}) \\ &= m_{\max}^2 + m_{\max} n + 8 m_{\max} + [5n + 4n_{\text{ext}} + 7]\,. \end{aligned} \tag{4.44}$$

Aus Gleichung (4.44) wird ersichtlich, dass der Speicherbedarf quadratisch von der gewählten Größe des Krylow-Unterraums abhängt. Da in der Regel jedoch $n \gg m_{\max}$ gilt, dominiert der zweite Summand $m_{\max} n$ die Beziehung und es folgt eine nahezu lineare Beziehung zwischen der Größe des Krylow-Unterraums und dem Speicherbedarf, wie auch in Abbildung 4.7 zu erkennen ist.

Zum Vergleich ist in diesem Diagramm auch der Speicherbedarf der BiCGSTAB-Iteration angegeben. Dieser berechnet sich nur aus der Größe der Matrix n und der Anzahl der Freiheitsgrade n_{ext}, für die kommuniziert werden muss:

$$SB_{\text{BiCGSTAB}} = 4 \cdot n + 4 \cdot (n + n_{\text{ext}}). \tag{4.45}$$

Für die vorhergehenden und folgenden Vergleichsrechnungen wird eine Krylow-Unterraumgröße von 120 gewählt, um einen vertretbaren Kompromiss zwischen Rechenzeit

Krylow-Verfahren	Vorkonditionierer	Größe des Krylow-Unterraums	Anzahl Vorkond.-Iterationen	Relaxationsparameter ω
	BILU			
BiCGSTAB	Jacobi		2	0,3
	SGS		2	0,9
	BILU	120		
GMRES	Jacobi	120	10	0,4
	SGS	120	2	0,9

Tabelle 4.2: „Optimale“ Löserparameter für die sechs untersuchten Kombinationen von Krylow-Verfahren und Vorkonditionierer.

und stetig steigendem Speicherbedarf einzugehen. Ab dieser Unterraumgröße verbessert sich die Rechenzeit auch mit den anderen Vorkonditionierern nicht mehr wesentlich.

Zusammenfassung

In Tabelle 4.2 sind die bei den voranstehenden Untersuchungen gefundenen besten Löserparameter für die sechs untersuchten Kombinationen aus Krylow-Verfahren und Vorkonditionierer zusammengefasst. Für diese sechs Kombinationen sind in Abbildung 4.8 die benötigten Krylow-Iterationen gegen die Rechenzeit aufgetragen, um sie abschließend auch untereinander zu vergleichen.

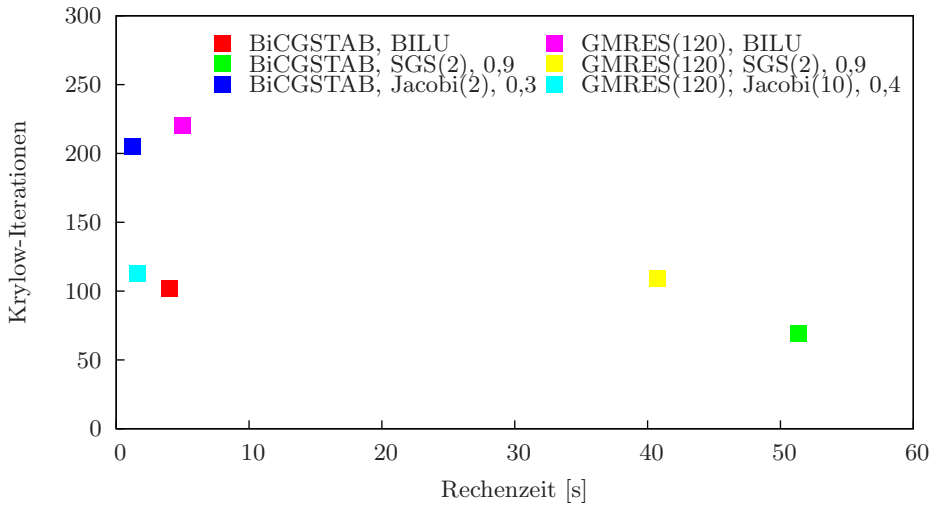


Abbildung 4.8: Vergleich der Rechenzeiten und benötigten Krylow-Iterationen für die „optimalen“ Löserparameter.

Wird nur die Rechenzeit betrachtet, ist festzustellen, dass die Wahl des Krylow-Verfahrens fast keinen Einfluss auf die Geschwindigkeit des Lösers hat. Das GMRES-Verfahren benötigt zwar bei identischem Vorkonditionierer grundsätzlich mehr Iterationen, aber nicht mehr Rechenzeit als das BiCGSTAB-Verfahren, da wie in Abschnitt 4.4 beschrieben, der Rechenaufwand pro Iteration geringer ist. Auch Zhang (2000) hat festgestellt, dass bei der Lösung von unsymmetrischen Gleichungssystemen aus CFD-Anwendungen die Wahl des Vorkonditionierers viel wichtiger ist als die Wahl des Krylow-Verfahrens.

Beim Vergleich der drei Vorkonditionierer in Abbildung 4.8 wird dies auch sofort deutlich: Der symmetrische Gauß-Seidel-Vorkonditionierer benötigt mit beiden Krylow-Verfahren bis zu zehnmal soviel Rechenzeit wie die anderen beiden Vorkonditionierer. Dabei fällt jedoch auf, dass gleichzeitig weniger Krylow-Iterationen benötigt werden, d.h. der Effekt des symmetrischen Gauß-Seidel-Vorkonditionierers, die Konvergenz des Krylow-Verfahrens zu beschleunigen, ist sehr gut, aber auf einem Vektorprozessor sehr zeitaufwändig.

Die Jacobi-Vorkonditionierung erreicht dagegen mit beiden Krylow-Verfahren die geringsten Rechenzeiten und geht daher eigentlich als bevorzugtes Verfahren aus dieser Untersuchung hervor. Dabei ist aber zu beachten, dass das Erreichen von Konvergenz beim Jacobi-Vorkonditionierer sehr stark von der richtigen Wahl des Relaxationsparameters ω abhängt. In Kombination mit dem GMRES-Verfahren liegen der optimale Relaxationsparameter $\omega = 0{,}4$ und die Divergenz des Krylow-Verfahrens ab $\omega \geq 0{,}5$ unmittelbar nebeneinander.

Mit der BILU-Zerlegung als Vorkonditionierer konvergieren beide Krylow-Verfahren dagegen sehr stabil und benötigen nicht wesentlich mehr Rechenzeit als in Kombination mit der Jacobi-Vorkonditionierung.

An dieser Stelle soll noch einmal betont werden, dass diese Parameter-Kombinationen nur für das in diesem Abschnitt untersuchte Beispiel das Gleichungssystem schnellst möglich lösen. Im folgenden Abschnitt wird gezeigt, welchen Einfluss Kenngrößen der Diskretisierung auf das Verhalten des Lösers haben können. Dabei werden grundsätzlich die in Tabelle 4.2 genannten Parameter benutzt und die in Abbildung 4.8 zusammengefassten Ergebnisse als Referenz verwendet.

4.6.4 Einfluss der Diskretisierungs-Parameter

Neben der Variation der direkten Löserparameter kann auch durch die Veränderung von Diskretisierungs-Kenngrößen und damit durch eine Modifikation des zu lösenden linearen Gleichungssystems die Rechenzeit des Lösers beeinflusst werden. Dabei können bei CFD-Problemen neben der Reynolds-Zahl auch die Zeitschrittweite und die verwendeten

	BiCGSTAB						GMRES(120)					
	BILU		SGS(2) $\omega = 0{,}9$		Jacobi(2) $\omega = 0{,}3$		BILU		SGS(2) $\omega = 0{,}9$		Jacobi(2) $\omega = 0{,}4$	
Referenz	4,09	102	51,27	69	1,24	205	5,12	220	41,36	109	1,60	113
$\Delta t = 0{,}125$	3,29	78			1,30	217	3,25	127	312,61	839	1,66	116
$\Delta t = 0{,}0025$	3,32	79			1,30	215	3,25	127			1,67	116
$\mathrm{Re} = 40$	4,27	107	53,01	71	1,21	198	5,08	218	59,69	159	1,71	120
GLS	4,83	123	75,84	103	1,68	294	7,30	328	87,30	234	9,02	707

Tabelle 4.3: Einfluss der Diskretisierungsparameter auf Rechenzeit und Konvergenzgeschwindigkeit des linearen Lösers (Rechenzeit [s] und Krylow-Iterationen).

Stabilisierungsparameter einen Einfluss auf die Konditionierung des Gleichungssystems und damit die Konvergenzgeschwindigkeit des Krylow-Verfahrens haben. Im Folgenden werden die genannten Diskretisierungsparameter variiert und deren Einfluss auf die Konvergenzgeschwindigkeit und Rechenzeit der sechs im letzten Abschnitt gefundenen „optimalen“ Kombinationen aus Krylow-Verfahren und Vorkonditionierer untersucht.

In Tabelle 4.3 sind die Ergebnisse dieser Vergleiche numerisch angegeben, wobei die erste Zahl in jeder Spalte die Rechenzeit in Sekunden und die zweite die Anzahl benötigter Krylow-Iterationen angibt. Grafisch dargestellt sind diese Zahlen in Abbildung 4.9. Dabei wurden wieder die benötigten Krylow-Iterationen gegen die Rechenzeit aufgetragen.

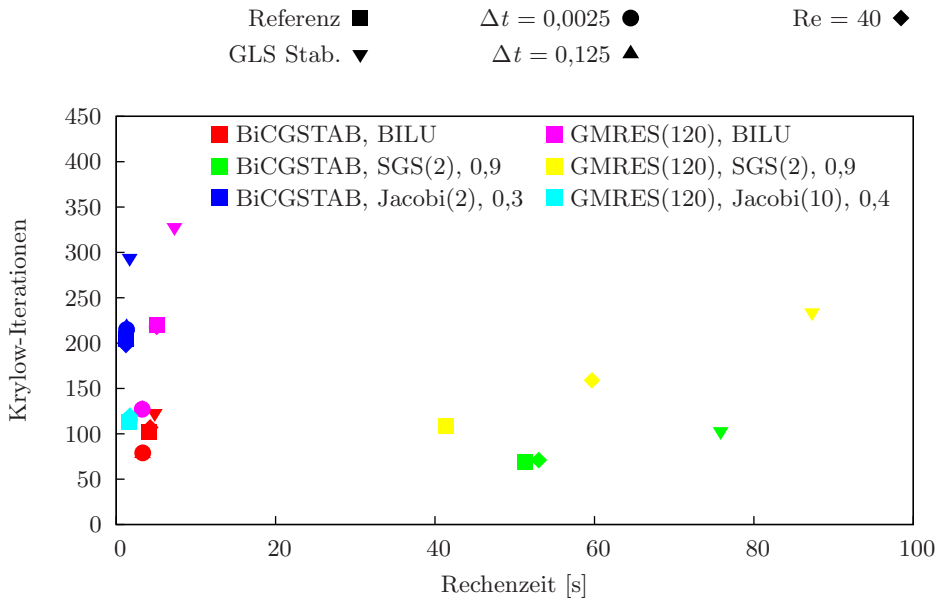


Abbildung 4.9: Einfluss der Diskretisierungsparameter auf Konvergenzgeschwindigkeit und Rechenzeit des linearen Lösers.

Aus Gründen der Übersichtlichkeit wurden die beiden grau hinterlegten Ergebnisse nicht in das Diagramm eingetragen. Als Referenz für den Vergleich dienen die im letzten Abschnitt bereits dargestellten Ergebnisse für die Rechenzeit und für die benötigten Krylow-Iterationen. Diese wurden für eine Reynolds-Zahl von $\mathrm{Re} = 4$ mit einem Zeitschritt $\Delta t = 0{,}025$ und einer SUPG/PSPG-Stabilisierung berechnet.

Reynolds-Zahl

Eine Erhöhung der Reynolds-Zahl um den Faktor 10 führt bei fast allen Kombinationen zu keiner merklichen Veränderung in der Rechenzeit im Löser und der benötigten Anzahl an Krylow-Iterationen. Einzig bei der Kombination GMRES(120) mit SGS(2) haben sich beide Größen erhöht. Auch stichprobenartige Untersuchungen für höhere Reynoldszahlen haben keinen entscheidenden Einfluss auf die Rechenzeit des linearen Lösers ergeben. Bereits Spieth (1999) war in einer am Institut für Baustatik der Universität Stuttgart angefertigten Seminararbeit zu dem gleichen Ergebnis gekommen.

Zeitschrittweite

Auch die Größe des Zeitschritts hat keinen eindeutigen Einfluss auf das Konvergenzverhalten des Lösers. So verändert weder eine Reduzierung des Zeitschritts um den Faktor 10 noch eine Vergrößerung um den Faktor 5 die benötigte Anzahl an Krylow-Iterationen oder die Rechenzeit bei einer Jacobi-Vorkonditionierung. Für den BILU-Vorkonditionierer konvergiert die Rechnung sowohl mit einem größeren als auch mit einem kleineren Zeitschritt schneller als für die Referenz-Zeitschrittgröße.

Im Gegensatz dazu hat ein größerer oder kleinerer Zeitschritt einen erheblichen Effekt auf die SGS-Vorkonditionierung. Die Rechnungen mit diesem Vorkonditionierer konvergieren bei verändertem Zeitschritt nur sehr langsam oder in vielen Fällen überhaupt nicht mehr.

Ein allgemeiner Zusammenhang zwischen der Größe des Zeitschritts und dem Konvergenzverhalten des linearen Lösers lässt sich aus diesen Ergebnissen nicht ableiten. Die Reaktion des Lösers auf einen veränderten Zeitschritt hängt sehr stark vom verwendeten Vorkonditionierer ab und variiert von schnellerer Konvergenz (BILU) über unverändertes Konvergenzverhalten (Jacobi) bis zu einem drastischen Verlangsamen der Konvergenz (SGS).

Stabilisierungsparameter

Einen deutlichen Einfluss auf das Konvergenzverhalten des linearen Lösers haben dagegen das verwendete Stabilisierungsverfahren und die Stabilisierungsparameter. Wird

	BiCGSTAB						GMRES(120)					
	BILU		SGS(2) $\omega = 0{,}9$		Jacobi(2) $\omega = 0{,}3$		BILU		SGS(2) $\omega = 0{,}9$		Jacobi(2) $\omega = 0{,}4$	
absolut												
Referenz	4,09	102	51,27	69	1,24	205	5,12	220	41,36	109	1,60	113
Core2 Duo	66,01	90	34,31	65	46,28	209	92,91	189	41,13	102	60,91	113
normiert												
Referenz	0,08	102	1,00	69	0,02	205	0,10	220	0,81	109	0,03	113
Core2 Duo	1,92	90	1,00	65	1,35	209	2,71	189	1,20	102	1,78	113

Tabelle 4.4: Einfluss der Hardware-Architektur auf Rechenzeit und Iterationszahl des linearen Lösers (Rechenzeit [s] und Krylow-Iterationen).

statt der SUPG/PSPG-Stabilisierung mit den in Gleichungen (2.56) und (2.57) angegebenen Stabilisierungsparametern nach Förster (2007) und Codina (2002) zusätzlich der viskose Anteil in der Stabilisierung mit berücksichtigt, d.h. eine GLS-Stabilisierung nach Wall (1999) verwendet, verschlechtert sich die Konvergenz deutlich. Bei allen sechs Löser-Kombinationen benötigt die Lösung des Gleichungssystems, das aus der Diskretisierung mit GLS-Stabilisierung stammt, die meisten Iterationen und auch die längste Rechenzeit (mit Ausnahme der Variante GMRES(120), SGS(2), 0,9 bei $\Delta t = 0{,}125$).

Bereits Spieth (1999) hat den großen Einfluss der Stabilisierung festgestellt, wenn auch nicht angegeben ist, in welcher Form die Stabilisierung variiert worden ist.

4.6.5 Einfluss der Hardware-Architektur

Abschließend wird an einem Vergleich der Einfluss der Hardware-Architektur auf die Rechenzeit des linearen Lösers verdeutlicht. Dazu werden die sechs Referenzrechnungen auf einem Intel Core2 Duo E6400 Prozessor wiederholt. Die Ergebnisse sind in Tabelle 4.4 dargestellt und werden mit den auf der NEC SX-8-Plattform erzielten Werten (Referenz) verglichen. Da auf dem SX-8-Vektorrechner ein gesamter Knoten mit acht Prozessoren und damit auch acht Rechenprozessen verwendet wurde und auf dem Intel Core2 Duo Prozessor mit nur zwei Prozessen auf einem Dual-Core-Prozessor gerechnet wurde, sind die absoluten Rechenzeiten nicht vergleichbar. Daher werden im zweiten Teil der Tabelle für jeden Prozessortyp die Rechenzeit an der Rechenzeit für den Fall BiCGSTAB, SGS(2), 0,9 normiert.

Diese normierten Rechenzeiten sind in Abbildung 4.10 gegen die benötigten Krylow-Iterationen aufgetragen. In diesem Diagramm ist zu erkennen, dass die Hardware-Ar-

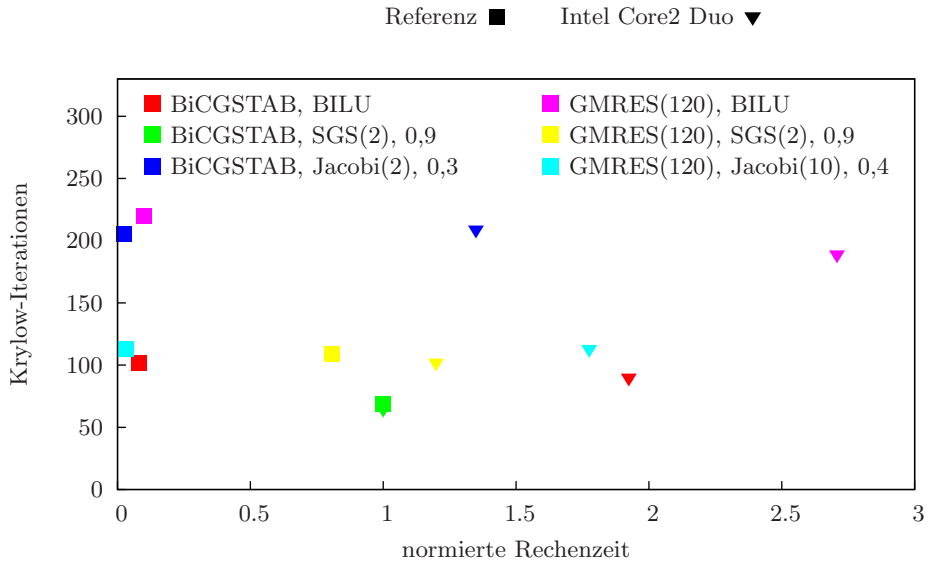


Abbildung 4.10: Einfluss der Hardware-Architektur auf Rechenzeit und Iterationszahl des linearen Lösers.

chitektur keinen Einfluss auf die Anzahl der Krylow-Iterationen hat. Die kleinen Unterschiede folgen aus den unterschiedlichen Arithmetiken in den verschiedenen Prozessoren. Großen Einfluss dagegen hat die Hardware auf die relativen Rechenzeiten der sechs Löser-Vorkonditionierer-Kombinationen. So benötigen die beiden Varianten mit SGS-Vorkonditionierung auf der NEC SX-8-Plattform am meisten Rechenzeit, sind aber auf dem Intel-Prozessor am schnellsten. Die BILU-Vorkonditionierung, die auf den SX-8-Prozessoren aufgrund ihrer Stabilität und Geschwindigkeit als bevorzugtes Verfahren aus dem Vergleich hervorgegangen ist, benötigt auf einem Intel-Prozessor mit Abstand die längste Rechenzeit.

Der Grund für die großen Unterschiede in der Effizienz der Vorkonditionierer liegt in den Datenabhängigkeiten beim SGS-Verfahren. Durch diese ist eine Vektorisierung ohne zusätzliche Hilfstechniken wie z.B. „coloring“ nicht möglich und die Vektorlänge beträgt im SGS-Vorkonditionierer nur eins. Dies führt bei der verwendeten Implementierung auf dem SX-8-Vektorprozessor zu sehr langen Rechenzeiten, hat aber auf dem Intel-Prozessor keinen Einfluss auf die Rechengeschwindigkeit. Hier zeigt die SGS-Vorkonditionierung eine ähnliche Effizienz wie der BILU- oder Jacobi-Vorkonditionierer und erreicht aufgrund der besseren Vorkonditionierungseigenschaften die geringste Gesamtrechenzeit.

Da bei der Auswahl der „optimalen“ Löserparameter nur die Rechenzeiten auf der SX-8-Plattform berücksichtigt wurden, ist es möglich, dass auf einem Intel-Prozessor mit anderen Löserparametern bessere Ergebnisse erzielt werden können oder sich die Rangfolge zwischen den sechs untersuchten Kombinationen verändert.

Abschließend kann gesagt werden, dass die richtige Wahl von Lösungsverfahren und Vorkonditionierer, aber auch der Löserparameter sehr stark von der Problemstellung und der verwendeten Hardware-Architektur abhängt. Dies kann für den Anwender gewisse Schwierigkeiten und zusätzlichen Aufwand bei der Auswahl der optimalen Parameter bedeuten, bietet dagegen aber auch die Möglichkeit, das Lösungsverfahren an die Hardware-Ressourcen anzupassen, z.B. im Fall des Speicherplatzes durch eine geeignete Wahl der Krylow-Unterraumgröße (Zhang 2000).

5

Kopplung von Fluid und Struktur

5.1 Einleitung

Aufgabenstellungen aus dem Bereich der Fluid-Struktur-Wechselwirkung gehören zur Klasse der gekoppelten Probleme, die von Zienkiewicz u. a. (2005) in einer allgemeinen Definition wie folgt charakterisiert werden:

Als gekoppelte Systeme werden Fragestellungen bezeichnet, bei denen (in mehreren Gebieten) mithilfe von verschiedenen, voneinander abhängigen Variablen häufig unterschiedliche physikalische Phänomene beschrieben werden. Dabei ist zu beachten, dass

a) kein Gebiet unabhängig von den anderen gelöst werden kann und

b) keiner der abhängigen Variablensätze bereits in den Differenzialgleichungen aus der Problemformulierung entfernt werden kann.

Diese Definition trifft strenggenommen auf die meisten Ingenieuraufgaben zu. In der Realität treten immer mehrere unterschiedliche physikalische Phänomene gleichzeitig auf. So haben zum Beispiel bei der Analyse eines Stahlbeton-Tragwerks neben den rein mechanischen Eigenschaften auch die Wärmeleitung, der Feuchtetransport und chemische Prozesse, wie die Hydratation des Zements oder die Korrosion des Stahls, einen Einfluss auf die Tragfähigkeit oder Dauerhaftigkeit. In der Praxis können diese Wechselwirkungen häufig aufgrund von theoretischen Überlegungen oder Erfahrungswerten vernachlässigt und für die einzelnen Felder separate Lösungen gefunden werden. Eine solche Entkopplung ist aber nur möglich und sinnvoll, wenn der gegenseitige Einfluss der physikalischen Phänomene nicht zu groß ist.

5.1.1 Klassifizierung von gekoppelten Problemen

Gekoppelte Probleme lassen sich anhand von verschiedenen Kriterien klassifizieren. So kann zunächst zwischen physikalischen Einfeld- und Mehrfeldproblemen unterschieden werden. Die in dieser Arbeit behandelte Fluid-Struktur-Interaktion gehört zu den Mehrfeldproblemen, da hier unterschiedliche physikalische Felder in Wechselwirkung stehen: einerseits die inkompressible Fluidströmung und andererseits die nichtlineare Strukturdynamik. Typisch für Mehrfeldprobleme sind die, aufgrund der unterschiedlichen Physik, verschiedenen Herangehensweisen bei der Modellierung und Lösung der einzelnen Felder. Bei physikalischen Einfeldproblemen wird dasselbe physikalische Problem in charakteristischen Teilbereichen des Gebiets, die sich zum Beispiel in ihrer Geometrie oder im Material unterscheiden, mit jeweils optimal angepassten Modellierungs- und Diskretisierungsmethoden gelöst.

Eine weitere Klassifizierung ist anhand der räumlichen Ausdehnung der Kopplung möglich. Bei den volumengekoppelten Mehrfeldproblemen überlappen sich die Gebiete der unterschiedlichen Felder teilweise oder vollständig. Hierzu zählen zum Beispiel Strömungen in porösen Medien oder die thermo-hygro-mechanische Modellierung von Beton (Grasberger 2003). Im Gegensatz dazu sind bei den oberflächengekoppelten Problemen die einzelnen Felder nur an den gemeinsamen Gebietsrändern über Kopplungsbedingungen miteinander verbunden. Eine typische oberflächengekoppelte Fragestellung ist die Fluid-Struktur-Interaktion, bei der das Fluid und die Struktur nur an der gemeinsamen Oberfläche über Kopplungsbedingungen interagieren.

5.1.2 Lösungsstrategien für gekoppelte Probleme

Bevor drei grundsätzliche Strategien zur Lösung von gekoppelten Problemen vorgestellt werden, folgen zunächst Kriterien, nach denen die verschiedenen Methoden beurteilt werden können. Diese Anforderungen lassen sich in drei Gruppen zusammenfassen:

Lösung: Neben der Stabilität und Genauigkeit der numerischen Lösung als wichtigste Anforderungen ist auch die Robustheit des Verfahrens im Sinne von anwenderunabhängigen Stabilitäts- und Konvergenzeigenschaften von Bedeutung.

Implementierung: Eine modulare Implementierung, bei der vorhandene und bewährte Einzelfeldlöser verwendet werden, kann die Implementierung stark vereinfachen. Neben einem möglichst geringen Rechenaufwand zur Lösung des gekoppelten Problems sind Möglichkeiten zur Parallelisierung und Vektorisierung wichtige Anforderungen an die Implementierung.

Kontinuität: Die Kontinuität der kinematischen und dynamischen Größen und damit auch der Erhalt von Masse, Impuls und Energie sollte durch das Verfahren gewährleistet werden.

Eine ausführlichere Beschreibung und Herleitung der Anforderungen, insbesondere der letzten Gruppe, findet sich in der Arbeit von Mok (2001).

Anhand dieser Anforderungen an mögliche Lösungsstrategien werden zunächst drei allgemeine Ansätze für gekoppelte Probleme nach Felippa u. a. (2001) bewertet.

Feld-Eliminations-Verfahren nutzen Integraltransformationen oder Modellreduktionen, um auf der Ebene der Differenzialgleichungen die Feldgrößen eines der Felder zu eliminieren. Dieses Lösungsverfahren ist auf sehr einfache, lineare Probleme beschränkt, die nach der oben angegebenen Definition nach Zienkiewicz u. a. (2005) auch kein gekoppeltes Problem darstellen. Die Reduzierung des Problems auf Differenzialgleichungsebene garantiert die Kontinuitätsanforderungen und eine stabile und robuste Lösung. Eine modulare Implementierung, bei der z.B. der Löser eines Felds ausgetauscht werden kann, ist jedoch nicht möglich. Da diese Art der Lösung für die Fluid-Struktur-Interaktion nicht eingesetzt werden kann, wird sie in dieser Arbeit nicht weiter berücksichtigt.

Die monolithische Lösung (auch simultane Lösung) eines gekoppelten Systems fasst alle physikalischen und algorithmischen Felder in einem System zusammen. Dabei wird das gesamte System gekoppelter nichtlinearer Differenzialgleichungen für die komplette Mehrfeldaufgabe diskretisiert und die Lösung aller Modellgleichungen simultan in einem Gleichungssystem ausgeführt. Durch den einheitlichen Lösungsansatz werden alle Wechselwirkungen zwischen den Feldern exakt berücksichtigt, wodurch sich Verfahren mit guten Stabilitätseigenschaften ergeben. Die Analyse des einheitlich formulierten Gleichungssystems ermöglicht außerdem mathematische Aussagen zur Stabilität und Genauigkeit des Verfahrens. Auch bei diesem Lösungsansatz ist die Modularität nicht gegeben. Es ist nicht möglich, z.B. den Lösungsalgorithmus eines Felds ohne größeren Aufwand auszutauschen oder einen vorhandenen „black-box“-Löser einzusetzen.

Bei den partitionierten Verfahren werden die einzelnen Felder separat voneinander modelliert und gelöst. Die Interaktion wird über den Austausch von Kopplungstermen in jedem Simulationsschritt sichergestellt. Je nach verwendetem Kopplungsverfahren ist dabei die Kontinuität der kinematischen und dynamischen Größen nicht unbedingt gegeben. Partitionierte Verfahren erlauben jedoch eine modulare Implementierung, bei der auch auf „black-box“-Löser zurückgegriffen werden kann.

Im Anschluss an die Darstellung des gekoppelten Problems der Fluid-Struktur-Interaktion im folgenden Abschnitt werden partitionierte Verfahren näher beschrieben und untereinander verglichen.

5.2 Das gekoppelte Problem

Für Aufgabenstellungen aus dem Bereich der Fluid-Struktur-Wechselwirkung besteht das gekoppelte Problem aus zwei physikalischen Feldern: dem Fluidfeld im Gebiet Ω_{F} und dem Strukturfeld im Gebiet Ω_{S}. Diese beiden Felder stehen über Kopplungsbedingungen am gemeinsamen Rand Γ in Interaktion. Im Folgenden werden zur vereinfachten Darstellung die in den Abschnitten 2.1.6 und 2.2.6 eingeführten nichtlinearen Operatoren $\mathcal{S}$ und $\mathcal{F}$ verwendet. Dabei wird von nun an die Bezeichnung des aktuellen Zeitschritts $(\bullet)_{n+1}$ weggelassen, sofern sie nicht für das Verständnis wichtig ist:

$$\mathbf{d}_\Gamma = \mathcal{S}(\mathbf{f}_\Gamma) \tag{5.1}$$

$$\mathbf{f}_\Gamma = \mathcal{F}(\mathbf{u}_\Gamma, \mathbf{d}_{\text{F}}). \tag{5.2}$$

Der Operator $\mathcal{S}$ fasst die Lösung des nichtlinearen Struktur-Problems zusammen und bestimmt aus gegebenen Kopplungskräften $\mathbf{f}_\Gamma$ die Verschiebungen $\mathbf{d}_\Gamma$ auf dem Kopplungsrand. Diese Verschiebungen und daraus resultierenden Geschwindigkeiten $\mathbf{u}_\Gamma$ auf dem Kopplungsrand sind die Eingangswerte für den Operator $\mathcal{F}$, der für die Lösung der Fluiddynamik steht und die Kräfte berechnet, die vom Fluid auf den Kopplungsrand wirken ($\mathbf{f}_\Gamma$). Dabei muss die neue Lage des Fluidgebiets ($\mathbf{d}_{\text{F}}$) berücksichtigt werden. Diese wird, wie im Abschnitt 2.3 beschrieben, aus den Verschiebungen am Kopplungsrand $\mathbf{d}_\Gamma$ mithilfe des linearen Operators $\mathcal{N}$ bestimmt:

$$\mathbf{d}_{\text{F}} = \mathcal{N}(\mathbf{d}_\Gamma). \tag{5.3}$$

Die Netzverschiebung kann als eigenständiges (algorithmisches) Feld aufgefasst werden, das am Kopplungsrand mit der Struktur oberflächengekoppelt und auf dem gesamten Gebiet mit dem Fluid volumengekoppelt ist. Es ist aber auch möglich, die Netzverschiebung als Teil des Fluidlösers anzusehen und mit diesem zu einem erweiterten Fluid-Operator $\hat{\mathcal{F}}$ zu kombinieren:

$$\mathbf{f}_\Gamma = \hat{\mathcal{F}}(\mathbf{u}_\Gamma, \mathbf{d}_\Gamma) = \mathcal{F}(\mathbf{u}_\Gamma, \mathcal{N}(\mathbf{d}_\Gamma)). \tag{5.4}$$

Bei der Einführung dieser Operatoren wurde bereits berücksichtigt, dass im Rahmen dieser Arbeit nur eine Dirichlet-Neumann-Partitionierung des gekoppelten Problems verwendet werden soll. Bei dieser „natürlichen“ Aufteilung wird auf dem Fluidgebiet ein Dirichlet-Problem mit vorgegebenen Verschiebungen und Geschwindigkeiten und auf dem Strukturgebiet ein Neumann-Problem mit vorgegebenen Kräften gelöst. Andere Aufteilungen der Randbedingungen, wie sie in Mok (2001) vorgestellt werden, sind auch möglich, bringen im Allgemeinen jedoch keine erheblichen Vorteile bei der Lösung des gekoppelten Problems. Eine Ausnahme bilden die Robin[1]-Randbedingungen, deren Verwendung im Abschnitt 5.6.4 kurz vorgestellt wird.

5.2.1 Kopplungsbedingungen

Die Kopplungsbedingungen auf dem gemeinsamen Rand Γ der beiden physikalischen Felder Fluid und Struktur sollen die geometrische Kompatibilität sowie den Erhalt von Masse, Impuls und Energie am Kopplungsrand gewährleisten (Wall 1999). Dies geschieht mithilfe der kinematischen und dynamischen Kontinuitätsbedingungen (Donea u. a. 2004).

Die kinematische Kontinuitätsbedingung gewährleistet, dass zwischen dem Fluidgebiet und der Struktur keine Klaffung oder Überlappung entsteht. Für viskose Strömungen wird in der Regel am Kopplungsrand eine Haft-Randbedingung („no-slip“) angenommen. Daraus folgt die Kontinuität der Verschiebungen und der Geschwindigkeiten auf dem Kopplungsrand zu allen Zeiten T:

$$\mathbf{d}_{\mathrm{S}} = \mathbf{d}_{\mathrm{F}} \qquad \text{und} \qquad \dot{\mathbf{d}}_{\mathrm{S}} = \mathbf{u} \qquad \text{auf } \Gamma \times T. \tag{5.5}$$

Wird dagegen am Kopplungsrand eine Gleit-Randbedingung („slip“) angenommen, wie z.B. bei reibungsfreien Fluiden, gilt die Kontinuitätsbedingung nur für die Normalkomponenten der Verschiebungen und Geschwindigkeiten.

Um Geschwindigkeiten als Dirichlet-Randbedingung für das Fluidgebiet aufbringen zu können, müssen die Verschiebungen der Struktur in Geschwindigkeiten umgerechnet werden. Die Fluidgeschwindigkeiten auf dem Interface können einerseits mit dem Euler-Rückwärts-Verfahren (5.6 a) bestimmt werden. Dieses Verfahren erfüllt die geometrischen Bilanzgleichungen nicht exakt und ist leicht dissipativ. Wird andererseits die Trapezregel (5.6 b) verwendet, werden die Bilanzgleichungen exakt erfüllt, es können aber

[1] Victor Gustave Robin, * 1855, † 1897, französischer Physiker und Mathematiker.

Oszillationen in den Fluidgeschwindigkeiten auftreten (FÖRSTER 2007).

$$\text{a)} \qquad \mathbf{u}_{\Gamma,n+1} = \frac{\mathbf{d}_{\Gamma,n+1} - \mathbf{d}_{\Gamma,n}}{\Delta t} \qquad \text{b)} \qquad \mathbf{u}_{\Gamma,n+1} = 2\frac{\mathbf{d}_{\Gamma,n+1} - \mathbf{d}_{\Gamma,n}}{\Delta t} - \mathbf{u}_{\Gamma,n} \tag{5.6}$$

Das dynamische Gleichgewicht am Kopplungsrand wird durch die dynamische Kontinuitätsbedingung hergestellt. Sie verlangt die Kontinuität des Spannungsvektors auf dem Kopplungsrand zu allen Zeiten T:

$$\mathbf{n} \cdot \boldsymbol{\sigma}_{\mathrm{S}} = \mathbf{n} \cdot \boldsymbol{\sigma}_{\mathrm{F}} \qquad \text{auf } \Gamma \times T. \tag{5.7}$$

Dies gilt für die wahren, physikalischen Spannungen $\boldsymbol{\sigma}_{\mathrm{S}}$ (Cauchy-Spannungen) und nicht für die auf Strukturseite eingeführten Zweiten Piola-Kirchhoff-Spannungen $\mathbf{S}$. Dabei gilt folgender Zusammenhang:

$$\boldsymbol{\sigma}_{\mathrm{S}} = (\det \mathbf{F})^{-1}\, \mathbf{F}\mathbf{S}\mathbf{F}^{\mathrm{T}}. \tag{5.8}$$

Die Kräfte, die am Kopplungsrand vom Fluid auf die Struktur wirken, resultieren aus dem Cauchy-Spannungstensor $\boldsymbol{\sigma}_{\mathrm{F}}$, der sich für Newton'sche Fluide nach Gleichung (2.32) aus einem viskosen und einem Druck-Anteil zusammensetzt. Da der viskose Anteil der Spannungen je nach Viskosität, Strömungsgeschwindigkeit und Geometrie nicht immer vernachlässigbar klein ist, werden in dieser Arbeit die Kopplungskräfte grundsätzlich aus den kompletten Cauchy-Spannungen berechnet.

5.2.2 Schreibweisen für das gekoppelte Fluid-Struktur-Problem

Werden nun die Kopplungsbedingungen mit den Operatoren für die drei Felder kombiniert, führt dies auf das gekoppelte Problem der Fluid-Struktur-Interaktion. Die drei (nichtlinearen) Gleichungen für die Netzbewegung ($\mathcal{N}$), das Fluid ($\mathcal{F}$) und die Struktur ($\mathcal{S}$) werden über die gemeinsamen Variablen, die Interface-Verschiebungen ($\mathbf{d}_\Gamma$) und -Geschwindigkeiten ($\mathbf{u}_\Gamma$), die Netzposition im Fluidgebiet ($\mathbf{d}_{\mathrm{F}}$) und die Kräfte am Kopplungsrand ($\mathbf{f}_\Gamma$) gekoppelt und bilden so ein gekoppeltes, nichtlineares Gleichungssystem:

$$\begin{pmatrix} \mathcal{N}(\ \mathbf{d}_\Gamma\ ) - \mathbf{d}_{\mathrm{F}} = \mathbf{0} \\ \mathcal{F}(\ \mathbf{u}_\Gamma\ ,\ \mathbf{d}_{\mathrm{F}}\ ) - \mathbf{f}_\Gamma = \mathbf{0} \\ -\ \mathbf{d}_\Gamma + \mathcal{S}(\ \mathbf{f}_\Gamma\ ) = \mathbf{0} \end{pmatrix}. \tag{5.9}$$

Durch Auflösen der ersten Gleichung nach den Netzpositionen $\mathbf{d}_{\mathrm{F}}$ und der zweiten Gleichung nach den Kopplungskräften $\mathbf{f}_\Gamma$, können diese beiden Variablen aus dem Glei-

chungssystem eliminiert werden. Die dritte Gleichung führt auf eine Iterationsvorschrift für die Interface-Verschiebungen $\mathbf{d}_\Gamma$:

$$\begin{aligned}\mathbf{d}_\Gamma^{(k+1)} &= \mathcal{S}\Big(\mathcal{F}\big(\mathbf{u}_\Gamma^{(k)}, \mathcal{N}(\mathbf{d}_\Gamma^{(k)})\big)\Big) \\ &= \mathcal{T}(\mathbf{d}_\Gamma^{(k)}).\end{aligned} \tag{5.10}$$

Diese nichtlineare Kopplungs-Abbildung $\mathcal{T}(\mathbf{d}_\Gamma)$ der Interface-Verschiebungen beschreibt eine instationäre, nichtlineare Iteration erster Ordnung, deren Fixpunkt $\mathbf{d}_\Gamma^*$ die Lösung des gekoppelten Problems darstellt:

$$\mathbf{d}_\Gamma^* = \mathcal{T}(\mathbf{d}_\Gamma^*). \tag{5.11}$$

Ausgehend von der Fixpunktgleichung kann das gekoppelte Problem auch als Nullstellensuche formuliert werden. Hierzu wird zunächst die Gleichung (5.11) umgestellt und damit der Interface-Kompatibilitäts-Operator $\mathcal{L}(\mathbf{d}_\Gamma)$ definiert:

$$\mathcal{L}(\mathbf{d}_\Gamma^*) = \mathbf{d}_\Gamma^* - \mathcal{T}(\mathbf{d}_\Gamma^*) = 0 \tag{5.12}$$

Die Nullstelle $\mathbf{d}_\Gamma^*$ dieses Operators $\mathcal{L}(\mathbf{d}_\Gamma)$ entspricht wiederum der Lösung des gekoppelten Fluid-Struktur-Interaktions-Problems.

Auf der Basis dieser unterschiedlichen Schreibweisen des gekoppelten Problems lassen sich verschiedene Lösungsansätze herleiten. Diese werden in den folgenden Abschnitten vorgestellt und zum Teil näher untersucht.

5.3 Überblick über Lösungsverfahren für die Fluid-Struktur-Kopplung

In diesem Abschnitt wird zunächst ein Überblick über zwei Klassen von Lösungsverfahren für gekoppelte Probleme gegeben, bevor in den folgenden Abschnitten 5.5 und 5.6 die in dieser Arbeit eingesetzten partitionierten Lösungsmethoden näher betrachtet werden. In der Literatur findet sich eine gute Übersicht über Lösungsmethoden für Fluid-Struktur-Wechselwirkungs-Probleme zum Beispiel in den Sonderausgaben der Zeitschrift „Computer Methods in Applied Mechanics and Engineering“ zum Thema Fluid-Struktur-Interaktion (Ohayon und Felippa 2001, Ohayon und Kvamsdal 2006 und Papadrakakis und Allix 2008), in Tagungsbänden entsprechender Konferenzen und Workshops (Bungartz und Schäfer (2006)) oder in einer Vielzahl von Dissertationen zu diesem Thema (Mok 2001).

5.3.1 Monolithische Lösungsverfahren

Monolithische Lösungsverfahren für gekoppelte Probleme bieten sich besonders bei volumengekoppelten Problemstellungen an, z.B. bei einem gekoppelten thermo-hygro-mechanischen Schädigungsmodell für Beton (Grasberger u. a. 2003). Hierbei ist die simultane Diskretisierung und Lösung der Gleichungen aller beteiligten Felder ein naheliegender Ansatz. Aber auch bei oberflächengekoppelten Problemen, wie z.B. der Fluid-Struktur-Wechselwirkung, kommen monolithische Verfahren zum Einsatz (Hübner und Dinkler 2005). Rugonyi und Bathe (2000) haben monolithische Verfahren eingesetzt, um Konvergenzprobleme der gestaffelten Verfahren zu umgehen.

So bietet die Formulierung eines gekoppelten Gleichungssystems die Möglichkeit, Stabilitäts- und Fehleranalysen am gekoppelten Problem durchzuführen (Walhorn u. a. 2003). Die Lösung dieses sehr großen Systems von gekoppelten nichtlinearen algebraischen Gleichungen kann iterativ mit dem Newton-Raphson-Verfahren erfolgen und erreicht in der Nähe der Lösung eine quadratische Konvergenz. Die zur iterativen Lösung linearisierten Gleichungssysteme sind aufgrund der unterschiedlichen physikalischen Variablen häufig sehr schlecht konditioniert und erfordern spezielle Vorkonditionierer (Heil 2004). Außerdem verlangt die simultane Diskretisierung einen einheitlichen Zeitschritt in allen Feldern. Bei stark unterschiedlichen Zeitskalen in den einzelnen physikalischen Feldern kann dies zu einem erheblichen Mehraufwand führen (Hübner u. a. 2004).

Aus dem Blickwinkel der Implementierung erfordert ein monolithisches Verfahren immer einen hochgradig spezialisierten Code, bei dem z.B. bereits vorhandene Löser für die Einzelfelder nicht verwendet werden können. Ein solcher hoch komplexer Code ist auf ein bestimmtes gekoppeltes Problem spezialisiert und ist nur sehr schwer zu pflegen oder (auf andere gekoppelte Probleme) zu erweitern (Cervera u. a. 1996).

Eine Variante der monolithischen Lösungsverfahren stellt die Formulierung aller beteiligten Felder mit einem einheitlichen Satz von Differenzialgleichungen dar (Greenshields und Weller 2005). Eine Unterscheidung zwischen den physikalischen Feldern erfolgt nur über die verwendeten Materialgesetze und -parameter. Diese Herangehensweise ermöglicht dann auch eine einheitliche Diskretisierung und Lösung des gesamten Berechnungsgebiets, ist aber auf eine spezielle Klasse von gekoppelten Problemen beschränkt, bei denen alle physikalischen Felder mit denselben Gleichungen beschrieben werden können.

5.3.2 Partitionierte Lösungsverfahren

Im Gegensatz zur simultanen Diskretisierung und Lösung bei den monolithischen Lösungsverfahren werden bei den partitionierten Verfahren die beteiligten Felder jeweils

separat diskretisiert. Man erhält so, wie im Kapitel 2 eingeführt, für jedes Feld einen eigenen Lösungsalgorithmus. Bei der Wahl der Diskretisierungstechniken können dabei die spezifischen Eigenschaften des jeweiligen Felds berücksichtigt werden oder aber auch bereits vorhandene, spezialisierte Löser für ein Teilproblem verwendet werden. Durch den modularen Aufbau der partitionierten Verfahren ist es auch ohne großen Aufwand möglich, den Löser eines Felds gegen einen anderen für dieselbe Physik oder auch einen Löser für eine andere Aufgabenstellung auszutauschen.

Über die Kopplungsbedingungen werden die einzelnen Felder je nach Algorithmus mehr oder weniger eng miteinander verbunden. Man unterscheidet dabei grundsätzlich zwischen zwei Verfahrensklassen:

- **Einfach gestaffelte Verfahren** (Abschnitt 5.5)
  explizite oder schwach koppelnde Verfahren
  („staggered schemes“, „explicit coupling“, „loose coupling schemes“)
- **Iterativ gestaffelte Verfahren** (Abschnitt 5.6)
  implizite oder stark koppelnde Verfahren
  („iterative staggered schemes“, „implicit coupling“, „strong coupling schemes“)

Diese Bezeichnungen werden in der Literatur weitgehend einheitlich verwendet. Dagegen bezeichnen z.B. Zhang und Hisada (2004) mit „strong coupling methods“ monolithische Verfahren und mit „weak coupling methods“ partitionierte Methoden. Trotz dieser etwas eigenwilligen Bezeichnungen bietet diese Veröffentlichung eine gute Kategorisierung von partitionierten Verfahren. Ausgehend von einer monolithischen Formulierung werden über Abschätzung und Vernachlässigung einzelner Kopplungsterme verschiedene partitionierte Verfahren abgeleitet.

Eine vergleichende Darstellung der gebräuchlichsten partitionierten Lösungsverfahren, wie der Block-Gauß-Seidel-Iteration und den Newton-Krylow-Methoden, findet sich bei Deparis u. a. (2006). Zusätzlich stellen die Autoren ein weiteres partitioniertes Verfahren vor, das auf der Basis der „domain decomposition“-Theorie beruht und in einer vorkonditionierten Richardson[2]-Iteration auf den Freiheitsgraden des Fluid-Struktur-Interfaces resultiert.

Einen ausführlichen Überblick über partitionierte Lösungsverfahren und deren historische Entwicklung bietet außerdem die Dissertation von Mok (2001).

[2]Lewis Fry Richardson, * 11. Oktober 1881 in Newcastle upon Tyne, † 30. September 1953 in Kilmun, britischer Mathematiker und Friedensforscher.

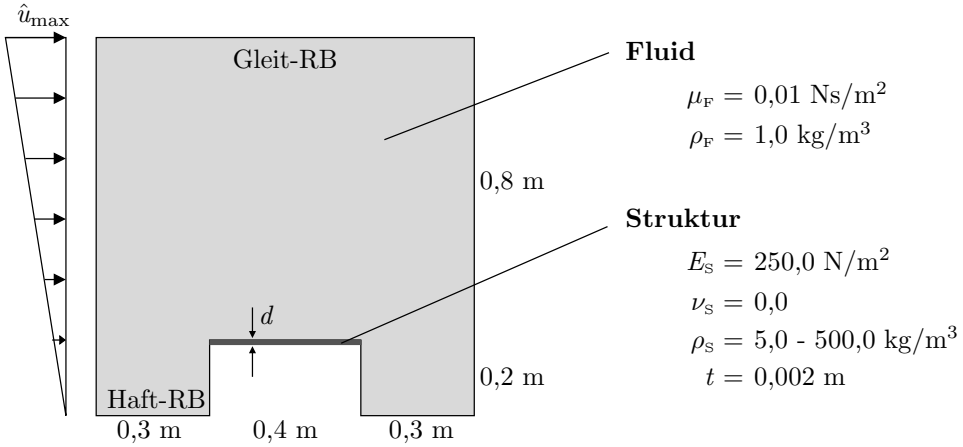


Abbildung 5.1: Geometrie und Materialparameter des 2-D-Membrandachs.

5.4 Demonstrationsbeispiel

Einige der in den folgenden Abschnitten beschriebenen Lösungsverfahren für gekoppelte Probleme werden in ihren Eigenschaften anhand einer beispielhaften Fluid-Struktur-Wechselwirkungs-Simulation verglichen. Gewählt wird dazu eine vereinfachte, zweidimensionale Simulation eines horizontalen Membrandachs, wie in Abbildung 5.1 dargestellt.

Für die Struktur werden drei verschiedene Massendichten ρ_S von 5,0 kg/m^3, 50,0 kg/m^3 und 500,0 kg/m^3 verwendet, um unterschiedlich starke Kopplungen zwischen Fluid und Struktur zu untersuchen. Die maximale Einströmgeschwindigkeit $\hat{u}_{\max}$ wird linear von 0,0 m/s auf 0,6 m/s innerhalb von 2 s gesteigert. In der Regel werden die ersten 20 Zeitschritte der Simulation betrachtet, wobei die Größe eines Zeitschritts 0,1 s beträgt.

Die Gebiete für Fluid und Struktur wurden, wie in Tabelle 5.1 angegeben, mit insgesamt 8.256 Elementen vernetzt. Aus dieser groben Diskretisierung wird, wie im Abschnitt 3.2.4 beschrieben, durch gleichmäßige Unterteilung in beide Raumrichtungen mit dem Faktor $subdiv = 2$ ein feines Netz generiert. In der Regel wird die Lösung mit der feinen Diskretisierung berechnet; für die in Abschnitt 5.6.2 beschriebenen Zwei-Level-Verfahren wird zusätzlich das grobe Gitter verwendet.

5.5 Einfach gestaffelte Lösungsverfahren

Der Begriff gestaffelte Lösungsverfahren („staggered schemes“) geht zurück auf die Arbeit von Felippa u. a. (1977), in der unter Verwendung zweier separater Program-

	grobes Netz	$\Longrightarrow$	feines Netz
Knoten	8.573		33.005
Elemente	8.256	$subdiv = 2$	32.384
Freiheitsgrade	22.547		88.839

Tabelle 5.1: Diskretisierungen des Demonstrationsbeispiels.

me für Fluid und Struktur die numerische Lösung des gekoppelten elasto-akustischen Problems eines durch Schockwellen belasteten U-Boots vorgestellt wird. Dieser Ansatz wurde von Felippa und Park (1980) auf allgemeine gekoppelte Problemstellungen erweitert. Einen Überblick über diese Klasse von Lösungsverfahren bietet der Artikel Felippa u. a. (1998).

Ab 1990 wurde die Weiterentwicklung der einfach gestaffelten Verfahren maßgeblich durch die Arbeiten im Bereich der Aeroelastik in der Gruppe um C. Farhat geprägt. Farhat u. a. (1995) und Piperno u. a. (1995) vergleichen eine Reihe von sequenziell gestaffelten Verfahren im Hinblick auf Stabilität, Genauigkeit und Parallelisierbarkeit. Mit der Verbesserung der Verfahren durch einen Strukturprädiktor (Piperno 1997) kann Masse, Energie und Impuls annähernd exakt erhalten werden. Durch die Verwendung von asynchronen Verfahren (Farhat und Lesoinne 2000) können kontinuierliche Verschiebungen und Geschwindigkeiten am Interface erreicht werden und gleichzeitig die geometrischen Bilanzgleichungen (GCL) erfüllt werden. Mit speziell abgestimmten Zeitintegrationsverfahren für Fluid, Struktur und Netzverschiebung und einem Strukturprädiktor zweiter Ordnung können einfach gestaffelte Lösungsverfahren konstruiert werden, die eine Genauigkeit zweiter Ordnung in der Zeit erreichen (Farhat u. a. 2006).

Typisch für die einfach gestaffelten Verfahren ist, dass es innerhalb eines Zeitschritts keine Iteration über die Teilgebiete gibt, d.h. jedes Feld wird pro Zeitschritt nur einmal gelöst und die Kopplungsvariablen werden nur einmal pro Richtung ausgetauscht.

Im Folgenden werden zunächst die beiden Grundverfahren der einfach gestaffelten Lösungsverfahren vorgestellt und im Anschluss kurz einige Variationsmöglichkeiten beschrieben.

5.5.1 Parallel gestaffeltes Lösungsverfahren

Werden beide Kopplungsinformationen, d.h. die Kopplungskräfte vom Fluid an die Struktur und die Verschiebungen des Kopplungsrands von der Struktur an das Fluid,

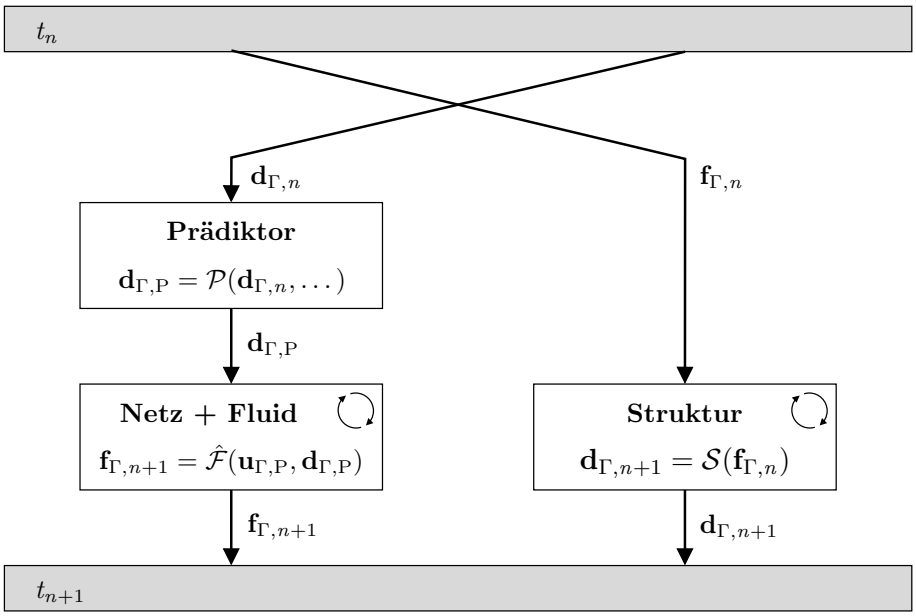


Abbildung 5.2: Ablaufdiagramm zum parallel gestaffelten Lösungsverfahren mit Prädiktor.

am Anfang des Zeitschritts übergeben, können beide Felder parallel zueinander gelöst werden.

Dieses Verfahren ist in Abbildung 5.2 als Ablaufdiagramm dargestellt. Es wird vorausgesetzt, dass die Zustandsgrößen des Fluids und der Struktur zum Zeitpunkt t_n bekannt sind. Die Verschiebungen des Kopplungsrands zu diesem Zeitpunkt $\mathbf{d}_{\Gamma,n}$ werden von der Strukturpartition an das Fluid übertragen. Dabei kann, wie in Abbildung 5.2 dargestellt, mit einem Prädiktor $\mathbf{d}_{\Gamma,\mathrm{P}}$ (siehe Abschnitt 5.5.3) die Lage des Kopplungsrands zum Zeitpunkt t_{n+1} abgeschätzt werden. Der erweiterte Fluid-Löser $\hat{\mathcal{F}}$ berechnet daraus zunächst die neue Position des Netzes und die Geschwindigkeiten am Kopplungsrand. Mit diesen Werten werden die Fluid-Gleichungen gelöst und die Kräfte auf den Kopplungsrand zum Zeitpunkt t_{n+1} berechnet. Gleichzeitig werden vom Fluid die Kopplungskräfte des alten Zeitschritts $\mathbf{f}_{\Gamma,n}$ an die Struktur übergeben. Der Struktur-Löser $\mathcal{S}$ berechnet aus dieser Neumann-Randbedingung die neue Lage der Struktur zum Zeitpunkt t_{n+1}. Die kreisförmigen Pfeile in den Kästen für Fluid- und Struktur-Löser im Ablaufdiagramm sollen veranschaulichen, dass an dieser Stelle die Lösung eines nichtlinearen Problems erforderlich ist, die ein Iterationsverfahren (z.B. Newton-Raphson) erfordert.

Da Fluid- und Struktur-Löser innerhalb des Zeitschritts unabhängig voneinander sind, können sie auch parallel auf verschiedenen Prozessoren ohne jegliche Kommunikation arbeiten. Es ist sogar denkbar, dass Fluid- und Strukturcode auf unterschiedlichen Computern laufen, deren Hardware-Architektur optimal zum jeweiligen Löser passt. Die

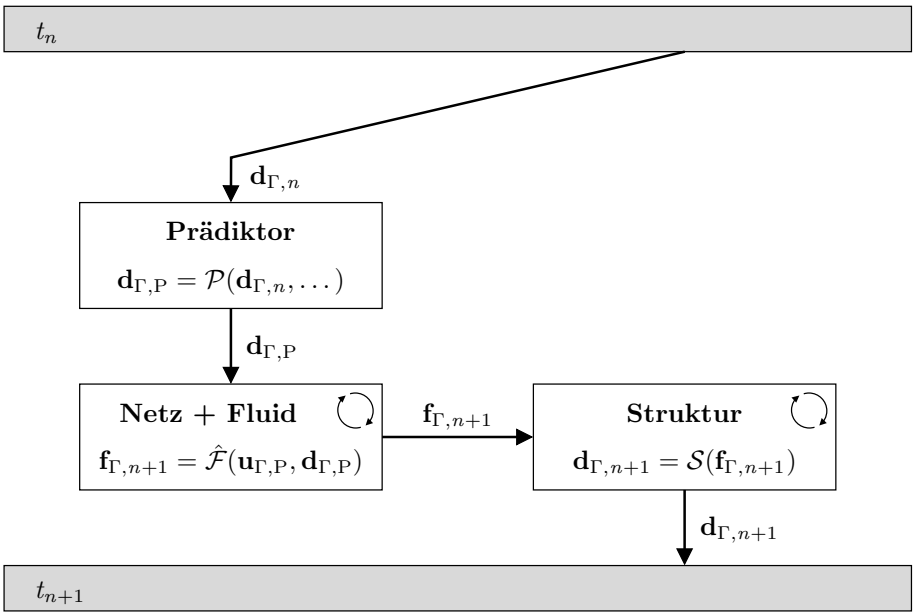


Abbildung 5.3: Ablaufdiagramm zum sequenziell gestaffelten Lösungsverfahren mit Prädiktor.

Kopplungsinformationen werden zu Beginn jedes Zeitschritts über ein entsprechendes Netzwerk ausgetauscht.

5.5.2 Sequenziell gestaffeltes Lösungsverfahren

Bei den sequenziell gestaffelten Lösungsverfahren werden die Kopplungsinformationen nicht gleichzeitig ausgetauscht. Daher können die beiden Felder auch nicht parallel zueinander gelöst werden.

Ausgehend vom bekannten Zustand von Fluid und Struktur zum Zeitpunkt t_n werden die Verschiebungen des Kopplungsrands an den Fluid-Löser übergeben. Dabei kann, wie schon beim parallel gestaffelten Verfahren, ein Prädiktorschritt zum Einsatz kommen. Der erweiterte Fluid-Löser berechnet daraus die neue Position des Netzes und bestimmt iterativ die Lösung der nichtlinearen Fluidgleichungen zum Zeitpunkt t_{n+1}. Danach werden die Kräfte des Fluids auf den Kopplungsrand $\mathbf{f}_{\Gamma,n+1}$ an die Struktur übergeben (Abbildung 5.3). Der Struktur-Löser $\mathcal{S}$ berechnet aus dieser Neumann-Randbedingung die Lage der Struktur und damit auch des Kopplungsrands zum Zeitpunkt t_{n+1}. Da bei der Lösung der Struktur bereits die aktuellen Werte der Kopplungskräfte $\mathbf{f}_{\Gamma,n+1}$ verwendet werden, liefern die sequenziellen Algorithmen in der Regel genauere Ergebnisse als die parallel gestaffelten Verfahren.

Bei sequenziell gestaffelten Verfahren wird die dynamische Kontinuität zum Zeitpunkt t_{n+1} und damit auch die Impulserhaltung exakt erfüllt. In der Regel stimmt aber die Lage des Kopplungsrands $\mathbf{d}_{\Gamma,n+1}$, die vom Struktur-Löser berechnet und als Lösung des Zeitpunkts t_{n+1} angenommen wird, nicht mit der im Fluid-Löser angenommenen Position $\mathbf{d}_{\Gamma,\mathrm{P}}$ überein. Diese Verletzung der kinematischen Kontinuität und damit auch des Masse- und Energieerhalts kann zwar durch einen entsprechend genauen Prädiktor minimiert werden, lässt sich aber ohne eine Iteration über die gekoppelten Felder (vgl. Abschnitt 5.6) nicht vermeiden.

Variiert werden können die einfach gestaffelten Verfahren durch die Anordnung der Zeitintervalle der beiden Felder. Bei den bisher beschriebenen synchronen Verfahren beginnen die Zeitintervalle für Fluid und Struktur beide zum gleichen Zeitpunkt t_n. Ein Zeitversatz zwischen den beiden Zeitintegrationsverfahren führt auf sogenannte asynchrone Verfahren, wie sie von Farhat und Lesoinne (2000) vorgeschlagen werden. Auch wenn mit diesen Varianten kontinuierliche Verschiebungen und Geschwindigkeiten am Kopplungsrand erreicht werden können, so zeigen sie jedoch in Verbindung mit inkompressiblen Strömungen ein schlechtes Stabilitätsverhalten (Mok 2001).

5.5.3 Strukturprädiktor

Wird die Position des Kopplungsrands zum Zeitpunkt t_n direkt an den Fluid-Löser übergeben, so entspricht dies einem Prädiktor nullter Ordnung:

$$\mathbf{d}_{\Gamma,\mathrm{P}} = \mathcal{P}^0(\mathbf{d}_{\Gamma,n}, \dots) = \mathbf{d}_{\Gamma,n}. \tag{5.13}$$

Bei sequenziell gestaffelten Verfahren wird jedoch die Genauigkeit des Gesamtverfahrens durch die Ordnung der niedrigsten Komponente bestimmt. Daher ist es erstrebenswert einen Prädiktor höherer Ordnung zu verwenden. Piperno (1997) schlägt Prädiktoren zur Extrapolation der Verschiebungen am Kopplungsrand vor, die erster bzw. zweiter Ordnung genau sind:

$$\begin{aligned} \mathcal{P}^1(\mathbf{d}_{\Gamma,n}, \dots) &= \mathbf{d}_{\Gamma,n} + \Delta t\, \dot{\mathbf{d}}_{\Gamma,n} \\ \mathcal{P}^2(\mathbf{d}_{\Gamma,n}, \dots) &= \mathbf{d}_{\Gamma,n} + \Delta t \left(\frac{3}{2} \dot{\mathbf{d}}_{\Gamma,n} - \frac{1}{2} \dot{\mathbf{d}}_{\Gamma,n-1} \right). \end{aligned} \tag{5.14}$$

Die Wahl des Prädiktors bei sequenziell gestaffelten Verfahren beeinflusst nicht nur die Genauigkeit des Verfahrens in der Zeit, sondern auch den Beginn von Instabilitäten aufgrund des sogenannten „artificial added mass“-Effekts.

5.5.4 „Artificial Added Mass“-Effekt

Sequenziell gestaffelte Verfahren weisen eine Instabilität auf, die u.a. von Wall u. a. (1999) und Le Tallec und Mouro (2001) beschrieben wurde und „artificial added mass“-Effekt genannt wird. Die „artificial added mass“ ist ein künstlicher Anteil an der „added mass“, einer zusätzlichen Kopplungslast, die aus der Trägheit des Fluids resultiert und wie eine zusätzliche Fluidmasse auf die Beschleunigungen der Struktur wirkt. Dieser künstliche Anteil folgt aus der Verletzung der kinematischen Kontinuität und resultiert bei inkompressiblen Strömungen in einer fehlerhaften Belastung, die ohne Zeitverzögerung oder Abschwächung auf die Struktur trifft.

Causin u. a. (2005) haben gezeigt, dass der Beginn dieser Instabilität für ein reduziertes Modell auf der Basis der Druck-Poisson-Gleichung vorhergesagt werden kann. Eine ausführliche Untersuchung des „artificial added mass“-Effekts auf der Basis der inkompressiblen Navier-Stokes-Gleichungen findet sich in Förster u. a. (2007). Dort werden auf der Basis eines „added mass“-Operators für stabilisierte Finite Elemente Grenzen für die Instabilität in Abhängigkeit des Zeitintegrationsverfahrens des Fluids und des Strukturprädiktors angegeben. Es wird gezeigt, dass die „artificial added mass“-Instabilität von folgenden Parametern beeinflusst wird:

Massendichtenverhältnis von Struktur und Fluid: Nur für Problemklassen, bei denen die Massendichte der Struktur deutlich größer ist als die Massendichte des Fluids, sind stabile Berechnungen möglich.

Strukturprädiktor: Bei zunehmender Genauigkeit des Strukturprädiktors verschiebt sich der Beginn der Instabilität deutlich nach vorne.

Zeitintegrationsverfahren im Fluid: Auch das Zeitintegrationsverfahren im Fluid hat einen Einfluss auf die Instabilität. So ergibt sich z.B. mit dem BDF2-Verfahren eine doppelt so strenge Instabilitäts-Bedingung wie mit dem Euler-Rückwärts-Verfahren.

Zeitschrittgröße: Für stabilisierte Fluid-Formulierungen setzt die Instabilität bei kleiner werdenden Zeitschritten immer früher ein.

Mithilfe von Fourier[3]-Fehler-Analysen einer instationären, inkompressiblen Strömung durch ein flexibles Rohr haben Degroote u. a. (2008) den „artificial added mass“-Effekt untersucht. Auch sie kommen zu dem Ergebnis, dass der Zeitschritt und die Steifigkeit der Struktur einen erheblichen Einfluss auf die Anzahl der benötigten Iterationen der verwendeten iterativ gestaffelten Verfahren und damit auch auf die Instabilität sequenziell gestaffelter Verfahren hat. Sie stellen außerdem fest, dass besonders die räumlich

[3]Jean Baptiste Joseph Fourier, * 21. März 1768 bei Auxerre, † 16. Mai 1830 in Paris, französischer Mathematiker und Physiker.

langwelligen Lösungsanteile einen destabilisierenden Einfluss auf die Iterationsverfahren haben.

Um sequenziell gestaffelte Verfahren auch bei Fluid-Struktur-Interaktions-Simulationen mit sehr leichten Strukturen einsetzen zu können, wird von Tezduyar (2004) für die Berechnung von umströmten Fallschirmen eine „augmented mass“-Stabilisierung vorgeschlagen. Diese soll durch eine künstliche Erhöhung der Strukturmasse den „artificial added mass“-Effekt vermeiden. Andere Vorschläge zur Umgehung der Instabilität werden von Mok (2001) gemacht. Dort werden eine künstliche viskose Dämpfung, das Nullsetzen der Geschwindigkeiten an den Kopplungsknoten oder eine fiktive Fluid-Kompressibilität als Gegenmaßnahmen erwähnt. Da diese Lösungen stark problemabhängig sind und zudem noch die Physik der Aufgabenstellung verändern, werden sie hier nicht weiter verfolgt. In dieser Arbeit wird durch eine starke Kopplung die kinematische Kontinuität am Interface gewährleistet. Dies ist durch eine Iteration über die gekoppelten Felder möglich.

5.6 Iterativ gestaffelte Lösungsverfahren

Bereits Park und Felippa (1983) haben eine Iteration über die partitionierten Felder als Lösung für die bei einfach gestaffelten Verfahren auftretende Instabilität in Betracht gezogen. Sie stufen diese iterativen Verfahren aber generell als numerisch zu aufwändig ein (Felippa u. a. 1998; Park und Felippa 1983). Dies liegt vor allem an den sehr schlechten Konvergenzeigenschaften der von ihnen eingesetzten einfachen iterativen Verfahren. Andere Forschergruppen dagegen nehmen den höheren Aufwand der iterativen Verfahren in Kauf, um stabile und zuverlässige Ergebnisse zu erhalten. So verwenden Kalro und Tezduyar (2000) ein iterativ gestaffeltes Verfahren ohne Relaxation für die Simulation von umströmten Fallschirmen, da das zuvor eingesetzte sequenziell gestaffelte Verfahren nur durch systemverändernde Maßnahmen (starke künstliche viskose Dämpfung) stabilisiert werden konnte (Stein u. a. 1998, 2000).

Die Grundidee der iterativ gestaffelten Lösungsverfahren ist die implizite Behandlung der Kopplungsvariablen. Während der Iteration über die gekoppelten Felder werden die Kopplungsinformationen so lange verbessert, bis das Residuum der Kopplungsbedingungen unterhalb einer vorgegebenen Schranke liegt. Durch die im Rahmen der vorgegebenen Genauigkeit exakte Erfüllung der kinematischen und dynamischen Kopplungsbedingung kann eine starke algorithmische Kopplung erreicht werden. Der Erhalt von Masse, Impuls und Energie am Kopplungsrand ist bei diesen Verfahren, die gegen das Ergebnis einer monolithischen Lösung des gekoppelten Problems konvergieren, gewährleistet.

In den vergangenen Jahren haben sich die iterativ gestaffelten Verfahren für die Fluid-Struktur-Wechselwirkung von inkompressiblen Strömungen und leichten Strukturen als Standard gegen die einfach gestaffelten Methoden durchgesetzt. Die große Anzahl an iterativen Verfahren kann grundsätzlich in zwei Gruppen eingeteilt werden:

- **Block-Iterations-Verfahren**
  Block-Gauß-Seidel-Verfahren, Fixpunkt-Iterationsverfahren, Richardson-Iteration, iteratives Dirichlet-Neumann-Substruktur-Verfahren
- **Newton-Krylow-Verfahren**
  Newton-Verfahren, Quasi-Newton-Verfahren

Im Folgenden werden diese beiden großen Verfahrensgruppen zusammen mit verwandten Methoden und Varianten vorgestellt.

5.6.1 Block-Iterations-Verfahren

Durch die Anwendung verschiedener linearer Iterationsverfahren (Abschnitt 4.3) auf das gekoppelte System nichtlinearer Gleichungen (5.9) lässt sich eine Klasse von Iterationsverfahren zur Lösung von gekoppelten Problemen ableiten. Diese Verfahren können nach den zugrunde liegenden linearen Iterationsverfahren benannt werden, wobei bei der Namensgebung berücksichtigt wird, dass es sich um Block-Varianten der klassischen Iterationsverfahren handelt.

Block-Gauß-Seidel-Verfahren

Die Herleitung des Block-Gauß-Seidel-Verfahrens geschieht auf Basis der Darstellung des gekoppelten Systems als nichtlineares Gleichungssystem (5.9). Diese Darstellung wird hier wiederholt:

$$\begin{pmatrix} \mathcal{N}(\ \mathbf{d}_\Gamma\ ) - \mathbf{d}_\mathrm{F} = \mathbf{0} \\ \mathcal{F}(\ \mathbf{u}_\Gamma\ ,\ \mathbf{d}_\mathrm{F}\ ) - \mathbf{f}_\Gamma = \mathbf{0} \\ -\ \mathbf{d}_\Gamma + \mathcal{S}(\ \mathbf{f}_\Gamma\ ) = \mathbf{0} \end{pmatrix}. \tag{5.15}$$

Auf dieses nichtlineare Gleichungssystem kann nun die Block-Variante des Gauß-Seidel-Iterations-Verfahrens, wie in Abschnitt 4.3.2 beschrieben, angewendet werden (Vrahatis u. a. 2003). Jede Zeile des Gleichungssystems (5.15), d.h. jedes Feld, bildet dabei einen Block und der Vektor der Unbekannten setzt sich aus den drei korrespondierenden Vektoren der Unbekannten zusammen: $(\mathbf{d}_\mathrm{F}, \mathbf{f}_\Gamma, \mathbf{d}_\Gamma)^\mathrm{T}$. In jeder Iteration werden die drei Zeilen nacheinander jeweils einzeln gelöst, wobei bereits bekannte Ergebnisse aus dieser

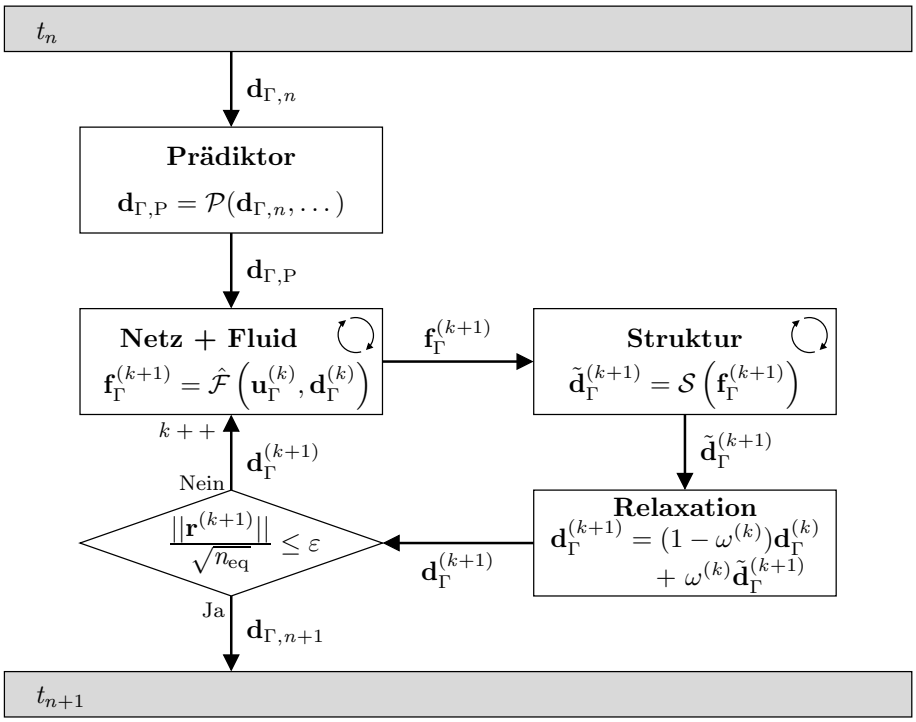


Abbildung 5.4: Ablaufdiagramm zum Block-Gauß-Seidel-Verfahren mit Prädiktor und Relaxation.

Iteration verwendet werden. Auf das nichtlineare Gleichungssystem (5.15) angewendet bedeutet dies, dass in jeder Iteration zunächst die Netzverschiebung $\mathbf{d}_{\mathrm{F}}^{(k+1)}$ berechnet wird. Als Eingabewert dient die Lage des Kopplungsrands aus der letzten Iteration $\mathbf{d}_\Gamma^{(k)}$, bzw. bei der ersten Iteration ein Prädiktorwert $\mathbf{d}_{\Gamma,\mathrm{P}}$. Dieser Prädiktor kann analog zu den einfach gestaffelten Verfahren (siehe Abschnitt 5.5.3) entweder der konvergierten Lösung des letzten Zeitschritts entsprechen oder höherer Ordnung sein. Mit der neuen Lage des Fluidnetzes werden der Fluid-Operator gelöst und die Kräfte auf den Kopplungsrand $\mathbf{f}_\Gamma^{(k+1)}$ berechnet. In Abbildung 5.4 sind die Berechnung der Netzverschiebung und das Lösen des Fluid-Problems im erweiterten Fluid-Operator $\hat{\mathcal{F}}$ zusammengefasst. Als drittes wird der Struktur-Operator gelöst und die Lage des Kopplungsrands $\tilde{\mathbf{d}}_\Gamma^{(k+1)}$ aufgrund der aktuellen Kopplungskräfte $\mathbf{f}_\Gamma^{(k+1)}$ bestimmt.

Wie bei den linearen Iterationsverfahren kann auch für das Block-Gauß-Seidel-Verfahren eine relaxierte Variante gebildet werden. Diese ist in der englischsprachigen Literatur auch unter dem Namen Block-SOR-Verfahren bekannt. Bei der Relaxation werden nur die Verschiebungen des Kopplungsrands $\mathbf{d}_\Gamma^{(k+1)}$ berücksichtigt und im letzten Schritt jeder Iteration die neue Lage des Kopplungsrands mit einer Linearkombination aus der alten Lage $\mathbf{d}_\Gamma^{(k)}$ und dem Ergebnis der Iterationsvorschrift $\tilde{\mathbf{d}}_\Gamma^{(k+1)}$ gebildet. Eine ausführlichere Beschreibung zur Bestimmung des Relaxationsparameters $\omega^{(k)}$ ist ab Seite 122 zu finden.

Die Konvergenz wird beim Block-Gauß-Seidel-Verfahren nur anhand der Verschiebungen des Kopplungsrands $\mathbf{d}_\Gamma^{(k+1)}$ überprüft und nicht mit dem kompletten Vektor der Unbekannten wie beim gewöhnlichen Gauß-Seidel-Verfahren. Dabei wird das aktuelle Residuum $\mathbf{r}^{(k+1)}$ als iterative Verbesserung der Interface-Verschiebungen bestimmt:

$$\mathbf{r}^{(k+1)} = \mathbf{d}_\Gamma^{(k+1)} - \mathbf{d}_\Gamma^{(k)}. \tag{5.16}$$

Als Abbruchkriterium kann entweder das Verhältnis der euklidischen Norm des aktuellen Residuums zur Norm des Residuenvektors der Startnäherung $\mathbf{r}^{(0)}$ (5.17 a) gewählt werden oder die mit der Anzahl der Freiheitsgrade gewichtete Norm des aktuellen Residuums (5.17 b):

$$\text{a)} \quad \frac{||\mathbf{r}^{(k+1)}||}{||\mathbf{r}^{(0)}||} \leq \varepsilon \qquad\qquad \text{b)} \quad \frac{||\mathbf{r}^{(k+1)}||}{\sqrt{n_{\text{eq}}}} \leq \varepsilon. \tag{5.17}$$

Das erste Kriterium hängt von der Startnäherung $\mathbf{d}_\Gamma^{(0)}$ ab und kann daher einerseits zu unnötig vielen Iterationen führen, wenn die Startnäherung relativ gut ist. Andererseits wird die Iteration zu früh abgebrochen, wenn die Startnäherung sehr schlecht und dadurch das zugehörige Residuum $\mathbf{r}^{(0)}$ sehr groß ist. Im Rahmen dieser Arbeit wird daher das zweite Kriterium angewendet, das die mit der Anzahl der Freiheitsgrade n_{eq} gewichtete euklidische Norm des aktuellen Residuenvektors abprüft.

Zu den ersten Arbeiten, in denen das Block-Gauß-Seidel-Verfahren unter dieser Bezeichnung zur Lösung von gekoppelten Problemen erwähnt wird, gehören Bungartz und Schulte (1995). Sie lösen zunächst mikro-elektromechanische Systeme und später auch Fluid-Struktur-Wechselwirkungs-Probleme (Bungartz u. a. 1998) mit Block-Jacobi-, Block-Gauß-Seidel- und Block-SOR-Verfahren. Cervera u. a. (1996) wenden das Block-Jacobi- und Block-Gauß-Seidel-Verfahren auf Fluid-Struktur-Interaktion und thermo-mechanische Kopplung an.

Derselbe Kopplungsalgorithmus wie beim Block-Gauß-Seidel-Verfahren ergibt sich aus der Fixpunkt-Formulierung (Deparis u. a. 2006) des gekoppelten Problems. Die Kopplungs-Abbildung (5.11) stellt eine Vorschrift für eine instationäre, nichtlineare Iteration erster Ordnung dar, deren Fixpunkt der Lösung des gekoppelten Problems entspricht. Ausgehend von einem Prädiktor für die Verschiebungen am Kopplungsrand wird in jedem Iterationsschritt $(k+1)$ die Abbildung $\mathcal{T}^{(k+1)}(\mathbf{d}_\Gamma^{(k)})$ ausgeführt. Dies entspricht wiederum der sukzessiven Anwendung des Netz-, Fluid- und Struktur-Operators, wie in Abbildung 5.4 gezeigt. Auch bei der Fixpunkt-Iteration können die Verschiebungen am Kopplungsrand am Ende jedes Iterationsschritts relaxiert werden.

Das von Quarteroni (1991) vorgeschlagene iterative Dirichlet-Neumann-Substruktur-Verfahren zur Lösung von allgemeinen oberflächengekoppelten Problemen wurde von Le

Tallec und Mouro (1998, 2001) auf die Fluid-Struktur-Wechselwirkung angewendet und auf Stabilität, Konvergenz, Energieerhaltung und Genauigkeit untersucht. Iterative Substruktur-Methoden werden auch als Schurkomplement[4]-Verfahren bezeichnet und beruhen auf nichtüberlappenden Gebietszerlegungsverfahren. Diese wurden in den 1960er Jahren im Bereich der Strukturmechanik entwickelt, um große Tragstrukturen auf den damals noch sehr beschränkten Speicher- und Rechenkapazitäten mit Matrizenmethoden zu analysieren. Durch Kondensation der inneren Freiheitsgrade der Teilgebiete wird das Problem auf ein relativ kleines Interface-System reduziert, das mit direkten Lösern berechnet wird. Bei den iterativen Substruktur-Verfahren wird dieses Interface-System mit vorkonditionierten Krylow-Unterraum-Verfahren gelöst, wobei die erforderlichen Schurkomplement-Matrizen nicht explizit berechnet werden, sondern nur ihre Anwendung auf einen Vektor durch die Lösung eines der Teilgebiete mit entsprechenden Randbedingungen bestimmt wird. Eine ausführlichere Beschreibung der Entwicklung der Substruktur-Methoden und deren Anwendung auf die Fluid-Struktur-Wechselwirkung mit Konvergenzuntersuchungen ist in Mok (2001) zu finden.

Diese drei Herleitungen eines iterativ gestaffelten Kopplungsverfahrens basieren auf unterschiedlichen Darstellungen des gekoppelten Problems, führen jedoch auf denselben Algorithmus. Daher werden sie in dieser Arbeit unter dem Namen Block-Gauß-Seidel-Verfahren mit Prädiktor und Relaxation zusammengefasst. Die notwendigen Rechenschritte, um ausgehend von einem bekannten Zustand zum Zeitpunkt t_n die gekoppelte Lösung zum neuen Zeitpunkt t_{n+1} zu berechnen, sind im Struktogramm nach Nassi und Shneiderman (1973) in Abbildung 5.5 noch einmal zusammengefasst. Das sequenziell gestaffelte Kopplungsverfahren entspricht einer Iteration des Block-Gauß-Seidel-Verfahrens.

Das Block-Gauß-Seidel-Verfahren, besonders in Verbindung mit einer Relaxation nach dem später beschriebenen Aitken[5]-Verfahren, hat sich neben verschiedenen Newton-Krylow-Verfahren besonders im Bereich der Wechselwirkung von leichten Strukturen mit Strömungen als Standardverfahren etabliert. Es wird von vielen verschiedenen Anwendern auf unterschiedlichen Problemfeldern der Fluid-Struktur-Wechselwirkung eingesetzt. Zum Beispiel setzen Wüchner (2007), Glück u. a. (2003), Glück (2002) und Halfmann u. a. (2000, 2001) Block-Gauß-Seidel-Verfahren zur Simulation der Interaktion von Membrantragwerken mit Windströmungen ein. Die Interaktion von dünnen, elastischen Ventilen, wie z.B. Herzklappen, mit inkompressiblen Strömungen unter Berücksichtigung von Kontakt lösen Diniz dos Santos u. a. (2008) mit Block-Iterations-Verfahren, während Hofman (2003) das Block-Gauß-Seidel-Verfahren zur Simulation der Klimatisierung von Flugzeugkabinen anwendet, wobei hier das Fluid mit einem

[4] Issai Schur, * 10. Januar 1875 in Mohilew, † 10. Januar 1941 in Tel Aviv, Mathematiker.
[5] Alexander Craig Aitken, * 1. April 1895 in Dunedin, † 3. November 1967 in Edinburgh, neuseeländischer Mathematiker.

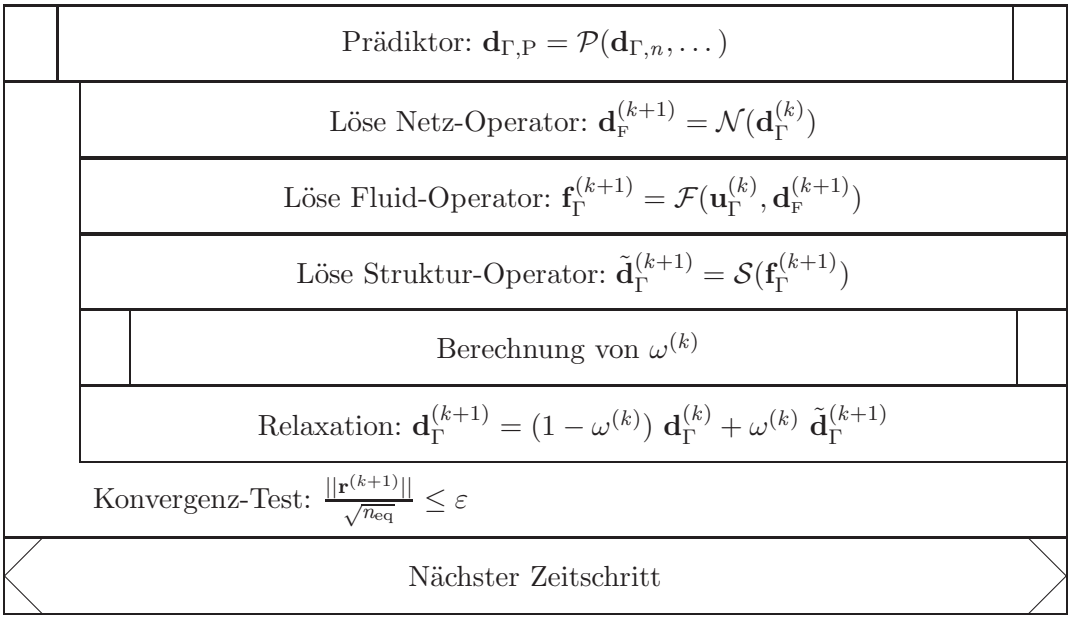


Abbildung 5.5: Struktogramm für einen Zeitschritt des Block-Gauß-Seidel-Verfahrens mit Prädiktor und Relaxation.

„controller"-Modell gekoppelt wird. Zunächst explizite (Geller u. a. 2005) und später auch implizite Algorithmen setzen Kollmannsberger u. a. (2006) zur Kopplung von Strukturelementen hoher Ordnung mit einem Lattice-Boltzmann[6]-Löser für das Fluid ein. Barcelos und Maute (2008) verwenden das nichtlineare Block-Gauß-Seidel-Verfahren zur Lösung von stationären Fluid-Struktur-Wechselwirkungs-Problemen im Rahmen der Formoptimierung von Flugzeugflügeln.

Block-Jacobi-Verfahren

Analog zu den klassischen Iterationsverfahren für lineare Gleichungssysteme kann auch auf das gekoppelte Problem (5.9) eine Jacobi-Iteration angewendet werden. Dies bedeutet, dass im Vergleich zum Block-Gauß-Seidel-Verfahren der Struktur-Operator nicht mit den aktuellen Kräften auf dem Kopplungsrand $\mathbf{f}_{\Gamma}^{(k+1)}$ berechnet wird, sondern mit den Kräften, die im letzten Iterationsschritt berechnet wurden $\mathbf{f}_{\Gamma}^{(k)}$ (Farhat und Lesoinne 2000). Zu Beginn der Iteration kann für beide Felder ein Prädiktor (vgl. Abschnitt 5.5.3) ausgeführt werden, um die Konvergenz zu beschleunigen. Der Ablauf der Berechnung innerhalb eines Zeitschritts ist in Abbildung 5.6 dargestellt.

Innerhalb einer Iteration sind die Lösungen des Fluid- und des Struktur-Operators unabhängig voneinander. Erst am Ende jedes Iterationsschritts werden die Kopplungs-

[6]Ludwig Boltzmann, * 20. Februar 1844 in Linz, † 5. September 1906 in Duino, österreichischer Physiker und Philosoph.

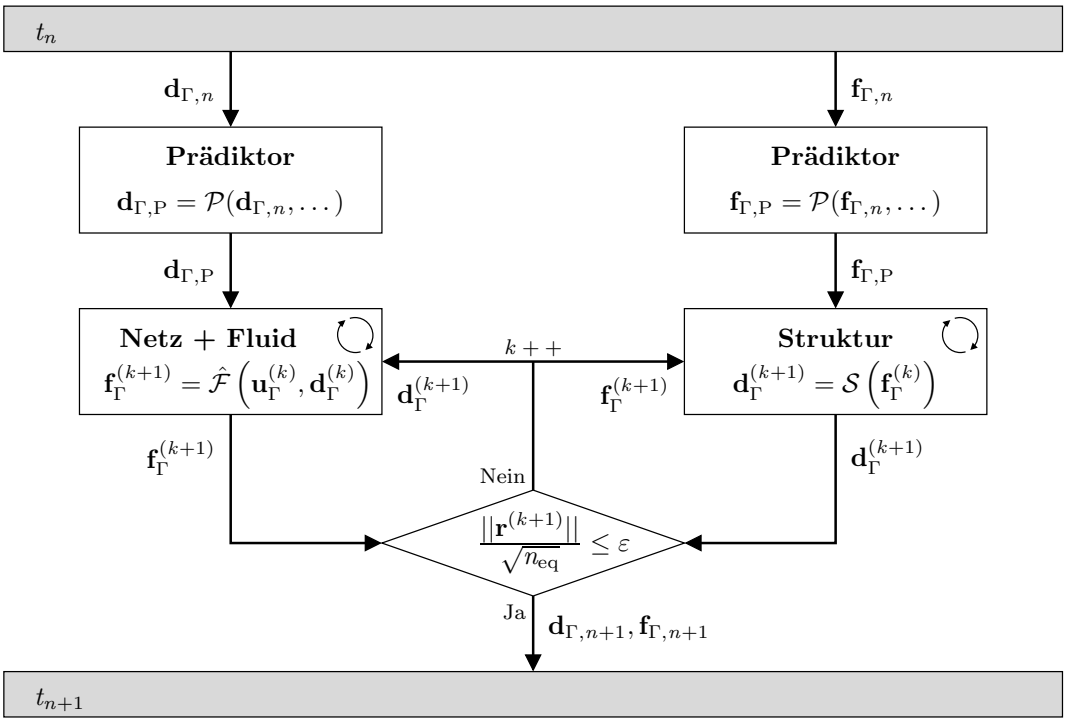


Abbildung 5.6: Ablaufdiagramm zum Block-Jacobi-Verfahren mit Prädiktor.

informationen in beide Richtungen ausgetauscht. Dadurch können Fluid und Struktur parallel zueinander gelöst werden. Beide Codes können auch hier auf unterschiedlichen, jeweils optimal geeigneten Hardware-Plattformen laufen und die Kopplungsinformationen über ein Netzwerk austauschen.

Wie bereits bei den Iterationsverfahren für lineare Gleichungssysteme ausgeführt (vgl. Abschnitt 4.3.2) konvergiert das Block-Jacobi-Verfahren durch den reduzierten Austausch der Kopplungsinformationen deutlich langsamer als das Block-Gauß-Seidel-Verfahren. Cervera u. a. (1996) ist zum Beispiel zu entnehmen, dass mit dem Block-Jacobi-Verfahren ungefähr doppelt so viele Iterationen benötigt werden, wie mit dem Block-Gauß-Seidel-Verfahren, um für ein Fluid-Struktur-Interaktions-Problem zum gleichen Ergebnis zu kommen. Verglichen mit den einfach gestaffelten Verfahren entspricht das Block-Jacobi-Verfahren einer wiederholten Anwendung des parallel gestaffelten Verfahrens in jedem Zeitschritt.

Konvergenz-Beschleunigung durch Relaxation

Sowohl beim Block-Jacobi- als auch beim Block-Gauß-Seidel-Verfahren ist die Konvergenz nicht in jedem Fall garantiert (Mok 2001; Mok und Wall 2001). Causin u. a. (2005) haben gezeigt, dass der „artificial added mass“-Effekt auch die Konvergenz der

iterativ gestaffelten Verfahren beeinflusst. Ihre Untersuchungen haben ergeben, dass für den Grenzfall $\Delta t \to 0$ eine Relaxation von $\omega < 1$ erforderlich ist, damit die Iteration über die Felder konvergiert.

Die Relaxation der Verschiebungen am Kopplungsrand ist in manchen Fällen einerseits zwingend erforderlich, um für die Iteration über die gekoppelten Felder Konvergenz zu erreichen. Andererseits kann durch einen optimal gewählten Relaxationsparameter die Konvergenz beschleunigt werden. Dies ist aufgrund des großen Rechenaufwands für jeden Iterationsschritt ein wichtiger Aspekt der Relaxation. Da der optimale Relaxationsparameter nicht für die gesamte instationäre Simulation konstant ist (Mok 2001), werden mit festen, experimentell oder durch „trial-and-error“ bestimmten Werten für ω nur bedingt gute Ergebnisse erzielt.

Um eine schnelle Konvergenz der Felditeration zu erreichen, ist es erforderlich für jeden Iterationsschritt den optimalen Relaxationsparameter zu bestimmen. Dazu wurden von Mok (2001) zwei Methoden vorgestellt, die im Folgenden kurz beschrieben werden.

Aitken-Verfahren Das ursprünglich von Aitken (1937) vorgeschlagene Grundverfahren „Aitken's Δ^2-Methode“ konvertiert eine beliebige, konvergierende, skalare Folge in eine schneller konvergierende Folge und wurde für die iterative Bestimmung von Eigenwerten angewendet. Von verschiedenen Autoren wurde das Verfahren für die Lösung von vektoriellen Gleichungen erweitert. Diese auch für nichtlineare Gleichungssysteme einsetzbare Aitken-Methode für vektorielle Gleichungen wird in der Formulierung von Irons und Tuck (1969) verwendet, um in jedem Iterationsschritt den optimalen Relaxationsparameter zu bestimmen.

Aus der Verbesserung der Interface-Verschiebungen im aktuellen ($\Delta\mathbf{d}_\Gamma^{(k+1)}$) und im letzten Iterationsschritt ($\Delta\mathbf{d}_\Gamma^{(k)}$)

$$\Delta\mathbf{d}_\Gamma^{(k+1)} = \mathbf{d}_\Gamma^{(k)} - \tilde{\mathbf{d}}_\Gamma^{(k+1)}; \qquad \Delta\mathbf{d}_\Gamma^{(k)} = \mathbf{d}_\Gamma^{(k-1)} - \tilde{\mathbf{d}}_\Gamma^{(k)} \tag{5.18}$$

wird der aktuelle Aitken-Faktor $\mu^{(k)}$ berechnet:

$$\mu^{(k)} = \mu^{(k-1)} + (\mu^{(k-1)} - 1)\frac{(\Delta\mathbf{d}_\Gamma^{(k)} - \Delta\mathbf{d}_\Gamma^{(k+1)})^{\mathrm{T}}\ \Delta\mathbf{d}_\Gamma^{(k+1)}}{(\Delta\mathbf{d}_\Gamma^{(k)} - \Delta\mathbf{d}_\Gamma^{(k+1)})^2} \qquad \text{für } i > 0. \tag{5.19}$$

Für den ersten Iterationsschritt in jedem Zeitschritt wird der Aitken-Faktor zu null gesetzt. Der aktuelle Relaxationsparameter ergibt sich aus dem Aitken-Faktor zu

$$\omega^{(k)} = 1 - \mu^{(k)}. \tag{5.20}$$

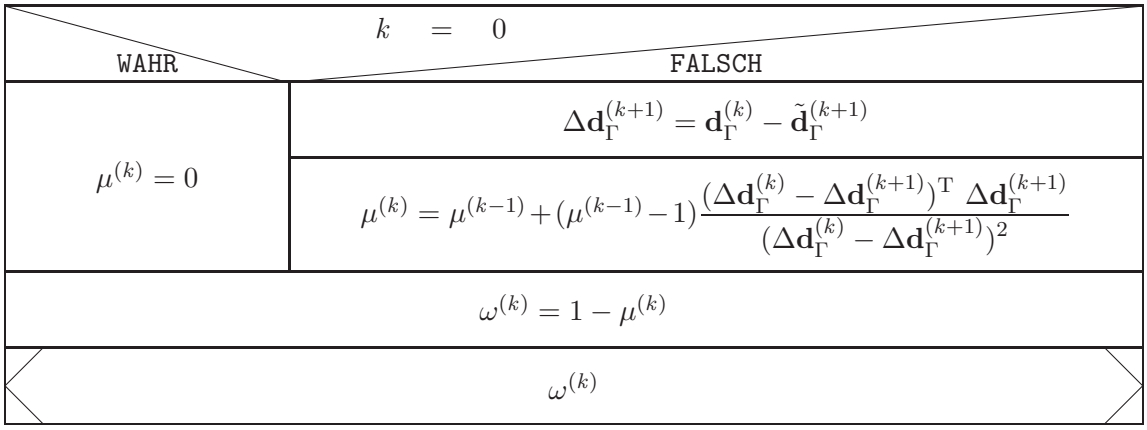


Abbildung 5.7: Struktogramm zur Berechnung des Relaxationsparameters $\omega^{(k)}$ mit dem Aitken-Verfahren.

Im Struktogramm in Abbildung 5.7 sind die notwendigen Schritte zur Berechnung des Relaxationsparameters nach dem Aitken-Verfahren zusammengefasst.

Wie aus den Gleichungen (5.18) bis (5.20) erkennbar, sind außer drei Vektor-Additionen und zwei Skalarprodukten mit den Interface-Verschiebungen nur skalare Operationen erforderlich, um den Relaxationsparameter zu bestimmen. Dieser sehr geringe Rechenaufwand ist der große Vorteil des Aitken-Verfahrens. Auch wenn für die Aitken-Methode für vektorielle Gleichungen keine gesicherten Konvergenzaussagen und -analysen existieren, hat sich gezeigt, dass die Methode für viele numerische Anwendungen sehr gut genutzt werden kann.

Gradienten-Verfahren Die Verwendung des Gradienten-Verfahrens zur Bestimmung optimaler, iterationsabhängiger Relaxationsparameter wurde zunächst von Le Tallec und Mouro (1998, 2001) angeregt, jedoch nicht weiter ausgeführt. Ausgehend von dieser Idee beschreibt Mok (2001) ausführlich die Anwendung des Gradienten-Verfahrens für die Konvergenzbeschleunigung.

Wie im Abschnitt 4.4.1 beschrieben, wird dabei als Suchrichtung $\mathbf{p}^{(k)}$ der aktuelle Gradient, der gleich dem Residuenvektor ist, gewählt (vgl. Gleichung (4.28)). Für das gekoppelte Problem entspricht der Residuenvektor der iterativen Verbesserung und kann als Differenz zwischen alter und neuer Interface-Verschiebung berechnet werden:

$$\mathbf{p}^{(k)} = \tilde{\mathbf{d}}_\Gamma^{(k+1)} - \mathbf{d}_\Gamma^{(k)}. \tag{5.21}$$

Die optimale Schrittweite folgt aus Gleichung (4.27), wobei der Ausdruck $\mathbf{A}\mathbf{p}^{(k)}$ unter Verwendung des Iterations-Operators des gekoppelten Systems ausgeschrieben werden

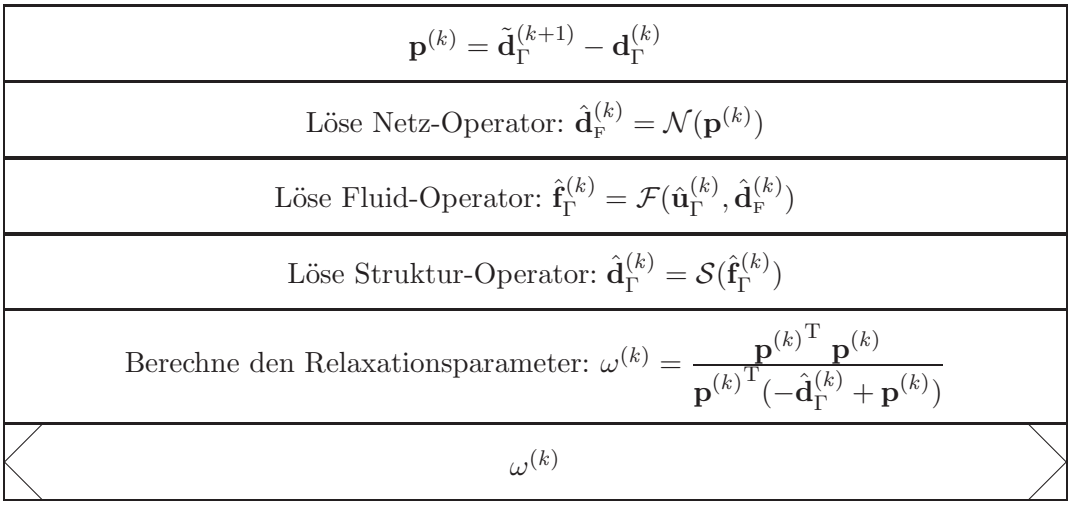


Abbildung 5.8: Struktogramm zur Berechnung des Relaxationsparameters $\omega^{(k)}$ mit dem Gradienten-Verfahren.

kann:

$$\omega^{(k)} = \frac{\mathbf{p}^{(k)^\mathrm{T}}\,\mathbf{p}^{(k)}}{\mathbf{p}^{(k)^\mathrm{T}}\,\mathbf{A}\mathbf{p}^{(k)}} = \frac{\mathbf{p}^{(k)^\mathrm{T}}\,\mathbf{p}^{(k)}}{\mathbf{p}^{(k)^\mathrm{T}}(\mathbf{S}_\mathrm{S}^{-1}\mathbf{S}_\mathrm{F}\;\mathbf{p}^{(k)} + \mathbf{p}^{(k)})}. \tag{5.22}$$

Für die Bestimmung des Ausdrucks $\mathbf{S}_\mathrm{S}^{-1}\mathbf{S}_\mathrm{F}\;\mathbf{p}^{(k)}$, der die Schurkomplement-Matrizen für die Struktur $\mathbf{S}_\mathrm{S}$ und das Fluid $\mathbf{S}_\mathrm{F}$ enthält, leitet Mok (2001) eine schurkomplementfreie Ermittlung her. Diese erfordert eine weitere Lösung beider physikalischer Felder, wobei als einzige Randbedingung der Residuenvektor die Lage des Kopplungsrands und damit des Fluidgebiets vorgibt.

$$\hat{\mathbf{d}}_\Gamma^{(k)} = \mathbf{S}_\mathrm{S}^{-1}\mathbf{S}_\mathrm{F}\;\mathbf{p}^{(k)} = \mathcal{S}\Big(\mathcal{F}\big(\mathbf{u}_\Gamma,\mathcal{N}(\mathbf{p}^{(k)})\big)\Big) \tag{5.23}$$

Die Schrittweite, d.h. der Relaxationsparameter ergibt sich damit aus:

$$\omega^{(k)} = \frac{\mathbf{p}^{(k)^\mathrm{T}}\,\mathbf{p}^{(k)}}{\mathbf{p}^{(k)^\mathrm{T}}(-\hat{\mathbf{d}}_\Gamma^{(k)} + \mathbf{p}^{(k)})}. \tag{5.24}$$

Die erforderlichen Operationen zur Bestimmung des Relaxationsparameters mit dem Gradienten-Verfahren sind im Struktogramm in Abbildung 5.8 zusammengefasst.

Mit dem Gradienten-Verfahren können bezüglich der aktuellen Suchrichtung optimale Relaxationsparameter bestimmt werden (lokale Optimalitätseigenschaft). In manchen Fällen führt dies zu einer schnelleren Konvergenz als mit der Aitken-Methode. Jedoch ist der numerische Aufwand deutlich höher als beim Aitken-Verfahren. Zur Bestimmung des Relaxationsparameters mit dem Gradienten-Verfahren müssen sowohl das Fluid als

auch die Struktur erneut gelöst werden. Dies sind jedoch keine iterativen, nichtlinearen Lösungen, da sich das Residuum $\mathbf{p}^{(k)}$ auf die aktuelle Systemkonfiguration bezieht und alle nichtlinearen Terme mit den aktuellen Verschiebungen bzw. Geschwindigkeiten ausgewertet werden. Für jedes Feld ist somit nur ein einmaliges Lösen erforderlich.

Da der höhere numerische Aufwand im Vergleich zur nur geringen Verbesserung der Konvergenz in den meisten Fällen zu längeren Rechenzeiten führt, wird das Gradienten-Verfahren in dieser Form in der Regel nicht angewendet. Es wird an dieser Stelle nur als Grundlage für ein Zwei-Level-Verfahren, das im folgenden Abschnitt vorgestellt wird, beschrieben.

5.6.2 Zwei-Level-Verfahren

Bereits Fedorenko (1964) führte zur Lösung der Wärmeleitungsgleichung mit der Methode der Finiten Differenzen eine zweite, gröbere Diskretisierung ein und konnte zeigen, dass mit dieser Methode die Anzahl benötigter Operationen zur Erreichung einer vorgegebenen Genauigkeit nur $O(n)$ beträgt. Seitdem ist besonders im Bereich der Lösung von partiellen Differenzialgleichungen die Aufmerksamkeit für Mehrgitter-Verfahren aufgrund ihres geringen numerischen Aufwands, speziell für sehr große Gleichungssysteme, stetig gestiegen.

Mehrgitter-Methoden verwenden die Tatsache, dass Iterationsverfahren kurzwellige Fehleranteile besonders schnell aus der Lösung entfernen. Wird daraufhin ein Residuum, das fast nur noch aus langwelligen Fehlern besteht, auf eine gröbere Diskretisierung übertragen, erscheinen die Fehler dort als kurzwellige Fehleranteile, die mit wenigen Iterationen eliminiert werden können. Zu den grundlegenden Arbeiten, die eine gute Einführung und einen Überblick über Mehrgitter-Methoden bieten, zählen Hackbusch (1985), Braess (1995), Wagner (1998), Briggs u. a. (2000) und Stüben (2001).

Auch bei der Simulation von Fluid-Struktur-Wechselwirkungs-Problemen werden vermehrt Mehrgitter-Verfahren in den Einzelfeldlösern eingesetzt. So stellen Gee u. a. (2007) sowohl einen algebraischen Mehrgitter-Löser für dünne Schalenstrukturen (Gee 2004) als auch spezielle Prolongatoren für anisotrope Problemstellungen (Gee u. a. 2008) vor und wenden diese für die Einzelfeldlöser von partitionierten Fluid-Struktur-Wechselwirkungs-Simulationen an.

Relativ neu ist die Idee, diese Algorithmen auch auf die Lösung von gekoppelten Problemen zu übertragen. Da es sich hierbei in der Regel um nichtlineare Zusammenhänge handelt, lässt sich die Theorie der Mehrgitter-Verfahren nicht direkt auf diese Problemklasse anwenden. Es ist jedoch möglich zugrunde liegende Prinzipien auf die Lösung von

gekoppelten Problemen anzuwenden. Dabei hat sich gezeigt, dass die Verwendung von mehreren Leveln auch hier zu einer Effizienzsteigerung führen kann.

van Zuijlen u. a. (2007) haben, wahrscheinlich als Erste, zwei verschiedene Zwei-Level-Algorithmen zur partitionierten Lösung von Fluid-Struktur-Wechselwirkungs-Problemen vorgestellt. Hierbei handelt es sich um eine Grob-Gitter-Korrektur des Partitionierungsfehlers und einen Grob-Gitter-Prädiktor. Die Autoren haben den Effizienzgewinn durch diese beiden Algorithmen anhand einer Finite-Volumen-Diskretisierung eines eindimensionalen Modellproblems gezeigt.

Einen Mehr-Level-Ansatz für die Fluid-Struktur-Wechselwirkung stellen Sternel u. a. (2008) vor. Sie koppeln einen Mehrgitter-Löser für das Fluidfeld in jedem Level explizit mit dem Strukturlöser, der nicht unbedingt auf mehreren Leveln arbeiten muss. Dadurch gelingt es für das vorgestellte Beispiel eines überströmten Hohlraums mit flexibler Bodenplatte, die Anzahl an Kopplungsiterationen im Vergleich zum Block-Gauß-Seidel-Verfahren um ungefähr die Hälfte zu reduzieren. Es werden jedoch keine Aussagen über die Rechenzeiten gemacht, für die die Ersparnis geringer ausfallen dürfte, da eine Iteration des vorgestellten „multi-level transfer“-Algorithmus numerisch aufwändiger ist als beim Block-Gauß-Seidel-Verfahren.

In den folgenden Abschnitten werden zwei Verfahren für die dreidimensionale Fluid-Struktur-Wechselwirkung vorgestellt, die im Rahmen dieser Arbeit entwickelt wurden (von Scheven u. a. 2007, Ramm u. a. 2008). Sie beruhen auf den Grundideen der Mehrgitter-Methoden und den in den vorangegangenen Abschnitten vorgestellten Kopplungsverfahren.

Grob-Gitter-Prädiktor

Das erste Verfahren, der Grob-Gitter-Prädiktor („coarse grid predictor“ CGP), versucht durch einen Prädiktorschritt, der nicht nur auf den Lösungen vorangegangener Zeitschritte beruht, die benötigten Iterationen auf der feinen Diskretisierung, dem eigentlichen Rechennetz, zu reduzieren. Die Lage des Kopplungsrands, die aus der Lösung des aktuellen Zeitschritts auf einer gröberen Diskretisierung hervorgeht, wird hierbei als Prädiktor für die Iteration auf dem feinen Netz verwendet.

Dabei spielt es keine Rolle, welches Iterationsverfahren auf den beiden Diskretisierungen verwendet wird. In der folgenden Beschreibung des Grob-Gitter-Prädiktors wird ein Block-Gauß-Seidel-Verfahren auf beiden Leveln angenommen, es können aber auch Newton-Krylow-Verfahren (siehe Abschnitt 5.6.3) oder andere implizite Kopplungsverfahren verwendet werden. Auch ist es möglich für die Lösung auf dem groben Netz ein anderes Kopplungsverfahren einzusetzen als auf der feinen Diskretisierung.

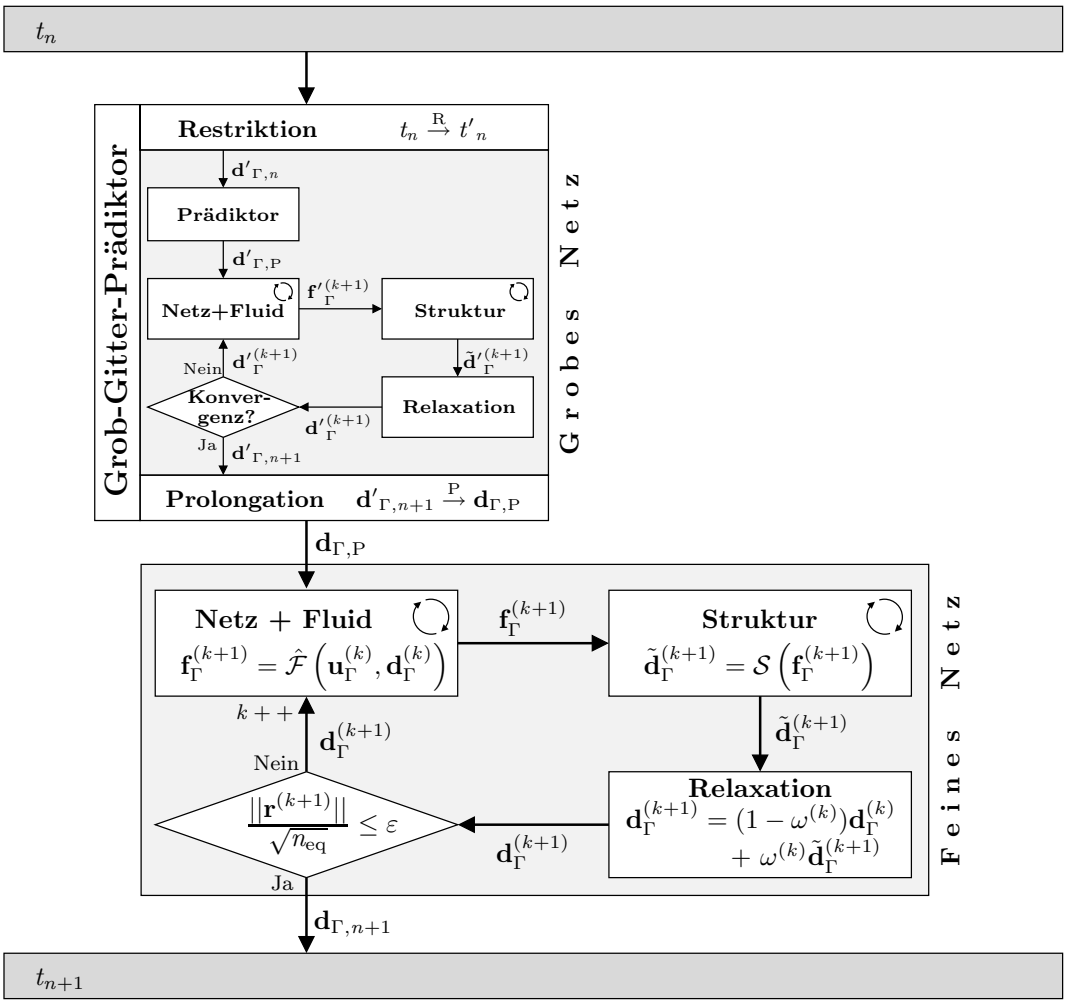


Abbildung 5.9: Ablaufdiagramm zum Block-Gauß-Seidel-Verfahren mit Grob-Gitter-Prädiktor.

Der Ablauf eines Block-Gauß-Seidel-Verfahrens mit Grob-Gitter-Prädiktor ist in Abbildung 5.9 dargestellt. Die Größen, die sich auf das grobe Netz beziehen, werden mit einem Apostroph gekennzeichnet: z.B. $\mathbf{d}'^{(k)}_{\Gamma}$.

Zu Beginn eines jeden Zeitschritts werden die aktuellen Zustandsvariablen von Fluid und Struktur vom feinen Gitter auf die grobe Diskretisierung übertragen. Diese sogenannte Restriktion wird hier mit der Schreibweise $t_n \xrightarrow{\mathrm{R}} t'_n$ abgekürzt und im Anschluss an die Erläuterung des Grob-Gitter-Prädiktors näher beschrieben. Durch die Restriktion aller Zustandsgrößen in jedem Zeitschritt wird sichergestellt, dass nicht zwei voneinander unabhängige, instationäre Berechnungen auf den beiden Diskretisierungen durchgeführt werden, deren Lösungen sich immer weiter voneinander entfernen. Der Prädiktorschritt basiert so immer auf der letzten auskonvergierten Lösung auf dem feinen Netz.

Auf dem groben Gitter wird nach der Restriktion eine Block-Gauß-Seidel-Iteration gemäß Abschnitt 5.6.1 durchgeführt. Dabei wird ein klassischer Prädiktor (Abschnitt 5.5.3) und ein beliebiges Relaxationsverfahren verwendet. Diese Iteration wird bis zur Konvergenz ausgeführt, dabei ist es aber möglich, das Abbruchkriterium weniger streng als auf dem feinen Gitter zu wählen.

Die konvergierte Lösung für die Verschiebungen des Kopplungsrands auf dem groben Gitter $\mathbf{d}'_{\Gamma,n+1}$ wird abschließend auf die Randknoten des feinen Netzes prolongiert

$$\mathbf{d}'_{\Gamma,n+1} \xrightarrow{\mathrm{P}} \mathbf{d}_{\Gamma,\mathrm{P}} \tag{5.25}$$

und dient als Startwert für eine Block-Gauß-Seidel-Iteration auf der feinen Diskretisierung.

Wird als Prädiktor für die Grob-Gitter-Iteration kein klassischer Prädiktor gemäß Gleichung (5.14) gewählt, sondern wiederum ein Grob-Gitter-Prädiktor auf einer dritten, noch gröberen Diskretisierung, lassen sich Mehr-Level-Verfahren mit beliebig vielen Leveln konstruieren. Diese durch rekursive Anwendung des Grob-Gitter-Prädiktors gebildeten Lösungsverfahren entsprechen einem halben V-Zyklus der klassischen Mehrgitter-Algorithmen.

Die notwendigen Berechnungsschritte für einen Prädiktor auf der Basis einer Grob-Gitter-Lösung des gekoppelten Problems sind im Struktogramm in Abbildung 5.10 zusammengefasst.

Restriktion und Prolongation Eine Restriktion bezeichnet im Allgemeinen eine Einschränkung der Definitionsmenge einer Funktion. Im Zusammenhang von Mehrgitter-Lösern wird damit die Approximation einer Funktion (z.B. Lösung oder Residuum), die auf einem feinen Gitter definiert ist, durch eine Funktion auf einem groben Gitter mit weniger Freiwerten beschrieben.

Wird das feine Gitter durch eine gleichmäßige Unterteilung eines gröberen Netzes (wie in Abschnitt 3.2.4 beschrieben) erzeugt, kann die Restriktion durch Vernachlässigung aller Freiwerte an Knoten des feinen Gitters erfolgen, die nicht mit einem Knoten des groben Gitters zusammenfallen. Dadurch ergibt sich auf dem groben Netz eine Approximation der ursprünglichen Funktion, die an den Grob-Gitter-Knoten exakt ist und dazwischen die Ordnung der Grob-Gitter-Ansatzfunktionen hat.

Für den Grob-Gitter-Prädiktor werden alle Zustandsgrößen des aktuellen Zeitschritts mittels Restriktion auf das grobe Netz übertragen. Dazu gehören neben den aktuellen Geschwindigkeiten und Drücken im Fluidgebiet auch deren Werte für vergangene Zeitschritte und die Lage des Kopplungsrands und damit des Fluidgebiets. Für die Struk-

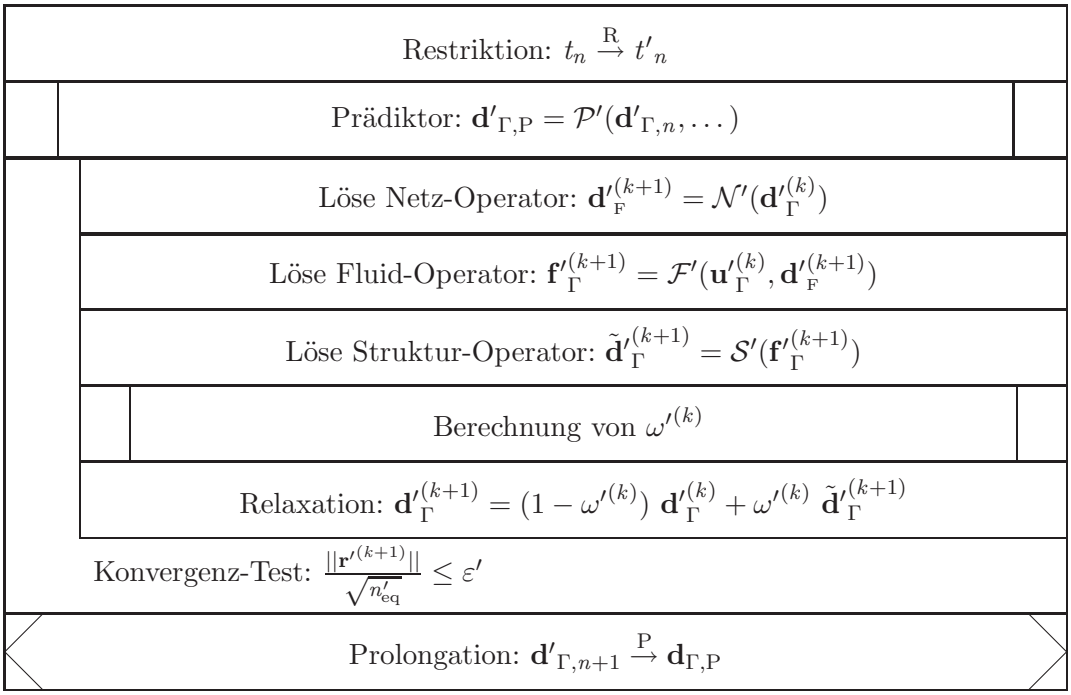


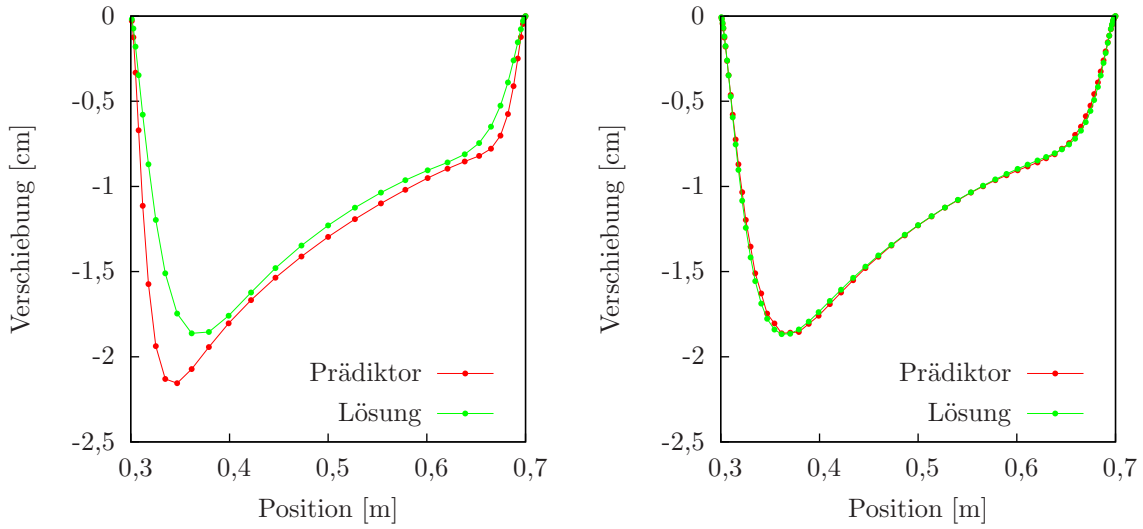
Abbildung 5.10: Struktogramm für den Grob-Gitter-Prädiktor.

tur müssen der aktuelle Verschiebungszustand und je nach verwendeter Kinematik und konstitutiven Gleichungen noch weitere Größen auf das grobe Gitter übertragen werden. Das Ziel ist die Restriktion des kompletten aktuellen Zustands des gekoppelten Problems auf das grobe Gitter, damit der Ausgangspunkt für den Prädiktorschritt immer mit der letzten auskonvergierten Lösung des feinen Netzes übereinstimmt.

Die Prolongation beschreibt den entgegengesetzten Prozess zur Restriktion. Durch sie wird eine Funktion, die auf dem groben Netz definiert ist, auf das feinere Gitter übertragen. Wird das feine Gitter durch eine gleichmäßige Unterteilung des groben Netzes erzeugt, können die Funktionswerte an den Knoten des feinen Netzes mithilfe der Ansatzfunktionen der Grob-Gitter-Elemente bestimmt werden. Dies führt auf dem feinen Gitter auf einem identischen Funktionsverlauf, der aber mit mehr Freiwerten definiert ist.

Beim Grob-Gitter-Prädiktor wird nur die auf dem groben Netz auskonvergierte Position des Kopplungsrands auf das feine Gitter prolongiert und als Prädiktorwert für die Iteration auf diesem verwendet. Für alle anderen Größen (z.B. Geschwindigkeiten und Drücke) werden die Werte des letzten Zeitschritts auf dem feinen Netz verwendet.

Untersuchung der Konvergenz Im folgenden Abschnitt wird die Wirkungsweise des Grob-Gitter-Prädiktors untersucht. Dazu wird für den dritten Zeitschritt der Simulation



(a) Grob-Gitter-Iteration

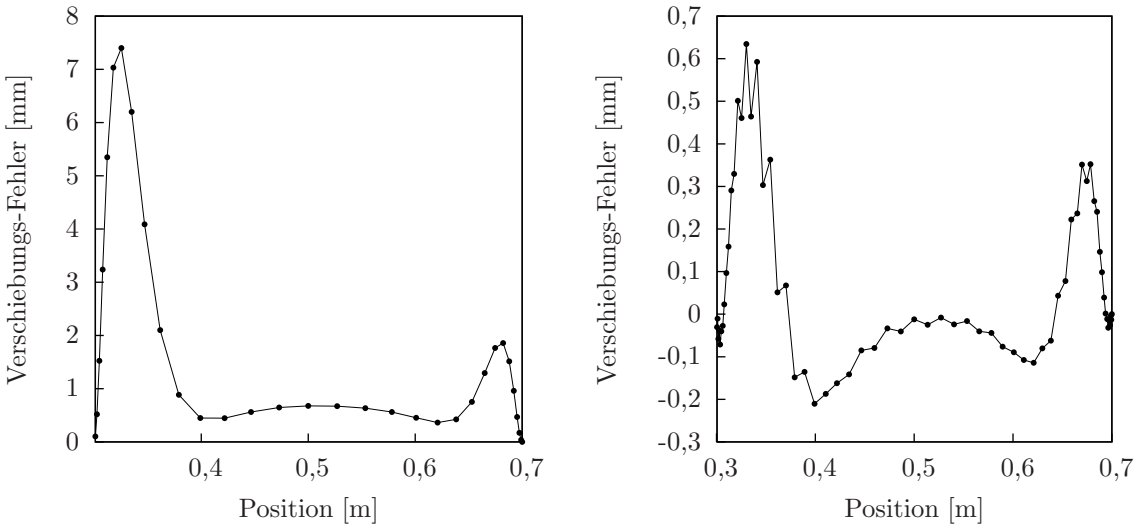
(b) Fein-Gitter-Iteration

Abbildung 5.11: Verschiebungen am Kopplungsrand auf dem groben und feinen Gitter jeweils zu Beginn und am Ende der Iteration.

des in Abschnitt 5.4 (Abbildung 5.1) eingeführten Beispiels mit $\rho_S = 50{,}0\ \mathrm{kg/m^3}$ die Lage des Kopplungsrands in verschiedenen Stadien innerhalb des Zeitschritts betrachtet. In den folgenden Diagrammen (Abbildung 5.11) ist jeweils die vertikale Verschiebungskomponente am Kopplungsrand und damit an der Oberseite der Struktur, die von $x = 0{,}3$ m bis $x = 0{,}7$ m reicht, aufgetragen.

Zur Lösung dieses Zeitschritts mit dem Block-Gauß-Seidel-Verfahren mit Aitken-Relaxation, aber ohne einen Grob-Gitter-Prädiktor, sind acht Iterationen auf der feinen Diskretisierung erforderlich. Verwendet man dagegen zusätzlich einen Grob-Gitter-Prädiktor wird zunächst der bekannte Zustand zu Beginn des Zeitschritts auf das grobe Gitter übertragen und dort die Block-Gauß-Seidel-Iteration mit einem klassischen Prädiktor zweiter Ordnung gemäß Gleichung (5.14) gestartet. Die Lage des Kopplungsrands nach diesem Prädiktor zweiter Ordnung auf dem groben Gitter ist in Abbildung 5.11 (a) durch die gestrichelte Linie angegeben. Nach acht Block-Gauß-Seidel-Iterationen mit Aitken-Relaxation auf dem groben Gitter ist dort die konvergierte Lösung gefunden, die im selben Diagramm durch die durchgezogenen Linie gezeigt ist. Der Fehler in den Verschiebungen, der durch diese acht Iterationen auf dem groben Netz aus der Lösung entfernt wurde, ist in Abbildung 5.12 (a) dargestellt. Er zeigt einen relativ glatten Verlauf mit zwei Maxima in der Nähe der beiden Ränder der Struktur, wo hohe Gradienten in der Lösung auftreten.

Die konvergierte Lösung des groben Gitters wird nun auf die feine Diskretisierung prolongiert und dient dort als Prädiktor für die Block-Gauß-Seidel-Iteration. Der Verlauf



(a) Grob-Gitter-Iteration **(b)** Fein-Gitter-Iteration

Abbildung 5.12: Korrektur der Verschiebungen am Kopplungsrand durch die Grob- und Fein-Gitter-Iteration.

ist in Abbildung 5.11 (b) als gestrichelte Linie angegeben. Die Block-Gauß-Seidel-Iteration mit Aitken-Relaxation auf dem feinen Netz konvergiert nach nur fünf Iterationen. Die auskonvergierte Lösung, die einer Lösung auf der feinen Diskretisierung ohne Grob-Gitter-Prädiktor entspricht, ist wiederum durch die durchgezogene Linie gezeigt. Wie bereits aus Abbildung 5.11 (b) ersichtlich, wird durch die Iterationen auf dem feinen Gitter nur noch ein relativ kleiner Fehleranteil aus der Lösung eliminiert. Bei genauer Betrachtung dieser Korrektur (Abb. 5.12 (b)) zeigt sich, dass der globale Verlauf dem der Korrektur durch die Grob-Gitter-Iteration entspricht, die Werte jedoch um eine Zehnerpotenz kleiner sind. Dieser Verlauf ist aber mit einem sehr kurzwelligen Fehleranteil überlagert, der auf dem groben Gitter nicht abgebildet werden kann.

Bereits an diesem exemplarischen Zeitschritt hat sich gezeigt, dass durch die Verwendung eines Grob-Gitter-Prädiktors die Zahl der Iterationen auf der feinen Diskretisierung von acht auf fünf reduziert werden konnte. Trotz der zusätzlich erforderlichen Iterationen auf dem groben Netz bedeutet dies eine Reduzierung der Gesamtrechenzeit für den Zeitschritt. Ausführliche Untersuchungen zu den Einsparmöglichkeiten bei Iterationsanzahl und Rechenzeit durch einen Grob-Gitter-Prädiktor werden im Abschnitt 5.7.3 beschrieben.

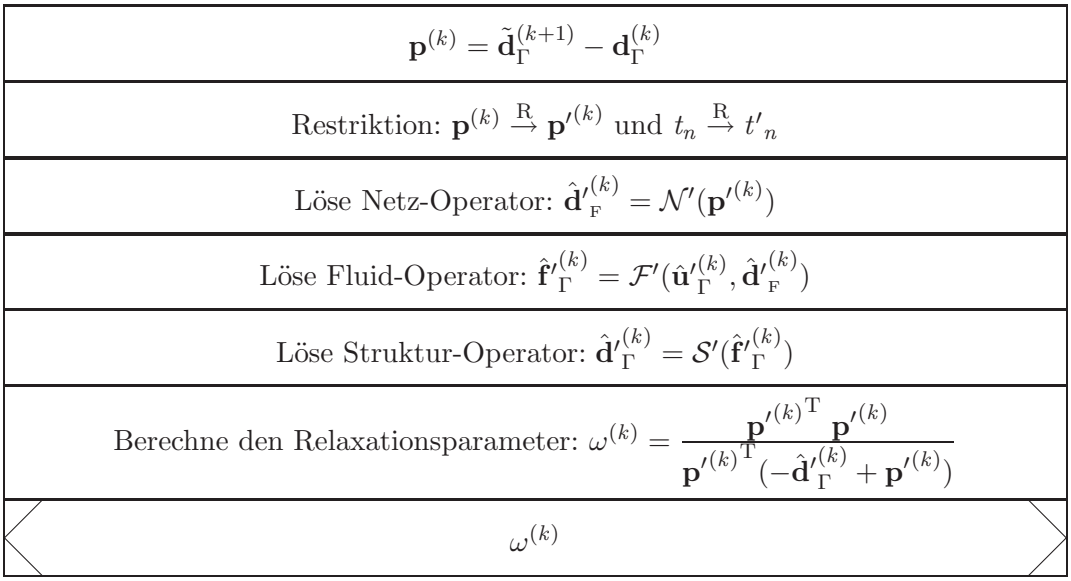


Abbildung 5.13: Struktogramm zur Berechnung des Relaxationsparameters ω_i mit dem Gradienten-Verfahren auf dem Grob-Gitter.

Grob-Gitter-Gradient

Eine weitere Möglichkeit zur Verwendung einer Lösung des gekoppelten Problems auf einem groben Gitter besteht bei der Bestimmung des Relaxationsparameters mit dem Gradienten-Verfahren. Die sehr zeitaufwändige Berechnung der Schrittweite $\omega^{(k)}$ nach Gleichung (5.23) und (5.24), bei der eine zusätzliche Lösung des Fluid- und Strukturproblems erforderlich ist, kann auch auf der groben Diskretisierung durchgeführt werden. Das entsprechende Vorgehen ist als Struktogramm in Abbildung 5.13 dargestellt.

Zuerst wird die aktuelle Suchrichtung, der Residuenvektor $\mathbf{p}^{(k)}$, auf das grobe Netz übertragen. Zusätzlich ist, ähnlich wie beim Grob-Gitter-Prädiktor, die Restriktion des aktuellen Zustands von Fluid- und Strukturgebiet erforderlich. Nun ist es möglich auf der groben Diskretisierung zunächst das Fluidproblem $\mathcal{F}'$ zu lösen, wobei der Residuenvektor $\mathbf{p}'^{(k)}$ als einzige Randbedingung die Lage des Kopplungsrands und damit des Fluidgebiets vorgibt. Nach der Lösung der Strukturgleichungen auf dem groben Netz $\mathcal{S}'$ kann mithilfe der daraus gewonnenen Lage des Kopplungsrands $\hat{\mathbf{d}}'^{(k)}_\Gamma$ der Relaxationsparameter $\omega^{(k)}$ nach Gleichung (5.24) auf der groben Diskretisierung berechnet werden. Eine Prolongation ist bei diesem Vorgehen nicht erforderlich, da nur der skalare Wert $\omega^{(k)}$ an die feine Diskretisierung übergeben und dort zur Relaxation verwendet wird.

Durch die Verwendung einer groben Diskretisierung kann der große Nachteil des Gradienten-Verfahren, der erhebliche Mehraufwand durch eine zusätzliche Lösung von Fluid- und Strukturgebiet, deutlich abgemindert werden. Jedoch bleibt der Rechenaufwand in jedem Iterationsschritt immer noch höher als beim Aitken-Verfahren. Außerdem geht

Bestimmung von ω_i	Iterationen	Rechenzeit [s]	Rechenzeit/Iteration [s]
Aitken-Verfahren	6	92,96	15,4
Gradienten-Verfahren	5	110,38	22,1
Grob-Gitter-Gradient	5	79,55	15,9

Tabelle 5.2: Vergleich der benötigten Iterationen und Rechenzeiten für verschiedene Relaxationsverfahren.

durch die approximative Bestimmung des Relaxationsparameters auf der groben Diskretisierung die lokale Optimalitätseigenschaft des Verfahrens verloren, d.h. in seltenen Fällen kann es passieren, dass mit dem Grob-Gitter-Gradienten mehr Iterationen erforderlich sind als mit dem klassischen Gradienten-Verfahren.

Die benötigte Anzahl an Iterationen und die Rechenzeiten zur Lösung eines exemplarischen Zeitschritts des in Abschnitt 5.4 vorgestellten Problems sind in Tabelle 5.2 angegeben. Hier werden die Bestimmung des Relaxationsparameters mit dem Aitken-Verfahren, dem klassischen Gradienten-Verfahren auf der feinen Diskretisierung und dem Grob-Gitter-Gradienten verglichen. Für den Fall, dass mit der Bestimmung des Relaxationsparameters mit dem Gradienten-Verfahren auf der groben Diskretisierung weniger Iterationen benötigt werden als beim Aitken-Verfahren, weist die Grob-Gitter-Gradienten-Methode auch die geringste Rechenzeit auf. Dies wird auch beim Vergleich der durchschnittlichen Rechenzeit pro Iteration in der letzten Spalte von Tabelle 5.2 deutlich. Der Aufwand für eine Iteration mit einem Grob-Gitter-Gradienten ist nur unwesentlich höher als beim Aitken-Verfahren. Werden dagegen beim Grob-Gitter-Gradienten genauso viele oder sogar mehr Iterationen als mit dem Aitken-Verfahren benötigt, wird mit dem Aitken-Verfahren immer die geringste Rechenzeit erreicht.

Bewertung

Wie in den beiden vorangegangenen Abschnitten gezeigt werden konnte, ist es möglich mithilfe einer Lösung des gekoppelten Problems auf einer gröberen Diskretisierung den Rechenaufwand für ein iterativ gestaffeltes Lösungsverfahren deutlich zu reduzieren. Die vorgestellten Ideen lassen sich auch auf andere Lösungsalgorithmen übertragen. So ist es z.B. auch möglich die Finite Differenz zur Approximation der Jacobi-Matrix bei einem Newton-Krylow-Verfahren (Abschnitt 5.6.3) auf dem groben Gitter zu berechnen und so erheblich Rechenzeit zu sparen.

Es gibt jedoch auch Einschränkungen in der Anwendung der Zwei-Level-Verfahren. So hat z.B. der Unterschied in der Feinheit der beiden Diskretisierungen einen starken Einfluss auf den Effizienzgewinn. Besteht die gröbere Diskretisierung aus ähnlich vielen

Elementen wie die feine Diskretisierung, wird für die zusätzlichen Grob-Gitter-Iterationen mehr Zeit benötigt, als auf der feinen Diskretisierung eingespart werden kann. Werden dagegen zu wenige Grob-Gitter-Elemente verwendet, d.h. wird der Prädiktor auf einer zu groben Diskretisierung berechnet, stellt er eine zu schlechte Approximation der Fein-Gitter-Lösung dar und beschleunigt die Iteration auf der feinen Diskretisierung nicht ausreichend.

5.6.3 Newton-Krylow-Verfahren

Durch die Anwendung des Newton-Raphson-Verfahrens auf das gekoppelte Problem soll eine höhere Konvergenzordnung für die Iteration über die beteiligten Felder erreicht werden. Ausgangspunkt für die Lösung des gekoppelten Problems mit der Newton-Raphson-Methode ist die Formulierung als Nullstellenproblem, wie sie in Gleichung (5.12) bereits angegeben wurde. Eine Nullstelle $\mathbf{d}_\Gamma^*$ des Interface-Kompatibilitäts-Operators $\mathcal{L}(\mathbf{d}_\Gamma)$ entspricht dabei der Lösung des gekoppelten Problems.

$$\mathcal{L}(\mathbf{d}_\Gamma^*) = \mathbf{d}_\Gamma^* - \mathcal{T}(\mathbf{d}_\Gamma^*) = 0 \tag{5.26}$$

Die Anwendung des mehrdimensionalen Newton-Raphson-Verfahrens, wie es in Abschnitt 2.1.5 beschrieben worden ist, führt über die Taylorreihenentwicklung des Interface-Kompatibilitäts-Operators $\mathcal{L}(\mathbf{d}_\Gamma)$ um den aktuellen Verschiebungszustand des Kopplungsrands $\mathbf{d}_\Gamma^{(k)}$ auf ein lineares Gleichungssystem zur Bestimmung des Verschiebungsinkrements:

$$\mathcal{L}(\mathbf{d}_\Gamma^{(k+1)}) = \mathcal{L}(\mathbf{d}_\Gamma^{(k)}) + \mathcal{L}'(\mathbf{d}_\Gamma^{(k)})\ \Delta\mathbf{d}_\Gamma^{(k+1)} = \mathbf{0}. \tag{5.27}$$

Die Verschiebungen am Kopplungsrand werden mit diesen Inkrementen solange verbessert

$$\mathbf{d}_\Gamma^{(k+1)} = \mathbf{d}_\Gamma^{(k)} + \Delta\mathbf{d}_\Gamma^{(k+1)}, \tag{5.28}$$

bis eine Norm des Residuums $\mathcal{L}(\mathbf{d}_\Gamma^{(k+1)})$ kleiner als eine vorgegebene Schranke ε ist. Die Tangente $\mathcal{L}'(\mathbf{d}_\Gamma^{(k)})$ bezeichnet dabei die Ableitung des Interface-Kompatibilitäts-Operators nach den aktuellen Verschiebungen am Kopplungsrand $\mathbf{d}_\Gamma^{(k)}$ und wird im Allgemeinen als Jacobi-Matrix bezeichnet:

$$\mathcal{L}'(\mathbf{d}_\Gamma^{(k)}) = \frac{\partial\mathcal{L}(\mathbf{d}_\Gamma^{(k)})}{\partial\mathbf{d}_\Gamma^{(k)}}. \tag{5.29}$$

Aufgrund starker Nichtlinearitäten ist es relativ aufwändig die Jacobi-Matrix explizit zu bestimmen. In ausgeschriebener Form wird deutlich, dass der Interface-Kompatibilitäts-Operator neben den Nichtlinearitäten des Fluid-Operators (Konvektion, Stabilisierung) und des Struktur-Operators (große Verschiebungen) zusätzliche nichtlineare Terme aus der Verschiebung des Fluidgebiets enthält:

$$\mathcal{L} = \mathbf{d}_\Gamma - \mathcal{S}\Big(\mathcal{F}\big(\mathbf{u}_\Gamma, \mathcal{N}(\mathbf{d}_\Gamma)\big)\Big). \tag{5.30}$$

Zur Lösung des linearen Gleichungssystems für $\Delta\mathbf{d}_\Gamma^{(k+1)}$

$$\mathcal{L}'(\mathbf{d}_\Gamma^{(k)})\ \Delta\mathbf{d}_\Gamma^{(k+1)} = -\mathcal{L}(\mathbf{d}_\Gamma^{(k)}) \tag{5.31}$$

kann ein iteratives Lösungsverfahren verwendet werden. Diese innere Iteration mit dem Zähler l wird abgebrochen, wenn eine Norm des Residuums des Gleichungssystems (5.31) kleiner als eine vorgeschrieben Schranke ε ist:

$$||\mathcal{R}_{\Delta\mathbf{d}}^{(l+1)}|| \leq \varepsilon, \qquad \mathcal{R}_{\Delta\mathbf{d}}^{(l+1)} = \mathcal{L}(\mathbf{d}_\Gamma^{(k)}) + \mathcal{L}'(\mathbf{d}_\Gamma^{(k)})\ \big(\Delta\mathbf{d}_\Gamma^{(k+1)}\big)^{(l+1)}. \tag{5.32}$$

Wird für diese Iteration ein Krylow-Unterraum-Verfahren verwendet, ist es nicht erforderlich die Jacobi-Matrix explizit aufzustellen. Es genügt, die Wirkung der Jacobi-Matrix auf einen Vektor $\mathbf{z}$ in Form des Matrix-Vektor-Produkts $\mathcal{L}'(\mathbf{d}_\Gamma^{(k)})\mathbf{z}$ auszuwerten.

Der komplette Ablauf des Newton-Krylow-Verfahrens ist in Abbildung 5.14 dargestellt. Dabei wurden die Details der inneren Iteration ausgespart und nur die iterative Verbesserung von $\big(\Delta\mathbf{d}_\Gamma^{(k+1)}\big)^{(l+1)}$ durch ein Krylow-Verfahren angedeutet.

Basierend auf dem vorgestellten Ansatz, das gekoppelte Problem mit dem Newton-Raphson-Verfahren zu lösen, wurde eine Reihe von Verfahren entwickelt, die sich hauptsächlich in der Behandlung der Jacobi-Matrix unterscheiden. Bei den meisten der heute verwendeten Verfahren wird die Jacobi-Matrix entweder approximiert aufgestellt oder nur implizit verwendet. Bei einer Approximation geht dabei im Vergleich zu einer exakten Jacobi-Matrix die quadratische Konvergenz verloren. Die gebräuchlichsten Methoden werden in den folgenden Abschnitten kurz vorgestellt.

Exakte Jacobi-Matrix

Trotz des großen Aufwands, die Jacobi-Matrix des Interface-Kompatibilitäts-Operators explizit aufzustellen, wird dieses Vorgehen u.a. von Dettmer (2004), Fernández und Moubachir (2005) und Barcelos u. a. (2006) verwendet. Das resultierende Verfahren wird von allen genannten Autoren als sehr robust und effizient beschrieben; durch eine konsistente Linearisierung bleibt die quadratische Konvergenz des Newton-Raphson-

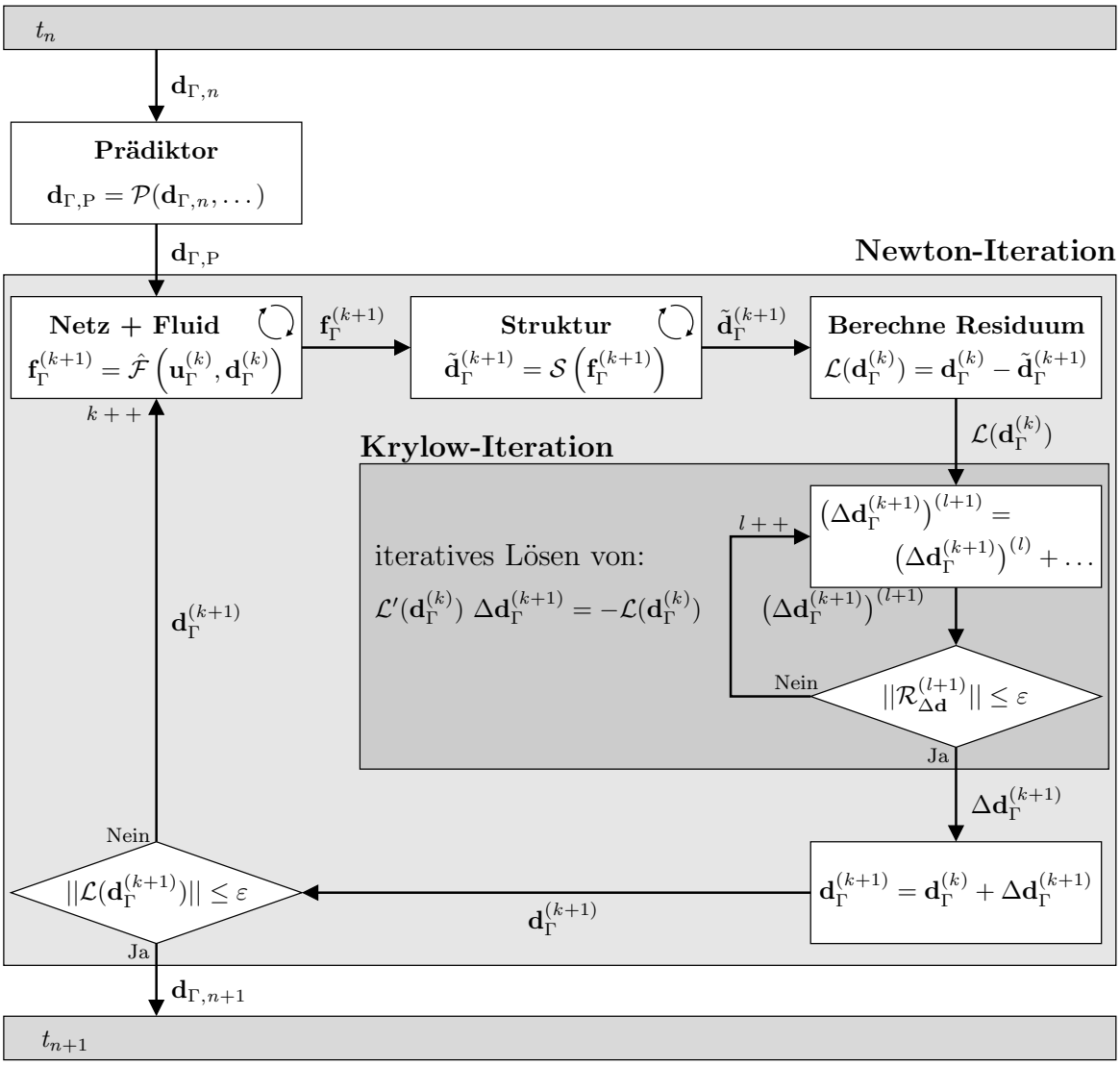


Abbildung 5.14: Ablaufdiagramm zum Newton-Krylow-Verfahren.

Verfahrens erhalten. Dadurch konvergiert die Newton-Raphson-Iteration in den meisten Fällen in weniger Schritten als z.B. ein Block-Gauß-Seidel-Verfahren mit Relaxation. Fernández und Moubachir (2005) stellen in diesem Zusammenhang fest, dass die Genauigkeit der Jacobi-Matrix einen „drastischen“ Einfluss auf die Konvergenz der Newton-Raphson-Iteration hat.

Jedoch merken die Autoren an, dass einerseits die Herleitung und Implementierung der Newton-Krylow-Verfahren mit exakter Jacobi-Matrix sehr langwierig ist (Dettmer und Perić 2006), andererseits auch der numerische Aufwand in jeder Iteration so groß ist, dass für schwach gekoppelte Probleme die Rechenzeiten länger sind als mit Block-Gauß-Seidel-Verfahren (Barcelos u. a. 2006).

Diagonale Jacobi-Matrix

Werden die Nebendiagonalblöcke der Jacobi-Matrix, d.h. die Ableitung des Fluid-Operators nach den Struktur-Variablen und umgekehrt, zu null gesetzt, vereinfacht sich das Matrix-Vektor-Produkt zu $\mathcal{L}'(\mathbf{d}_\Gamma^{(k)})\mathbf{z} = \mathbf{z}$ (Schmidberger 2006). Für die Bestimmung des Verschiebungsinkrements mit dem Newton-Raphson-Verfahren (vgl. Gl. (5.27)) folgt daraus:

$$\begin{aligned} \mathcal{L}'(\mathbf{d}_\Gamma^{(k)})\ \Delta\mathbf{d}_\Gamma^{(k+1)} &= -\mathcal{L}(\mathbf{d}_\Gamma^{(k)}) \\ \Delta\mathbf{d}_\Gamma^{(k+1)} &= \mathcal{S}\Big(\mathcal{F}\big(\mathbf{u}_\Gamma^{(k)}, \mathcal{N}(\mathbf{d}_\Gamma^{(k)})\big)\Big) - \mathbf{d}_\Gamma^{(k)} \\ \mathbf{d}_\Gamma^{(k+1)} &= \mathcal{S}\Big(\mathcal{F}\big(\mathbf{u}_\Gamma^{(k)}, \mathcal{N}(\mathbf{d}_\Gamma^{(k)})\big)\Big). \end{aligned} \tag{5.33}$$

Dies entspricht der Fixpunkt-Iterations-Vorschrift (5.10) und damit auch dem Block-Gauß-Seidel-Verfahren. Das Block-Gauß-Seidel-Verfahren kann also auch als Newton-Krylow-Verfahren interpretiert werden, bei dem die Kopplungsterme, d.h. die Nebendiagonalblöcke, vernachlässigt werden.

Approximation durch Finite Differenzen

Eine weitere Möglichkeit das Aufstellen der exakten Jacobi-Matrix zu vermeiden, ist die Approximation des Matrix-Vektor-Produkts $\mathcal{L}'(\mathbf{d}_\Gamma^{(k)})\mathbf{z}$ durch Finite Differenzen erster (Matthies und Steindorf 2003) oder zweiter Ordnung:

$$\mathcal{L}'(\mathbf{d}_\Gamma^{(k)})\mathbf{z} \approx \frac{\mathcal{L}(\mathbf{d}_\Gamma^{(k)} + \kappa\mathbf{z}) - \mathcal{L}(\mathbf{d}_\Gamma^{(k)})}{\kappa} = \frac{\mathcal{T}(\mathbf{d}_\Gamma^{(k)}) - \mathcal{T}(\mathbf{d}_\Gamma^{(k)} + \kappa\mathbf{z}) + \kappa\mathbf{z}}{\kappa} \tag{5.34}$$

$$\mathcal{L}'(\mathbf{d}_\Gamma^{(k)})\mathbf{z} \approx \frac{\mathcal{L}(\mathbf{d}_\Gamma^{(k)} + \kappa\mathbf{z}) - \mathcal{L}(\mathbf{d}_\Gamma^{(k)} - \kappa\mathbf{z})}{2\kappa}. \tag{5.35}$$

Bei der Implementierung der Finiten Differenz erster Ordnung (5.34) kann berücksichtigt werden, dass der Ausdruck $\mathcal{T}(\mathbf{d}_\Gamma^{(k)})$ bereits für die Berechnung des Residuums im linearen Gleichungssystem (5.27) ausgewertet wurde. Für die Bestimmung des zweiten Ausdrucks $\mathcal{T}(\mathbf{d}_\Gamma^{(k)} + \kappa\mathbf{z})$ ist eine weitere komplette Berechnung des Fluid-Struktur-Operators erforderlich, die eine vollständige nichtlineare Lösung des Fluid- und des Strukturfelds erfordert. Dieser recht hohe Aufwand, der in jeder Iteration des Krylow-Verfahrens zur Lösung des linearen Gleichungssystems in der Newton-Iteration erforderlich ist, kann nach Schmidberger (2006) reduziert werden, indem bei der Lösung des Fluids nur ein Schritt der nichtlinearen Iteration berechnet wird. Die ungenauere Approximation

verschlechtert die Konvergenz des Newton-Krylow-Verfahrens, jedoch werden mit dieser „beschleunigten Finiten Differenz“ geringere Rechenzeiten erzielt.

Die Länge der Finiten Differenz κ muss in der Regel vom Anwender vorgegeben werden. Die Wahl ist stark problemabhängig und hat einen sehr großen Einfluss auf die Konvergenzgeschwindigkeit des Krylow-Verfahrens und damit auf die Effektivität der gesamten Methode.

Steindorf (2002) verwendet die Approximation durch Finite Differenzen erster Ordnung für die einzelnen Blöcke der Jacobi-Matrix und zeigt, dass das approximative Newton-Krylow-Verfahren seine quadratische Konvergenzeigenschaft behält, sofern die Einzelfeldlöser quadratisch konvergieren.

Approximation mit reduziertem Fluid-Problem

Gerbeau und Vidrascu (2003) haben ein Verfahren vorgestellt, bei dem sie versuchen, die für die Kopplung entscheidenden Aspekte des Fluids, d.h. den „added mass“-Effekt, in der Jacobi-Matrix zu berücksichtigen, aber gleichzeitig die komplizierten Nichtlinearitäten zu vernachlässigen. Sie verwenden zum Aufstellen der Jacobi-Matrix ein reduziertes, lineares Fluidmodell, in dem sie die konvektiven Anteile vernachlässigen und ein reibungsfreies, gewichtsloses Fluid annehmen. Außerdem setzen sie voraus, dass das Fluidgebiet in seiner aktuellen Lage ortsfest ist und die Struktur in ihrem aktuellen Zustand linearisiert wird. Nach Elimination der Geschwindigkeiten $\mathbf{u}$ muss nur noch die skalare Poisson-Gleichung auf dem Fluidgebiet gelöst werden.

Deparis (2004) stellt ein ähnlich reduziertes Modell vor, bei dem er auch die Lage des Fluidgebiets festhält, aber Struktur und das vollständige Fluidmodell im aktuellen Zustand linearisiert. Die so aufgestellte Approximation der Jacobi-Matrix ist genauer als im vorherigen Modell, erfordert aber auch einen größeren numerischen Aufwand. Vergleichend stellt Deparis (2004) fest, dass das reduzierte Modell von Gerbeau und Vidrascu (2003) eine gröbere Approximation der Tangente darstellt und dadurch mehr Newton-Iterationen benötigt, die aber jeweils so schnell sind, dass in allen Fällen dieses Verfahren weniger Rechenzeit benötigt. Mit der genaueren Jacobi-Matrix jedoch kann das Verfahren in manchen Fällen robuster sein und zu besseren Ergebnissen führen.

Nach Gerbeau und Vidrascu (2003) eignen sich diese Ansätze sehr gut, um die Jacobi-Matrix des Fluid-Struktur-Problems zu approximieren. Die Verfahren haben sich bei allen untersuchten Problemstellungen sehr robust verhalten und waren effizient. Jedoch eignen sie sich nur bedingt für den Einsatz mit modularen Einfeldlösern. Es können zwar vorhandene und bewährte Löser für Fluid und Struktur bei der Berechnung des Resi-

duums eingesetzt werden, jedoch ist ein weiterer Löser erforderlich, der z.B. die skalare Poissongleichung auf dem aktuellen Fluidgebiet löst.

Approximation mit konstruiertem Fluid-Modell

Einen etwas anderen Ansatz, bei dem keinerlei Eingriff in die verwendeten Fluid- und Struktur-Löser erforderlich ist, wählt Vierendeels (2005, 2006). Der Autor verwendet die Reaktionen auf Druck- und Verschiebungsmoden aus vorangegangenen Iterationen, um ein „reduced order"-Modell für Fluid und Struktur aufzubauen. Die Jacobi-Matrizen dieses reduzierten Modells werden dann verwendet, um die implizite Kopplung der partitionierten Löser herzustellen. Zu Beginn werden in jedem Zeitschritt zwei Block-Gauß-Seidel-Iterationen benötigt, bevor das reduzierte Modell aus dem ersten Mode konstruiert werden kann.

Nach Vierendeels u. a. (2007) lässt sich dieses Verfahren auf der Basis von vorhandenen Einzelfeldlösern sehr leicht umsetzen. Für die im genannten Artikel vorgestellten Beispiele zeigt die Methode ein gutes Konvergenzverhalten, vergleichbar mit anderen Newton-Krylow-Verfahren.

Jacobi-freies Verfahren

Ein Verfahren, das ohne eine direkte Approximation der Jacobi-Matrix auskommt haben Michler u. a. (2005) vorgestellt. Die allgemeine Theorie der Jacobi-freien Newton-Krylow-Verfahren ist u.a. in Knoll und Keyes (2004) zu finden.

Um Informationen aus den vorangegangenen Iterationen wiederzuverwenden, wenden Michler u. a. (2005) Fixpunkt-Verfahren als Vorkonditionierer für ein Newton-Krylow-Verfahren (GMRES) an. Dabei kann die GMRES-Beschleunigung auf Freiheitsgrade am Kopplungsrand beschränkt werden, was laut der Autoren effizienter sein soll als die Anwendung eines Newton-Krylow-Verfahrens auf das gesamte monolithische System. Außerdem kann durch die Wiederverwendung von Krylow-Vektoren in nachfolgenden Newton-Iterationen oder auch Zeitschritten erheblich Rechenaufwand eingespart werden (Michler 2005). Dies geht jedoch auf Kosten der Robustheit des Verfahrens. Durch eine Fehler-Vergrößerungs-Analyse haben Michler u. a. (2006) gezeigt, dass das Verfahren mit einer Wiederverwendung der Krylow-Vektoren in manchen Fällen divergiert. Bei Verzicht auf die Wiederverwendung zeigt die Methode monotone Konvergenz.

Bewertung

Im Allgemeinen zeichnet die Newton-Krylow-Verfahren eine schnellere Konvergenz als bei den Block-Iterations-Verfahren aus. Dabei ist aber jeder einzelne Iterationsschritt numerisch aufwändiger, da zusätzlich zur Berechnung des Residuums die Jacobi-Matrix aufgestellt bzw. ausgewertet und ein lineares Gleichungssystem gelöst werden muss. Für schwach gekoppelte Probleme, bei denen auch Block-Iterations-Verfahren nur eine geringe Anzahl an Iterationen benötigen, lohnt sich der Einsatz von Newton-Krylow-Verfahren nicht, da die in diesen Fällen nur geringe Reduktion der benötigten Iterationen den höheren Aufwand pro Iteration nicht ausgleichen kann. Erst bei physikalisch stark gekoppelten Problemstellungen führt die schnellere Konvergenz auf eine insgesamt geringere Rechenzeit.

5.6.4 Weitere Kopplungsverfahren

Semi-Implizite Verfahren

Bei den semi-impliziten Verfahren werden die Fluidgleichungen mit einem Projektions-Verfahren gelöst, bei dem die Drucklösung vom restlichen Operator entkoppelt wird. Dadurch ist es bei der Kopplung mit der Struktur möglich, den viskosen und konvektiven Anteil des Fluids nur einmal pro Zeitschritt zu lösen, den Druck aber iterativ zu behandeln bis Konvergenz zwischen Fluid und Struktur erreicht wird (Badia und Codina 2007). Durch die implizite Kopplung des Drucks wird die „artificial added mass“-Instabilität vermieden. Fernández u. a. (2007) haben die bedingte Stabilität des Verfahrens bewiesen.

Robin-Kopplungs-Bedingungen

Badia u. a. (2008) schlagen eine neue Klasse von Verfahren vor, die auf einer Robin-Partitionierung anstelle der klassischen Dirichlet-Neumann-Partitionierung basieren. Dabei wird das partitionierte System mit einem relaxierten Block-Gauß-Seidel-Verfahren gelöst, wobei auf die Einzelfeld-Löser Robin-Randbedingungen (Linearkombinationen aus Dirichlet- und Neumann-Randbedingungen) aufgebracht werden. Für bestimmte Werte der Koeffizienten der Robin-Randbedingung ergibt sich z.B. die Dirichlet-Neumann-Partitionierung als Sonderfall der Robin-Verfahren. Nach Aussage der Autoren konvergiert das Robin-Neumann-Verfahren in allen Fällen auch ohne Relaxation und unabhängig vom „added mass“-Effekt. Jedoch ist die Konvergenz stark von der Wahl der Koeffizienten in der Robin-Randbedingung abhängig.

5.7 Vergleich der Kopplungsverfahren

In diesem Abschnitt wird eine Auswahl der vorgestellten Verfahren hinsichtlich ihrer Konvergenzgeschwindigkeit und der benötigten Rechenzeit verglichen. Da für den Anwender die Rechenzeit (und damit auch die Kosten für die Hardware-Nutzung) neben der Stabilität und Genauigkeit eines Verfahrens bei der Entscheidung, ob eine Simulation durchführbar ist, eine wichtige Rolle spielt, sollte diese bei entsprechenden Vergleichen neben der Iterationsanzahl immer mitberücksichtigt werden.

Im Folgenden werden die ersten 20 Zeitschritte der numerischen Simulation des in Abschnitt 5.4 beschriebenen Problems für verschiedene Massendichten der Struktur mit unterschiedlichen Verfahren berechnet. Dabei wird aus den drei großen Gruppen der einfach gestaffelten Verfahren, Block-Iterations-Verfahren und Newton-Krylow-Verfahren jeweils eine etablierte Methode ausgewählt und zusätzlich ein Verfahren mit Grob-Gitter-Prädiktor berücksichtigt. Verglichen werden für diese Verfahren die durchschnittlich benötigten Iterationen und Rechenzeiten zur Lösung eines Zeitschritts. Alle Berechnungen in diesem Abschnitt wurden auf einem Athlon 64 X2 4800+ Prozessor mit 2,5 GHz durchgeführt.

5.7.1 Sequenziell gestaffeltes Verfahren

Aus der Klasse der einfach gestaffelten Verfahren wird das sequenziell gestaffelte Verfahren (siehe Abschnitt 5.5.2) ausgewählt, da es durch den zusätzlichen Austausch von Kopplungsinformationen genauere Ergebnisse liefert als das parallel gestaffelte Verfahren.

Da bei den einfach gestaffelten Verfahren keine Iteration über die gekoppelten Felder ausgeführt wird, sondern jedes Feld nur einmal gelöst wird, ist die Konvergenzgeschwindigkeit und auch die Rechenzeit kein aussagekräftiges Kriterium für einen Vergleich. Vielmehr sind die Genauigkeit des Verfahrens und die Frage, ob und wann eine Instabilität aufgrund des „artificial added mass“-Effekts auftritt, entscheidend. Für das gewählte Beispiel tritt für eine Massendichte der Struktur von $\rho_\mathrm{S} = 5{,}0\ \mathrm{kg/m^3}$, d.h. eine sehr starke Kopplung, die Instabilität bereits im dritten Zeitschritt und für $\rho_\mathrm{S} = 50{,}0\ \mathrm{kg/m^3}$ im siebten Zeitschritt auf. Nur für eine schwache Kopplung von Fluid und Struktur mit $\rho_\mathrm{S} = 500{,}0\ \mathrm{kg/m^3}$ lassen sich die ersten 20 Zeitschritte stabil berechnen. Dabei werden im Durchschnitt 19,75 s pro Zeitschritt für die Lösung der Netzverschiebung, des Fluids und der Struktur benötigt.

Wie sich in diesem einfachen Beispiel auch zeigt, sind einfach gestaffelte Verfahren bei der Kopplung von leichten Strukturen mit inkompressiblen Strömungen in den meisten

	$\rho_S = 5{,}0$ kg/m^3	$\rho_S = 50{,}0$ kg/m^3	$\rho_S = 500{,}0$ kg/m^3
Iterationen pro Zeitschritt	16,3	7,0	3,9
Rechenzeit pro Zeitschritt [s]	252,47	113,65	58,03

Tabelle 5.3: Durchschnittlich benötige Anzahl an Iterationen und Rechenzeit für einen Zeitschritt mit dem Block-Gauß-Seidel-Verfahren und Aitken-Relaxation.

Fällen instabil und werden daher, wie bereits im Abschnitt 5.5.4 beschrieben, in dieser Arbeit nicht verwendet.

5.7.2 Block-Gauß-Seidel-Verfahren

Das Block-Gauß-Seidel-Verfahren einschließlich Bestimmung des Relaxationsparameters mit der Aitken-Methode hat sich in vielen Untersuchungen im Vergleich zu anderen Block-Iterations-Verfahren oder anderen Relaxationsmethoden als schnellstes Verfahren durchgesetzt. Es konvergiert in diesem Beispiel auch für das ungünstigste Massenverhältnis von $\rho_S = 5{,}0$ kg/m^3, wobei jedoch der von Causin u. a. (2005) beschriebene Einfluss des „artificial added mass“-Effekts auf die Konvergenz deutlich zu erkennen ist (Tabelle 5.3). So werden für eine Massendichte von $\rho_S = 5{,}0$ kg/m^3 im Durchschnitt mehr als viermal so viele Iterationen benötigt wie für eine Dichte von $\rho_S = 500{,}0$ kg/m^3.

In der benötigten Rechenzeit zeigt sich der große Nachteil der iterativ gestaffelten Verfahren. Für stark gekoppelte Probleme benötigt das Block-Gauß-Seidel-Verfahren mehr als 13-mal so viel Zeit wie ein (nur für schwache Kopplung stabiles) einfach gestaffeltes Verfahren. Dieser sehr große numerische Aufwand, um die kinematische Kontinuität am Kopplungsrand sicherzustellen, ist eine wichtige Motivation für den Einsatz von stabilisierten, einfach gestaffelten Verfahren und die Entwicklung von effizienteren iterativen Verfahren.

5.7.3 Grob-Gitter-Prädiktor

Der Einsatz eines Grob-Gitter-Prädiktors ist eine Möglichkeit, den numerischen Aufwand der iterativ gestaffelten Verfahren zu reduzieren. Die Bereitstellung eines zweiten Netzes bereitet zwar bei der Implementierung einen zusätzlichen Aufwand, ist jedoch die zweite Diskretisierung aufgestellt, kann ein vorhandener Löser ohne weitere Anpassungen darauf angewendet werden. Für dieses Beispiel wurde auf beiden Gittern das Block-Gauß-Seidel-Verfahren mit Aitken-Relaxation angewendet, auf dem groben Netz zusätzlich ein Prädiktor zweiter Ordnung.

	$\rho_S = 5{,}0$ kg/m^3	$\rho_S = 50{,}0$ kg/m^3	$\rho_S = 500{,}0$ kg/m^3
Iterationen (grob) pro Zeitschritt	15,7	6,8	3,7
Iterationen (fein) pro Zeitschritt	10,6	5,9	3,5
Rechenzeit pro Zeitschritt [s]	173,75	92,37	54,11

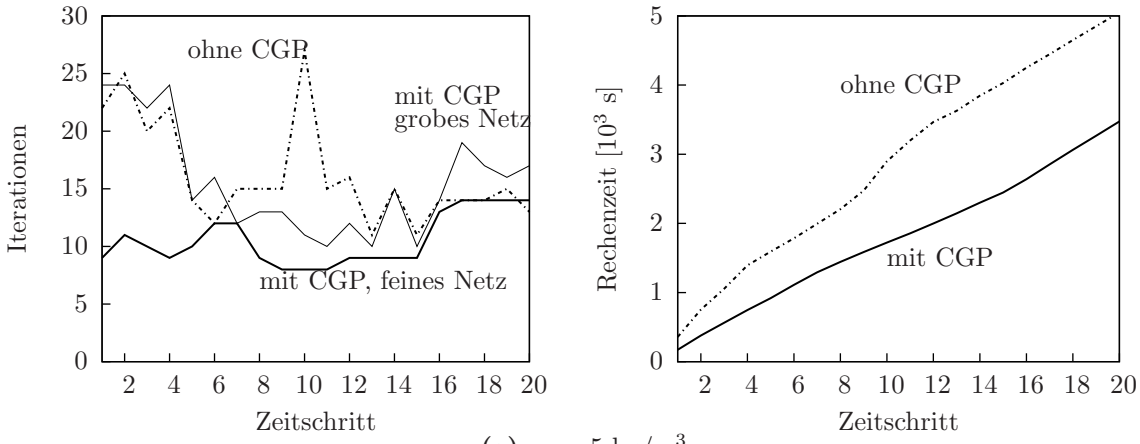
Tabelle 5.4: Durchschnittlich benötige Anzahl an Iterationen und Rechenzeit für einen Zeitschritt mit dem Block-Gauß-Seidel-Verfahren, Aitken-Relaxation und Grob-Gitter-Prädiktor.

In den gemittelten Ergebnissen in Tabelle 5.4 zeigt sich, dass auf dem groben Gitter ungefähr so viele Iterationen benötigt werden wie beim Block-Gauß-Seidel-Verfahren ohne Grob-Gitter-Prädiktor (Tabelle 5.3). Auf dem feinen Gitter jedoch werden aufgrund des sehr genauen Prädiktors deutlich weniger Iterationen benötigt. Da die Grob-Gitter-Iterationen vom numerischen Aufwand kaum ins Gewicht fallen, überträgt sich diese Einsparung direkt auf die Rechenzeit.

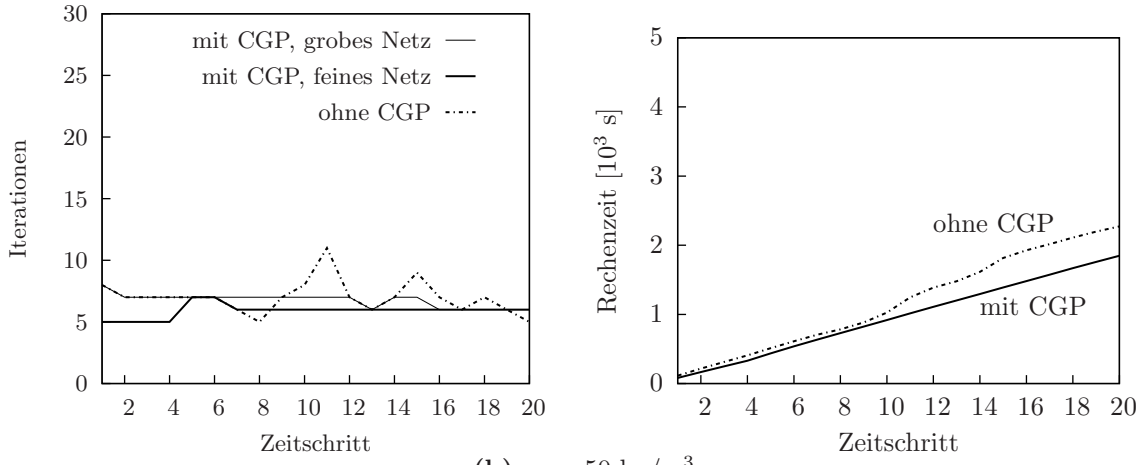
In Abbildung 5.15 sind die benötigten Iterationen und die aufsummierte Rechenzeit für die ersten 20 Zeitschritte für alle drei untersuchten Massendichten angegeben. Dabei bezeichnen die durchgezogenen Linien die Rechnung mit und die gestrichelten Linien die Rechnung ohne Grob-Gitter-Prädiktor (CGP). In den linken Diagrammen gibt die dünne Linie jeweils die benötigten Iterationen auf dem groben Gitter an.

In Abbildung 5.15 (a), d.h. für $\rho_S = 5{,}0$ kg/m^3, sind die Unterschiede in der Iterationsanzahl am größten. Man kann deutlich erkennen, dass mit Grob-Gitter-Prädiktor auf dem groben Netz (dünne Linie) ähnlich viele Iterationen benötigt werden wie ohne Grob-Gitter-Prädiktor (gestrichelte Linie). Auf dem feinen Gitter werden jedoch mit Grob-Gitter-Prädiktor (dicke, durchgezogene Linie) grundsätzlich weniger Iterationen benötigt als ohne Grob-Gitter-Prädiktor. Dieses Verhalten ist für die beiden anderen Massendichten gleich, ist aber aufgrund der geringen Unterschiede in der Anzahl der Iterationen nicht so deutlich zu erkennen.

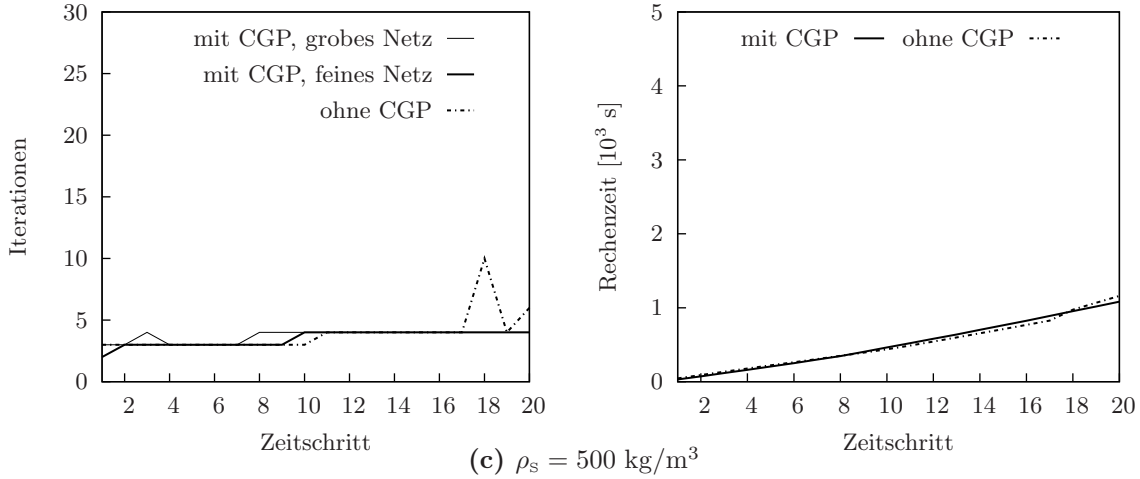
Die Einsparung in der Rechenzeit, die aus dieser Reduzierung der Iterationsanzahl folgt, lässt sich in den rechten Diagrammen ablesen. Hier ist die Gesamtrechenzeit, aufsummiert für die ersten n Zeitschritte, aufgetragen. Nach 20 Zeitschritten führt dies für $\rho_S = 5{,}0$ kg/m^3 auf eine Reduzierung der Rechenzeit um 31,2 %, für $\rho_S = 50{,}0$ kg/m^3 um 18,7 % und für $\rho_S = 500{,}0$ kg/m^3 noch um 6,8 %. Positiv zu bemerken ist dabei, dass die Einsparung größer wird je stärker die Kopplung des Problems ist, d.h. insbesondere in den Fällen, in denen iterativ gestaffelte Verfahren einen besonders hohen numerischen Aufwand haben, kann die Rechenzeit durch die Verwendung eines Grob-Gitter-Prädiktors erheblich reduziert werden.



(a) $\rho_\mathrm{s} = 5\ \mathrm{kg/m^3}$



(b) $\rho_\mathrm{s} = 50\ \mathrm{kg/m^3}$



(c) $\rho_\mathrm{s} = 500\ \mathrm{kg/m^3}$

Abbildung 5.15: Vergleich der benötigten Iterationen und aufsummierten Rechenzeit in den ersten 20 Zeitschritten für das Block-Gauß-Seidel-Verfahren mit und ohne Grob-Gitter-Prädiktor.

	$\rho_S = 5{,}0$ kg/m^3	$\rho_S = 50{,}0$ kg/m^3	$\rho_S = 500{,}0$ kg/m^3
Iterationen pro Zeitschritt	3,3	2,9	2
Rechenzeit pro Zeitschritt [s]	258,78	144,62	90,93

Tabelle 5.5: Durchschnittlich benötige Anzahl an Iterationen und Rechenzeit für einen Zeitschritt mit dem Newton-Krylow-Verfahren mit Finiter Differenz.

5.7.4 Newton-Krylow-Verfahren mit Finiter Differenz

Mit den vorher beschriebenen Verfahren wird abschließend ein Newton-Krylow-Verfahren verglichen, bei dem das Matrix-Vektor-Produkt mit der Jacobi-Matrix durch eine Finite Differenz erster Ordnung approximiert wird. Dabei wurde die von Schmidberger (2006) vorgeschlagene „beschleunigte Finite Differenz“ verwendet, bei der für die Berechnung der Finiten Differenz die Fluidlösung nicht auskonvergiert wird. Dadurch lassen sich in der Regel geringere Rechenzeiten erzielen, obwohl die Konvergenz etwas langsamer wird.

Die Iterationsanzahlen in Tabelle 5.5 zeigen deutlich die hervorragende Konvergenz der Newton-Krylow-Verfahren. Selbst für den stark gekoppelten Fall mit $\rho_S = 5{,}0$ kg/m^3 werden im Durchschnitt nicht viel mehr als drei Iterationen benötigt. In der Rechenzeit lässt sich aber auch der Nachteil der Methode erkennen. Selbst bei nur 3,3 Iterationen braucht das Newton-Krylow-Verfahren genauso lange zum Lösen eines Zeitschritts wie das Block-Gauß-Seidel-Verfahren bei 16,3 Iterationen. Für die weniger stark gekoppelten Fälle benötigt das Newton-Krylow-Verfahren sogar deutlich mehr Rechenzeit als das Block-Gauß-Seidel-Verfahren, da die Iterationsanzahl nicht so stark reduziert werden kann. Selbst wenn berücksichtigt wird, dass die hier zu Vergleichszwecken verwendete Implementierung des Newton-Krylow-Verfahrens nicht auf die Rechenzeit optimiert wurde und andere Varianten als die Approximation mit einer Finiten Differenz teilweise noch effizienter sind, kann abschließend gesagt werden, dass sich der Einsatz von Newton-Krylow-Verfahren zur partitionierten Lösung von gekoppelten Problemen bei schwach gekoppelten Problemen in der Regel nicht lohnt. Erst bei einer starken Kopplung kann im Vergleich zu einem Block-Gauß-Seidel-Verfahren Rechenzeit eingespart werden.

6

Numerische Beispiele

Im abschließenden Kapitel dieser Arbeit werden anhand von drei numerischen Beispielen die Einsatzmöglichkeiten einer effizienten Implementierung zur Lösung von Fluid-Struktur-Wechselwirkungs-Problemen exemplarisch gezeigt. Dafür wurden dreidimensionale Problemstellungen ausgesucht, bei denen die flexible Struktur vom Fluid umströmt wird und daher in der Simulation ein relativ großes Fluidgebiet benötigt wird, um alle Strömungseffekte im Umfeld und Nachlauf der Struktur zu erfassen. Wird dieses Gebiet so fein diskretisiert, dass alle Gradienten, besonders im Randschichtbereich, ausreichend genau abgebildet werden können, ergeben sich bis zu einer Million und mehr Freiheitsgrade. Um das transiente Verhalten von Fluid-Struktur-Wechselwirkungs-Problemen abzubilden, muss das aus der Diskretisierung des Fluids stammende große Gleichungssystem in mehreren Hundert bis Tausend Zeitschritten wiederholt aufgestellt und gelöst werden. Aufgabenstellungen dieser Größenordnung lassen sich auf gewöhnlichen Computern nicht mehr lösen und gehören in den Bereich des Hochleistungsrechnens.

Die im Folgenden vorgestellten Beispiele wurden alle auf dem parallelen Vektorcomputer NEC SX-8 am Höchstleistungsrechenzentrum der Universität Stuttgart berechnet. Dabei wurden pro Rechnung vier Knoten, d.h. insgesamt 32 Prozessoren verwendet und maximal 48 Stunden „wall-clock time“ (Echtzeit).

6.1 Gebäude mit starrem Dach

Die ersten beiden Beispiele sind einer Problemstellung von Hübner (2003) nachempfunden, die auch schon von Wüchner (2007) aufgegriffen worden ist. In diesen Arbeiten wird eine zweidimensionale Repräsentation, d.h. ein Längsschnitt eines quaderförmigen

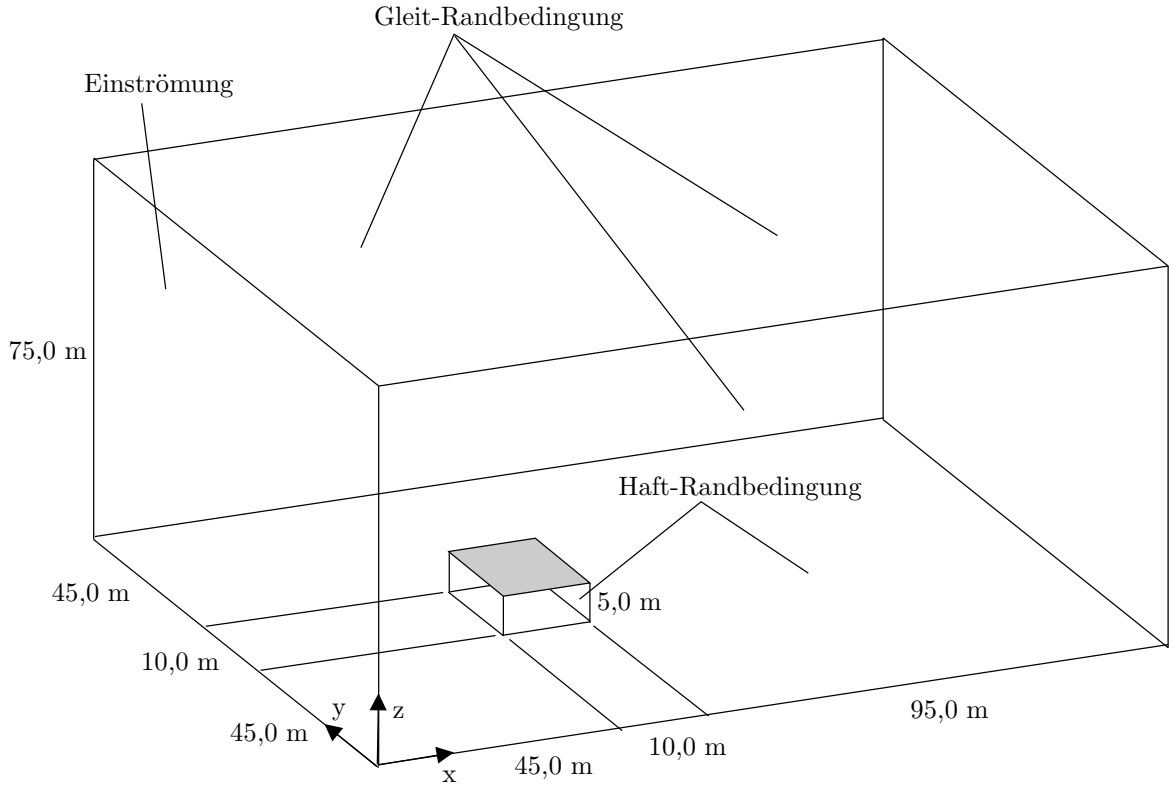


Abbildung 6.1: Gebäude mit starrem Dach: Geometrie und Randbedingungen.

Gebäudes mit Membrandach untersucht. Im Rahmen dieser Arbeit wird diese Problemstellung erweitert und in drei Dimensionen untersucht. Zunächst erfolgt in diesem Abschnitt eine Untersuchung des quaderförmigen Gebäudes mit einem starren Dach als reines Fluid-Problem. Im folgenden Abschnitt wird das flexible Membrandach und damit die Fluid-Struktur-Wechselwirkung berücksichtigt.

In Abbildung 6.1 ist das 150 m × 100 m × 75 m große Fluidgebiet dargestellt. Das 10 m × 10 m × 5 m große Gebäude wird durch eine Aussparung und entsprechende Randbedingungen berücksichtigt.

6.1.1 Diskretisierung

Die Materialparameter des Newton'schen Fluids werden gemäß Hübner (2003) gewählt:

$$\nu_{\mathrm{F}} = 0{,}08\ \mathrm{m}^2/\mathrm{s}, \qquad \rho_{\mathrm{F}} = 1{,}25\ \mathrm{kg}/\mathrm{m}^3. \tag{6.1}$$

Diskretisiert wird das Gebiet mit 8-knotigen Hexaeder-Elementen mit linearen Ansatzfunktionen für die Geschwindigkeiten und den Druck. Dabei wird eine SUPG/PSPG-

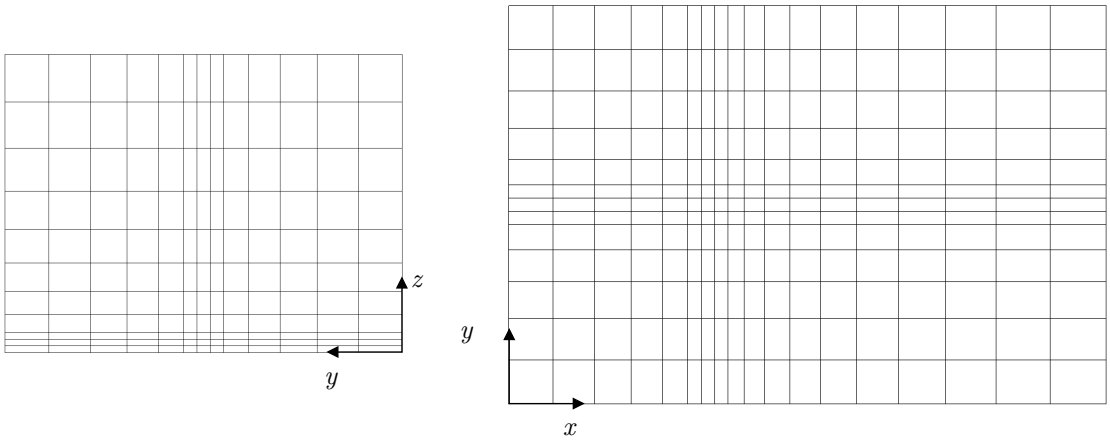


Abbildung 6.2: Gebäude mit starrem Dach: Grobes Netz in zwei Ansichten.

Stabilisierung verwendet, wie sie in Abschnitt 2.2.3 beschrieben ist. Um das erforderliche feine Netz erzeugen zu können, wird zunächst im Präprozessor ein grobes Netz mit 2.547 Elementen erzeugt. Dabei wird eine Konzentration der Elemente im Bereich des Gebäudes berücksichtigt, um dort auftretende Gradienten mit kleineren Elementen besser abbilden zu können. Das grobe Netz ist in zwei Ansichten in Abbildung 6.2 gezeigt. Auf dem Hochleistungsrechner wird diese grobe Diskretisierung in jede Richtung mit dem Faktor fünf verfeinert, um ein ausreichend feines Netz für die Strömungssimulation zu erhalten. Die Details zu beiden Diskretisierungen sind in Tabelle 6.1 angegeben.

Für die Diskretisierung in der Zeit wird das BDF2-Verfahren mit einem Zeitschritt $\Delta t = 0{,}01$ s verwendet.

6.1.2 Randbedingungen

Die Randbedingungen für die Strömung sind in Abbildung 6.1 dargestellt. Auf der Randfläche $x = 0{,}0$ m wird eine Einströmung für die Geschwindigkeit in x-Richtung vorgegeben:

$$\hat{u}_x(z,t) = 100{,}0 \text{ m/s} \cdot \hat{u}_t(t) \cdot \hat{u}_z(z) \tag{6.2}$$

Feld	Diskretisierung	Knoten	Elemente	Freiheitsgrade
Fluid	grob	3.180	2.547	
	fein	333.396	318.375	1.333.584

Tabelle 6.1: Gebäude mit starrem Dach: Diskretisierungen.

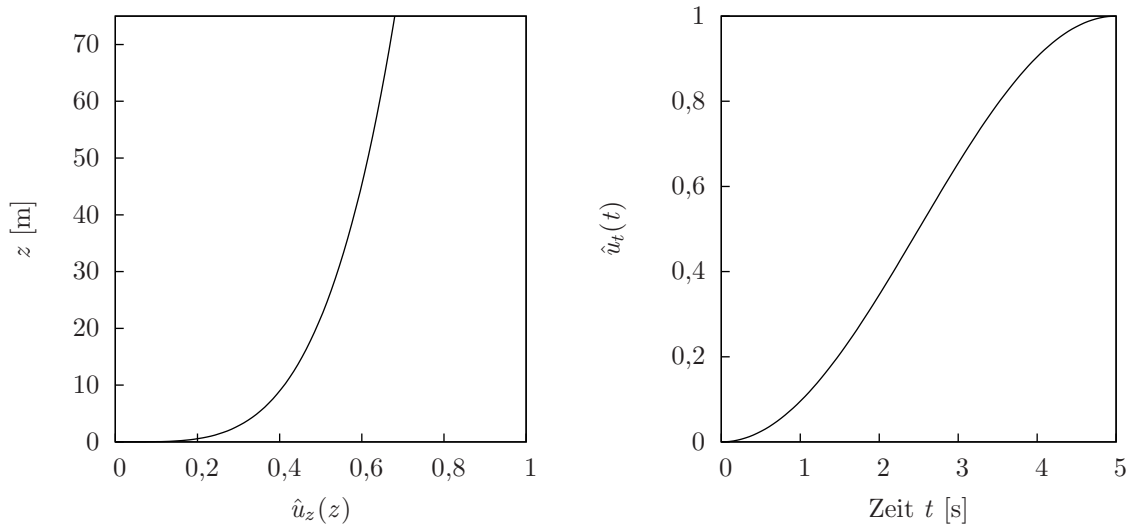


Abbildung 6.3: Gebäude mit starrem Dach: räumliche (links) und zeitliche (rechts) Veränderung der Einström-Randbedingung.

Diese variiert exponentiell in z-Richtung und sinusförmig in der Zeit:

$$\hat{u}_z(z) = \left(\frac{z}{350}\right)^{0,22} \qquad \hat{u}_t(t) = \left(\sin\left(\pi\left(\frac{t}{5,0} - 0,5\right)\right) + 1\right) \cdot 0,5 \tag{6.3}$$

Die Verläufe für die räumliche $\hat{u}_z(z)$ und zeitliche $\hat{u}_t(t)$ Veränderung der Einström-Randbedingung sind in Abbildung 6.3 gezeigt. Ab $t = 5{,}0$ s bleibt der Faktor für die zeitliche Veränderung konstant $\hat{u}_t(t) = 1{,}0$.

Die maximale Einströmgeschwindigkeit ergibt sich bei $z = 75{,}0$ m und ab $t = 5{,}0$ s zu 71,26 m/s. Mit der Länge des Gebäudes $l = 10$ m als Längenmaß folgt hieraus eine Reynoldszahl von Re ≈ 8900.

An der Unterseite des Fluidgebiets ($z = 0{,}0$ m) und an den fünf Flächen des Gebäudes wird eine Haft-Randbedingung („no-slip“) vorgegeben, während an den drei übrigen Längsseiten, die eine künstliche Begrenzung des Rechengebiets darstellen, Gleit-Randbedingungen („slip“) angenommen werden. Am Ausströmrand ($x = 150{,}0$ m) werden keine Randbedingungen („do-nothing“) vorgegeben.

6.1.3 Ergebnisse

Bei der Simulation der beschriebenen Problemstellung sind im Durchschnitt in jedem Zeitschritt vier Newton-Raphson-Iterationen zur Lösung des nichtlinearen Fluidproblems erforderlich. Auf 32 Prozessoren des SX-8-Vektorrechners können in 48 Stunden „wall-clock time“ ungefähr 1.500 Zeitschritte, d.h. 15 s Simulationszeit berechnet wer-

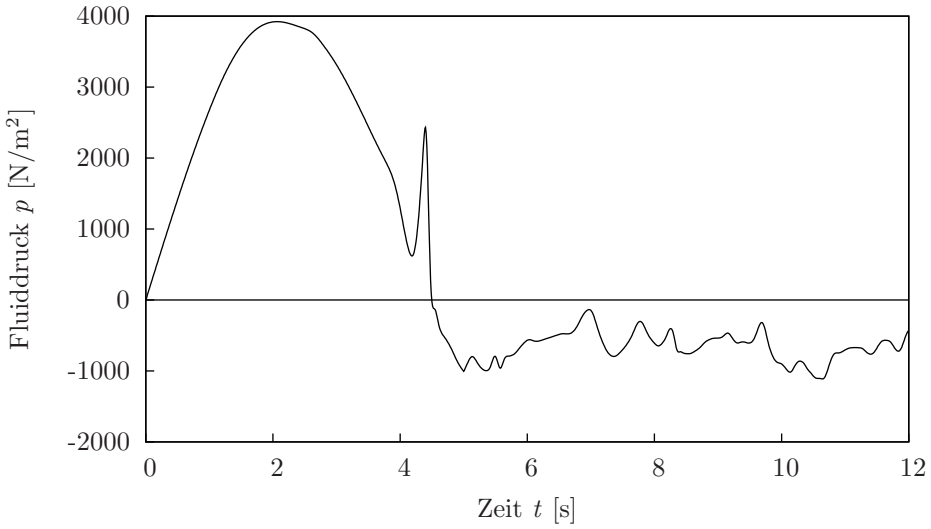


Abbildung 6.4: Gebäude mit starrem Dach: zeitliche Veränderung des Fluid-Drucks in Dachmitte ($\mathbf{x} = [50{,}0;\ 50{,}0;\ 5{,}0]^{\mathrm{T}}$ m).

den. Dabei wurde zur Lösung der linearen Gleichungssysteme ein GMRES-Verfahren mit der Unterraumgröße 80 und einer BILU-Vorkonditionierung verwendet.

Qualitativ zeigt diese Simulation dieselben Phänomene, wie von Hübner (2003) für den 2-D-Fall beschrieben. Da jedoch die Eingangsparameter an die dreidimensionale Betrachtungsweise angepasst werden mussten, weichen die Ergebnisse quantitativ von der Referenzlösung ab. In Abbildung 6.4 ist der Zeitverlauf des Fluiddrucks in der Mitte des starren Dachs aufgetragen. Während der ersten 5,0 s, in denen die Strömung von der Ruhelage aus bis zur maximalen Einström-Bedingung beschleunigt wird, wirkt ein positiver Druck von maximal ca. 4.000 N/m^2 auf das Membrandach. Ist die maximale Strömungsgeschwindigkeit bei 5,0 s erreicht, wirkt ein Sog auf das Dach, dessen Wert unregelmäßig zwischen ungefähr -500 und -1.000 N/m^2 schwankt.

Das Druckfeld in der Schnittebene $y = 50{,}0$ m in der Umgebung des Gebäudes (Abbildung 6.5) zeigt die Ursache für diesen Verlauf. Während an der linken Wand und im vorderen Bereich des Dachs das Druckfeld über die Zeit nahezu unveränderlich bleibt, lösen sich an der Hinterkante des Gebäudes Wirbel ab. Diese verweilen zunächst hinter dem Gebäude bis sie eine gewisse Größe erreicht haben und dann stromabwärts transportiert werden. Der Druckverlauf auf dem Gebäudedach wird durch diese Wirbel aber nur im rechten Bereich leicht beeinflusst.

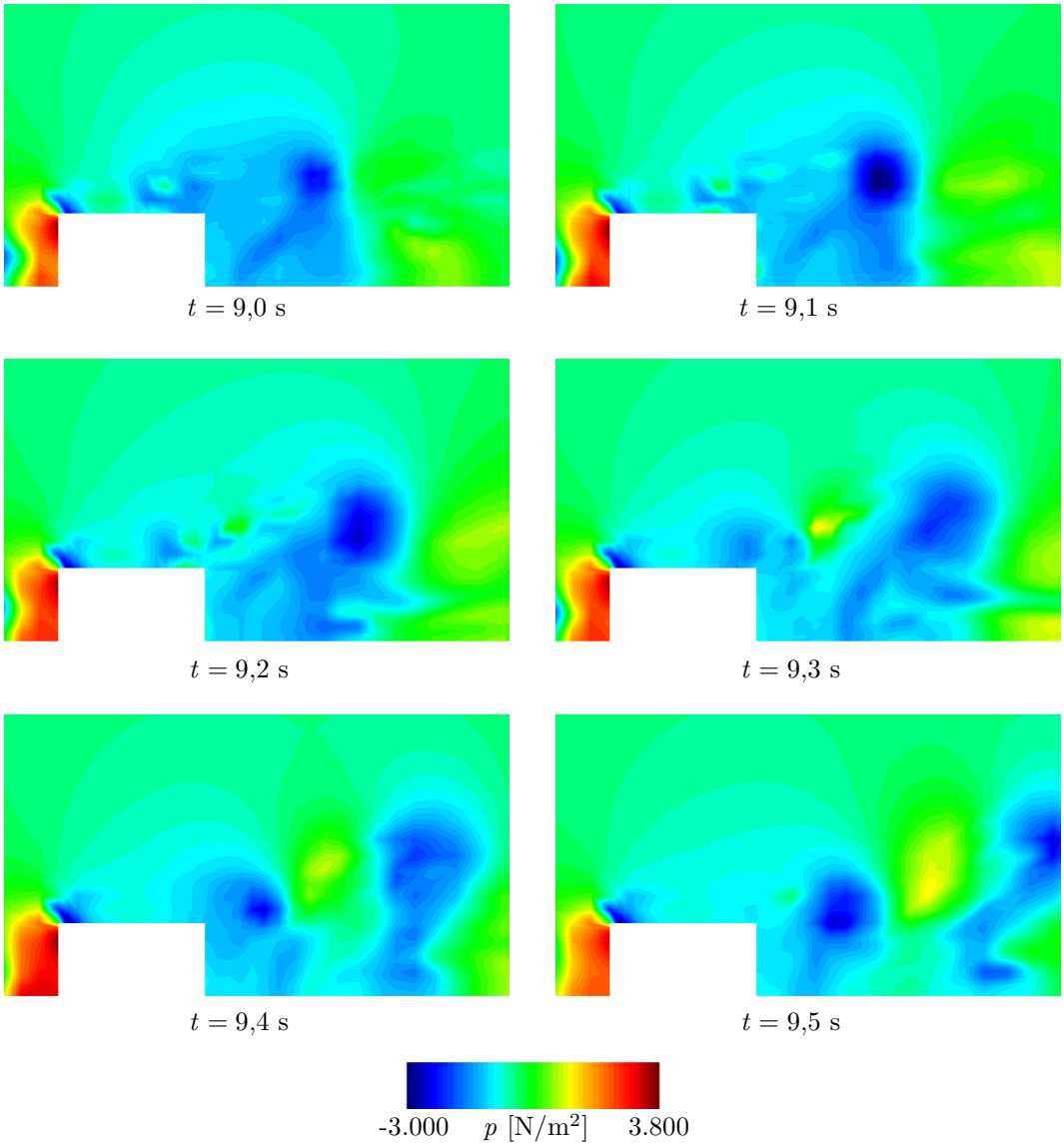


Abbildung 6.5: Gebäude mit starrem Dach: Momentaufnahmen des Fluid-Drucks in der Schnittebene $y = 50{,}0$ m (Ausschnitt $41{,}0 \text{ m} \leq x \leq 76{,}0$ m und $0{,}0 \text{ m} \leq z \leq 20{,}0$ m).

6.2 Gebäude mit flexiblem Dach

Ausgehend von der Problemstellung für das Gebäude mit starrem Dach wird in diesem Abschnitt untersucht, welche Unterschiede sich in der Fluidströmung und der daraus resultierenden Belastung auf das Dach des Gebäudes ergeben, wenn ein flexibles Membrandach berücksichtigt wird. Die Geometrie wird genauso wie in Abbildung 6.1 gewählt, das Dach des Gebäudes (graue Fläche) wird nun aber als flexible Struktur mitdiskretisiert.

Feld	Diskretisierung	Knoten	Elemente	Freiheitsgrade
Struktur	grob	16	9	
	fein	169	144	1.014
Fluid	grob	3.180	2.547	
	fein	172.653	163.008	690.612
ALE	grob	144	72	
	fein	5.577	4.608	16.731
Gesamt	fein	178.399	167.760	708.357

Tabelle 6.2: Gebäude mit flexiblem Dach: Diskretisierungen.

6.2.1 Fluid-Diskretisierung

Da das Fluidgebiet im Bereich des Dachs seine Größe verändern kann, muss hier eine ALE-Formulierung verwendet werden. Diese wird auf den Bereich des Fluids zwischen dem Membrandach und der oberen Berandung des Berechnungsgebiets beschränkt. Die Netzverschiebung wird in diesem Bereich mit einem Pseudo-Struktur-Ansatz berechnet. Dabei werden am Kopplungsrand mit der Struktur die Verschiebungen des Membrandachs vorgeschrieben und an den anderen Rändern, d.h. an den Rändern zum Euler-Gebiet und am oberen Rand des Berechnungsgebiets keine Verschiebungen zugelassen.

Die Diskretisierung des Fluidgebiets im Raum erfolgt in derselben Weise, wie im Abschnitt 6.1.1 für das erste Beispiel beschrieben. Ausgangspunkt ist wieder das grobe Netz aus Abbildung 6.2. Dieses wird nun mit einem Faktor vier in alle drei Richtungen verfeinert. Daraus folgt eine feine Diskretisierung mit 163.008 Elementen. Details zur Größe der Diskretisierungen von Fluid- und ALE-Gebiet sind in Tabelle 6.2 angegeben.

Die Diskretisierung in der Zeit erfolgt wieder mit dem BDF2-Verfahren, der Zeitschritt wird jedoch zu $\Delta t = 0{,}02$ s gewählt.

6.2.2 Struktur-Diskretisierung

Für die Beschreibung des Materialverhaltens der Struktur wird nach Hübner (2003) ein Saint-Venant-Kirchhoff-Modell mit den folgenden Parametern gewählt:

$$E_\mathrm{S} = 1{,}0 \cdot 10^9 \text{ N/m}^2, \qquad \nu_\mathrm{S} = 0{,}0, \qquad \rho_\mathrm{S} = 1000{,}0 \text{ kg/m}^3. \tag{6.4}$$

Die Dicke der Struktur beträgt 0,01 m. Daraus folgt für das Membrandach ein Länge/Dicke-Verhältnis von 1000.

Die Modellierung der dünnen Struktur erfolgt mit einer dreidimensional orientierten 7-Parameter-Schalenformulierung nach Büchter und Ramm (1992). Für die Diskretisierung werden zweidimensionale, dimensionsreduzierte Schalenelemente verwendet, wie sie z.B. in Gee (2004) beschrieben sind. Zur Vermeidung von unphysikalischen Versteifungseffekten kommen „enhanced assumed strain"- und „assumed natural strain"-Ansätze zum Einsatz. Ein „scaled director conditioning" (Gee u. a. 2005) mit dem Faktor 10,0 verbessert zusätzlich die Konditionierung des resultierenden Gleichungssystems. Das Membrandach wird für das grobe Netz mit neun 4-knotigen linearen Schalenelementen diskretisiert, die bei der Erzeugung des feinen Netzes mit einem Faktor vier in beide Richtungen unterteilt werden (Tabelle 6.2).

Die Diskretisierung in der Zeit erfolgt mit dem Generalisierten-α-Verfahren mit den folgenden Parametern:

$$\beta = 0{,}25 \qquad \gamma = 0{,}5 \qquad \alpha_{\mathrm{m}} = 0{,}5 \qquad \alpha_{\mathrm{f}} = 0{,}5 \tag{6.5}$$

Der Zeitschritt wird wie für das Fluid zu $\Delta t = 0{,}02$ s gewählt.

6.2.3 Randbedingungen

Für das Fluid werden neben den Kopplungsbedingungen am Interface mit dem Membrandach dieselben Randbedingungen wie für das starre Dach verwendet (siehe Abschnitt 6.1.2). Die Struktur ist an allen vier Rändern unverschieblich gelagert, d.h. die drei Verschiebungskomponenten sind gleich null, Rotationen jedoch möglich.

6.2.4 Kopplung

Um die Kopplung zwischen der Fluidströmung und der flexiblen Struktur implizit zu berücksichtigen, wird in diesem Beispiel ein Block-Gauß-Seidel-Verfahren (Abschnitt 5.6.1) verwendet. Es kommt ein Prädiktor nullter Ordnung nach Gleichung (5.13) zum Einsatz und der Relaxationsparameter wird in jeder Iteration nach dem Aitken-Verfahren berechnet. Die Iteration über die beiden physikalischen Felder wird beendet, wenn folgendes Abbruchkriterium erfüllt ist:

$$\frac{||\mathbf{r}^{(k+1)}||}{\sqrt{n_{\mathrm{eq}}}} \leq 1 \cdot 10^{-6}. \tag{6.6}$$

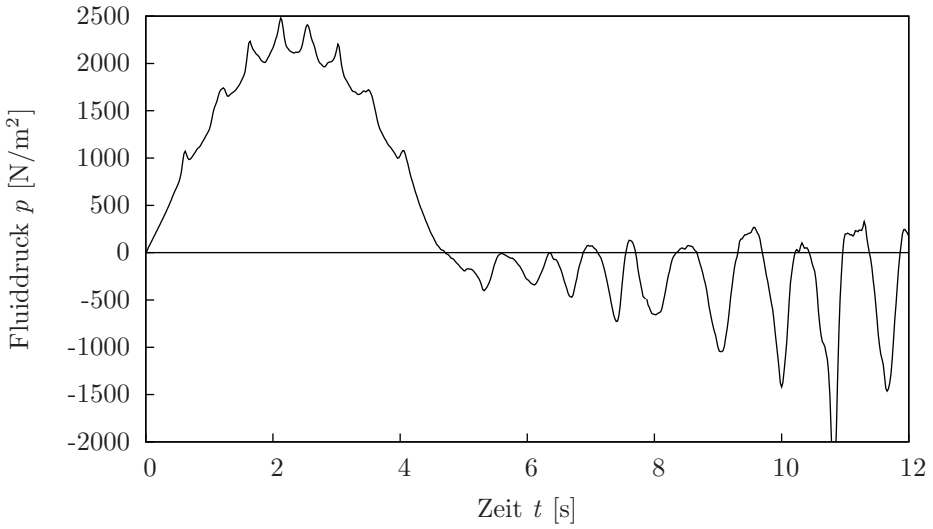


Abbildung 6.6: Gebäude mit flexiblem Dach: zeitliche Veränderung des Fluiddrucks p in Dachmitte ($\mathbf{x} = [50{,}0;\ 50{,}0;\ 5{,}0]^{\mathrm{T}}$ m).

6.2.5 Ergebnisse

Zur Lösung des gekoppelten Problems werden im Durchschnitt nur vier Iterationen über die beiden physikalischen Felder benötigt. Jede Lösung des Fluidfelds erfordert zu Beginn der Simulation drei, im späteren Verlauf etwa sechs bis acht Newton-Raphson-Iterationen. Die resultierenden linearen Gleichungssysteme werden mit einem GMRES-Verfahren mit der Unterraumgröße 120 und einer BILU-Vorkonditionierung gelöst. Für den nichtlinearen Strukturlöser steigt die Anzahl der Newton-Raphson-Iterationen im Laufe der Simulation von drei auf sechs.

Aufgrund der impliziten Erfüllung der Kopplungsbedingungen mithilfe der Iteration über die Felder ist die Fluid-Struktur-Wechselwirkungs-Simulation deutlich zeitaufwändiger als die reine Fluid-Simulation aus Abschnitt 6.1. Zur Berechnung der in diesem Beispiel verwendeten 600 Zeitschritte (12 s Simulationszeit) sind auf 32 Prozessoren des SX-8-Vektorrechners zweimal 48 Stunden „wall-clock time“ erforderlich.

Bereits im Zeitverlauf des Fluiddrucks in der Mitte des Membrandachs (Abbildung 6.6) zeigen sich Unterschiede zum starren Dach. Während der ersten 5,0 s wirkt auch bei der gekoppelten Simulation ein positiver Druck auf die Mitte des Dachs. Einerseits ist aber der Maximalwert mit 2.500 N/m^2 deutlich niedriger, andererseits ist der zeitliche Verlauf nicht mehr so glatt wie bei der ungekoppelten Simulation. Ist die maximale Einströmgeschwindigkeit bei 5,0 s erreicht, stellt sich zunächst, ähnlich wie beim starren Dach, ein Sog von ungefähr -500 N/m^2 ein. Daraus entwickelt sich im Anschluss aber eine Oszillation im Druck, die über die nächsten 5,0 s kontinuierlich größer wird. Ab

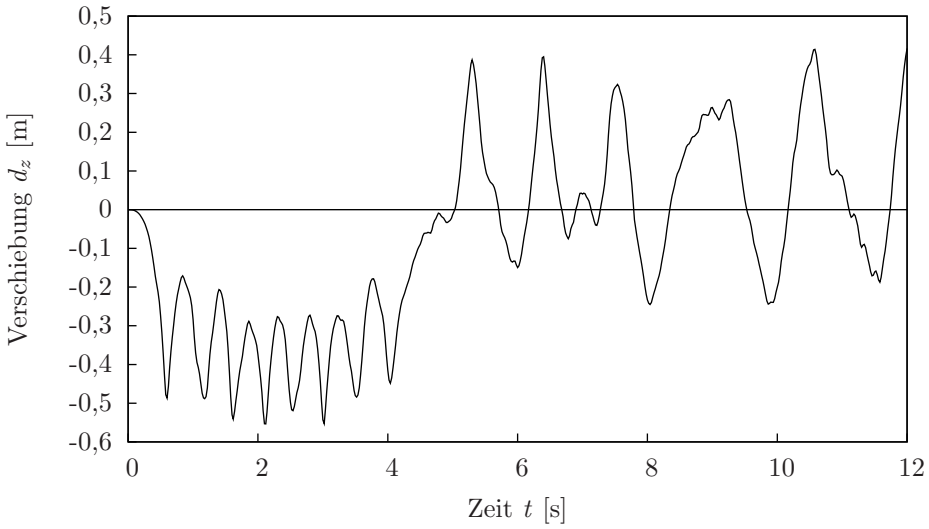


Abbildung 6.7: Gebäude mit flexiblem Dach: zeitliche Veränderung der vertikalen Verschiebungskomponente d_z in Dachmitte ($\mathbf{x} = [50{,}0;\ 50{,}0;\ 5{,}0]^{\mathrm{T}}$ m).

$t = 10{,}0$ s folgt eine unregelmäßige Schwankung des Drucks in der Mitte des Dachs zwischen -1.500 und $+500$ N/m^2. Diese Oszillation, wie auch schon die Druckspitzen während der ersten 5,0 s, werden durch Wirbel verursacht, die sich an der vorderen Gebäudekante ablösen und über das Membrandach stromabwärts transportiert werden.

Der wiederholte Vorzeichenwechsel im zeitlichen Verlauf des Fluiddrucks führt, unter Berücksichtigung des Eigengewichts der Struktur, zu einer Schwingung mit nahezu gleichgroßen Auslenkungen der Dachmitte nach oben und unten. Abbildung 6.7 zeigt den zeitlichen Verlauf der vertikalen Verschiebungskomponente in der Dachmitte. Die unregelmäßige Schwingung, die sich ab ca. $t = 8{,}0$ s einstellt, hat eine doppelt so große Periode wie die Oszillationen im Fluiddruck.

Abbildung 6.8 zeigt den Verformungszustand des Membrandachs zu verschiedenen Zeitpunkten während einer beispielhaften Schwingungsperiode. Die verformte Lage der Struktur ist hier zur besseren Visualisierung 5-fach überhöht dargestellt. Die Momentaufnahmen zeigen deutlich das wiederholte Durchschlagen der Membran während einer Periode und dreidimensionale Verformungszustände, die mit einer zweidimensionalen Simulation nicht erfasst werden können.

Im Vergleich zur Simulation ohne Fluid-Struktur-Kopplung hat die Berücksichtigung des flexiblen Membrandachs deutliche Unterschiede in der Systemantwort hervorgerufen. Einerseits wird die gegenseitige Beeinflussung der beiden Felder (Wirbelablösung an der vorderen Gebäudekante aufgrund der Verformung des Membrandachs) und die daraus resultierende Notwendigkeit einer gekoppelten Berechnung deutlich. Andererseits zeigt sich, dass durch eine Vernachlässigung der Kopplung die Lasten auf das Membran-

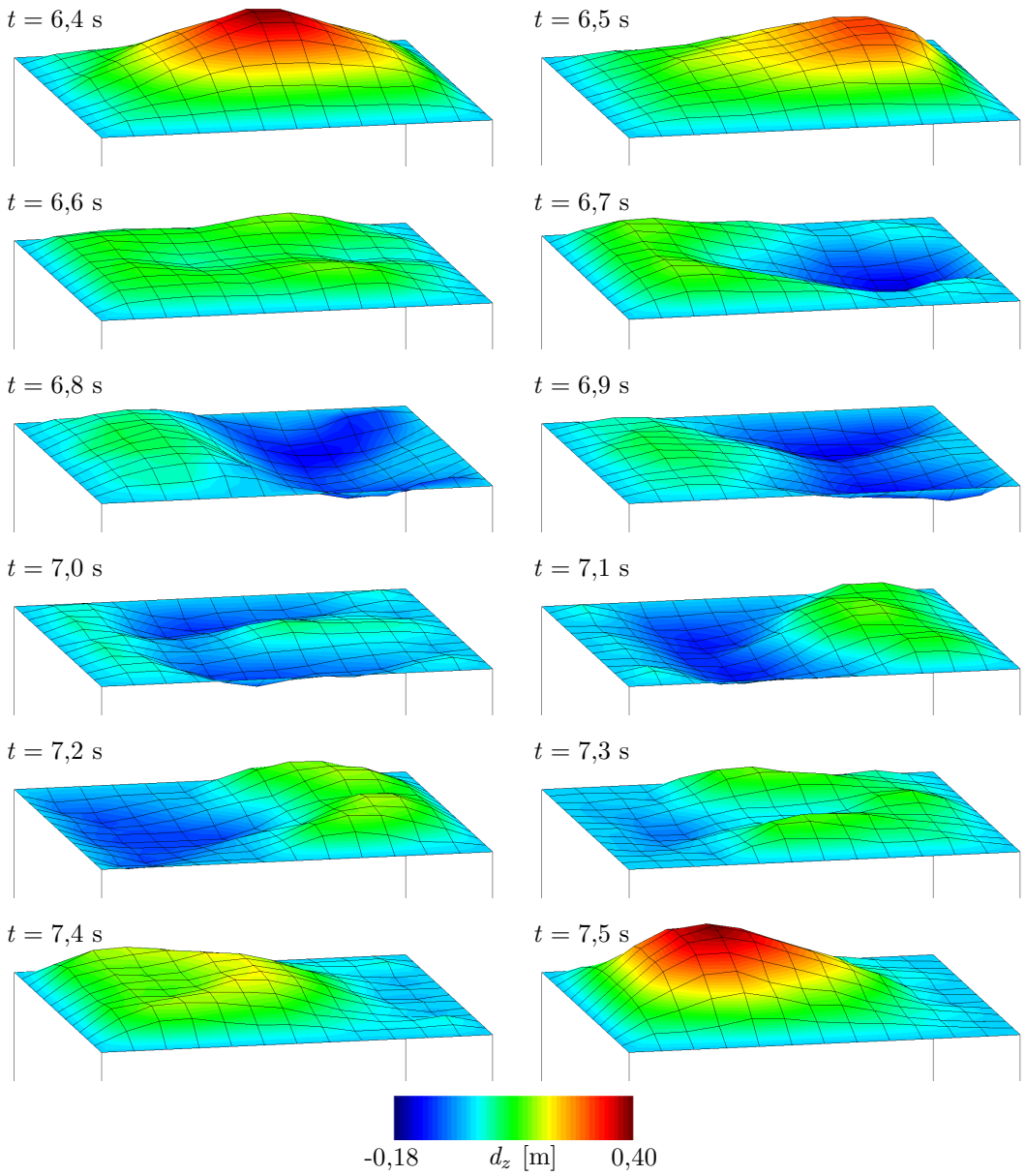


Abbildung 6.8: Gebäude mit flexiblem Dach: Momentaufnahmen des Verformungszustands des Membrandachs mit Farbkontur der vertikalen Veschiebungskomponente d_z (Verformung 5-fach überhöht).

dach unterschätzt werden und die dynamische Anregung des Dachs nicht berücksichtigt wird. Auch ist die Dreidimensionalität des Problems in den Verformungszuständen des Membrandachs deutlich zu erkennen.

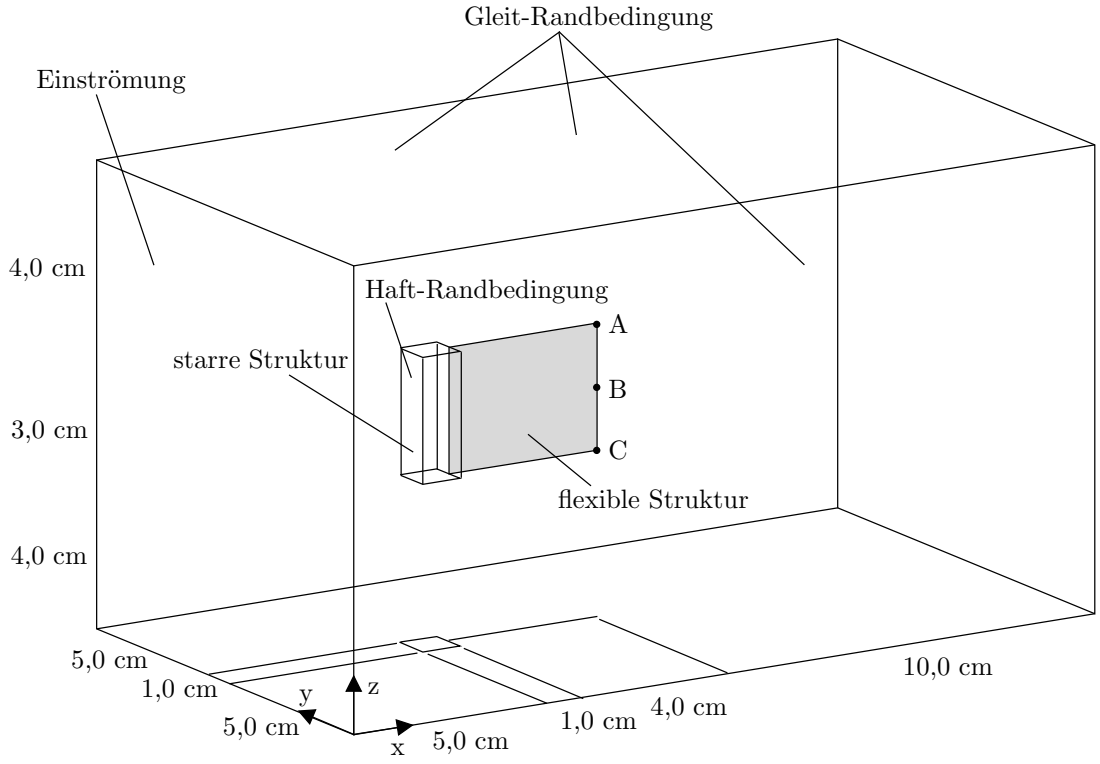


Abbildung 6.9: Fahne im Wind: Geometrie und Randbedingungen.

6.3 Flattern einer „Fahne“ im Wind

Im abschließenden Beispiel soll das Flattern einer „Fahne“ bei leichtem Wind simuliert werden. Um diesen sehr komplizierten Fall einer Fluid-Struktur-Wechselwirkung numerisch lösen zu können, müssen Vereinfachungen angenommen werden. Durch Vernachlässigung des Eigengewichts der Fahne kann Selbstkontakt vermieden werden, der in Kombination mit der Fluid-Struktur-Wechselwirkung eine besondere Herausforderung darstellt. Außerdem werden nur mäßige Windgeschwindigkeiten bis zu 100 cm/s betrachtet.

Untersucht wird eine Fahne der Größe $4{,}0 \times 3{,}0$ cm mit der Dicke $t_\mathrm{s} = 0{,}06$ cm, die an einem starren Hexaeder befestigt ist. Die Abmessungen der Fahne und alle Materialparameter sind der in Wall und Ramm (1998) vorgestellten zweidimensionalen Betrachtungsweise entnommen. Abbildung 6.9 zeigt die Geometrie und Randbedingungen der in diesem Abschnitt beschriebenen dreidimensionalen Untersuchung. Die Struktur ist in dieser Abbildung mithilfe ihrer Projektion auf die Grundfläche des Untersuchungsgebiets bemaßt.

Feld	Diskretisierung	Knoten	Elemente	Freiheitsgrade
Struktur	grob	126	48	
	fein	950	432	2.850
Fluid	grob	5.450	4.466	
	fein	123.674	115.362	494.696
ALE	grob	4.390	3.556	
	fein	99.062	92.052	297.186
Gesamt	fein	223.686	207.846	794.732

Tabelle 6.3: Fahne im Wind: Diskretisierungen.

6.3.1 Fluid-Diskretisierung

Für die Strömung wird ein Newton'sches Fluid mit den folgenden Materialparametern angenommen:

$$\nu_{\mathrm{F}} = 0{,}1542 \text{ cm}^2/\text{s}, \qquad \rho_{\mathrm{F}} = 1{,}18 \cdot 10^{-3} \text{ kg/cm}^3. \tag{6.7}$$

Dies entspricht den Eigenschaften von Luft bei 20°C.

Die Diskretisierung des Fluids erfolgt erneut mit SUPG/PSPG-stabilisierten linearen Hexaeder-Elementen. Aus einem groben Netz mit 4.466 Elementen, das der Präprozessor generiert, wird durch Verfeinerung mit dem Faktor drei das feine Rechengitter erzeugt. Die verwendete Anzahl an Knoten, Elementen und Freiheitsgraden für beide Diskretisierungen sind in Tabelle 6.3 angegeben. Die zeitliche Diskretisierung erfolgt mit BDF2-Verfahren und einem Zeitschritt $\Delta t = 0{,}01$ s.

Um die Veränderung des Fluidgebiets durch die Bewegung der Struktur berücksichtigen zu können, wird für den Bereich $6{,}0 \text{ cm} \leq x \leq 20{,}0 \text{ cm}$ eine ALE-Formulierung verwendet. Die Netzverschiebung in diesem Bereich wird mit einem Pseudo-Struktur-Ansatz berechnet. Am Kopplungsrand mit der Struktur werden die Verschiebungen der Fluid-Knoten als Dirichlet-Randbedingung vorgeschrieben. Am Rand des Berechnungsgebiets und dem Übergang von ALE- zu Euler-Formulierung sind keine Netzverschiebungen zugelassen.

6.3.2 Struktur-Diskretisierung

Das Materialverhalten der Fahne wird gemäß Wall und Ramm (1998) mit einem Saint-Venant-Kirchhoff-Modell mit den folgenden Parametern beschrieben:

$$E_{\mathrm{S}} = 2{,}0 \cdot 10^6 \text{ N/cm}^2, \qquad \nu_{\mathrm{S}} = 0{,}35, \qquad \rho_{\mathrm{S}} = 2{,}0 \text{ kg/cm}^3. \tag{6.8}$$

Die Dicke der Struktur beträgt $t_{\mathrm{S}} = 0{,}06$ cm und damit ist das Länge/Dicke-Verhältnis 66. Da in diesem Beispiel die dünne Struktur an beiden Seiten umströmt wird, ist es von Vorteil Kontinuums-Schalenelemente für die Diskretisierung zu verwenden. Zum Einsatz kommt hier wiederum die 7-Parameter-Schalenformulierung unter Verwendung von „enhanced assumed strain"- und „assumed natural strain"-Ansätzen.

Wie bereits beim Fluid, wird auch für die Struktur auf dem Hochleistungsrechner aus einer groben Diskretisierung mit 48 Elementen durch eine Unterteilung mit dem Faktor drei eine feine Diskretisierung erzeugt. Dabei ist zu beachten, dass in Dickenrichtung der Fahne keine Verfeinerung vorgenommen wird. Die Details zu den resultierenden Diskretisierungen sind auch in Tabelle 6.3 aufgelistet.

Für die Diskretisierung in der Zeit kommt das Generalisierte-α-Verfahren mit den Parametern des vorherigen Beispiels (Abschnitt 6.2.2) zum Einsatz. Der Zeitschritt wird wie im Fluid zu $\Delta t = 0{,}01$ s gewählt.

6.3.3 Randbedingungen

Für die Randfläche $x = 0{,}0$ m wird eine räumlich konstante Einström-Randbedingung für die Geschwindigkeitskomponente in x-Richtung vorgegeben

$$\hat{u}_x(t) = 100{,}0 \text{ cm/s} \cdot \hat{u}_t(t), \tag{6.9}$$

die von 0,0 bis 2,0 s gemäß Gleichung (6.10) gesteigert wird (Abbildung 6.10):

$$\hat{u}_t(t) = \left(\sin\left(\pi \left(\frac{t}{2{,}0} - 0{,}5 \right) \right) + 1 \right) \cdot 0{,}5 \ , \tag{6.10}$$

und ab $t = 2{,}0$ s konstant bleibt ($\hat{u}_t(t) = 1{,}0$). Bei Verwendung der Länge der Fahne ($l = 4{,}0$ cm) und der maximalen Einströmgeschwindigkeit ($\hat{u}_x = 100{,}0$ cm/s) folgt für diese Problemstellung eine Reynoldszahl von $\mathrm{Re} \approx 2500$.

An den vier Längsseiten des Rechengebiets werden Gleit-Randbedingungen („slip") vorgegeben und an allen Seiten des starren Hexaders Haft-Randbedingungen („no-slip").

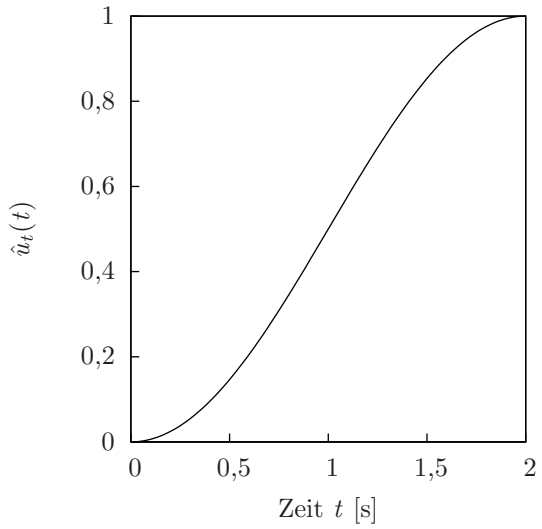


Abbildung 6.10: Fahne im Wind: zeitliche Veränderung der Einström-Randbedingung.

Für den Ausströmrand bei $x = 20{,}0$ cm werden keine Randbedingungen („do-nothing“) vorgeschrieben.

Die Struktur ist an allen Knoten am Befestigungsrand ($x = 6{,}0$ cm) unverschieblich gelagert.

6.3.4 Kopplung

Auch für diese Simulation wird eine implizite Kopplung von Fluid und Struktur eingesetzt. Nach einem Prädiktor nullter Ordnung wird die relaxierte Block-Gauß-Seidel-Iteration über die beiden physikalischen Felder bei Erreichen des Abbruchkriteriums

$$\frac{||\mathbf{r}^{(k+1)}||}{\sqrt{n_{\text{eq}}}} \leq 1 \cdot 10^{-6} \tag{6.11}$$

beendet. Der Relaxationsparameter wird in jedem Iterationsschritt nach dem Aitken-Verfahren berechnet.

6.3.5 Ergebnisse

Zu Beginn der Simulation (bis etwa $t = 2{,}3$ s) sind keine Iterationen über die Felder erforderlich, da sich die Struktur aufgrund der symmetrischen Belastung von beiden Seiten nicht verformt. Im weiteren Verlauf der Berechnung werden 3–4 Iterationen über die Felder benötigt, um die kinematischen Kopplungsbedingungen zu erfüllen.

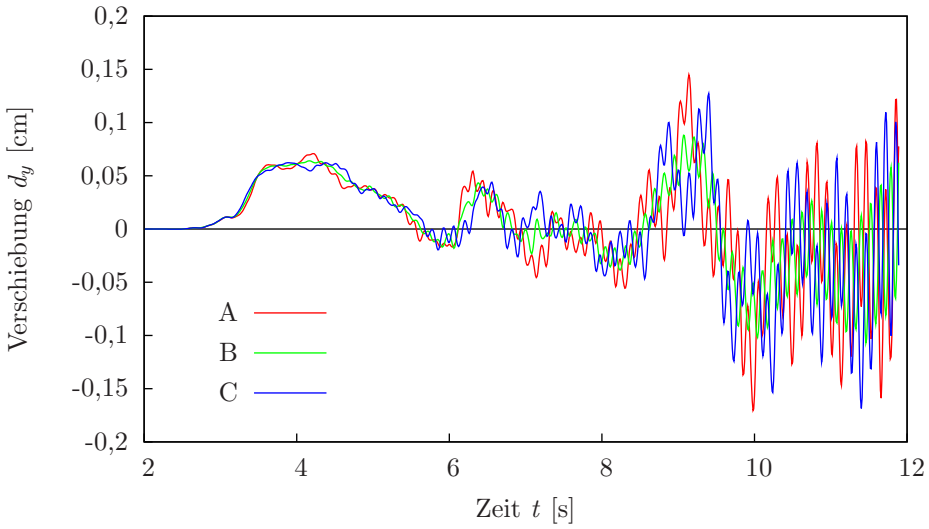


Abbildung 6.11: Fahne im Wind: Verschiebungen d_y in y-Richtung an drei Punkten an der hinteren Kante der Struktur (BiCGSTAB, Jacobi(4), 0,3).

Für die Lösung des nichtlinearen Fluidproblems werden in Abhängigkeit von der Kopplungsiteration 10 bis 20 Newton-Raphson-Iterationen benötigt. Erfolgt die Lösung der aus der Fluiddiskretisierung stammenden linearen Gleichungssysteme mit einem GMRES-Verfahren mit Unterraumgröße 120 und BILU-Vorkonditionierung, können auf 32 Prozessoren des SX-8-Vektorrechners in 48 Stunden „wall-clock time“ ungefähr 900 Zeitschritte gelöst werden. Kommt hingegen eine Jacobi-Vorkonditionierung mit einem Relaxationsfaktor $\omega = 0{,}3$ in Verbindung mit einem BiCGSTAB-Verfahren zum Einsatz, können in derselben Rechenzeit fast 1.200 Zeitschritte und damit ca. 12 s Simulationszeit berechnet werden. Die meisten der im Folgenden vorgestellten Ergebnisse resultieren aus dieser zweiten Simulation.

Für die Untersuchung des Verhaltens der Fahne im Wind wird zunächst die Verschiebungskomponente in y-Richtung an drei Punkten A, B und C an der hinteren Kante der Fahne betrachtet. Die genaue Lage der Punkte ist in Abbildung 6.9 eingetragen. In Abbildung 6.11 sind die zeitlichen Verläufe der drei Verschiebungen aufgetragen.

Von Beginn der Simulation bis ca. $t = 2{,}5$ s zeigt die Struktur in der laminaren Strömung aufgrund der symmetrischen Belastung von beiden Seiten keine Verformung. Erst nach Ablösung des ersten Wirbels am starren Hexaeder wird die Fahne, zunächst noch gleichmäßig, in positive y-Richtung ausgelenkt. Die zugehörige Verformung ist für den Zeitpunkt $t = 4{,}0$ s in Abbildung 6.12 a) dargestellt.

Ab $t = 6{,}0$ s wird dann die Symmetrie in der Verformung der Fahne durch sich unregelmäßig ablösende Wirbel durchbrochen. Zunächst lassen sich langwellige antimetrische Moden (Abbildung 6.12 b)), die teilweise mit symmetrischen Verformungen überlagert

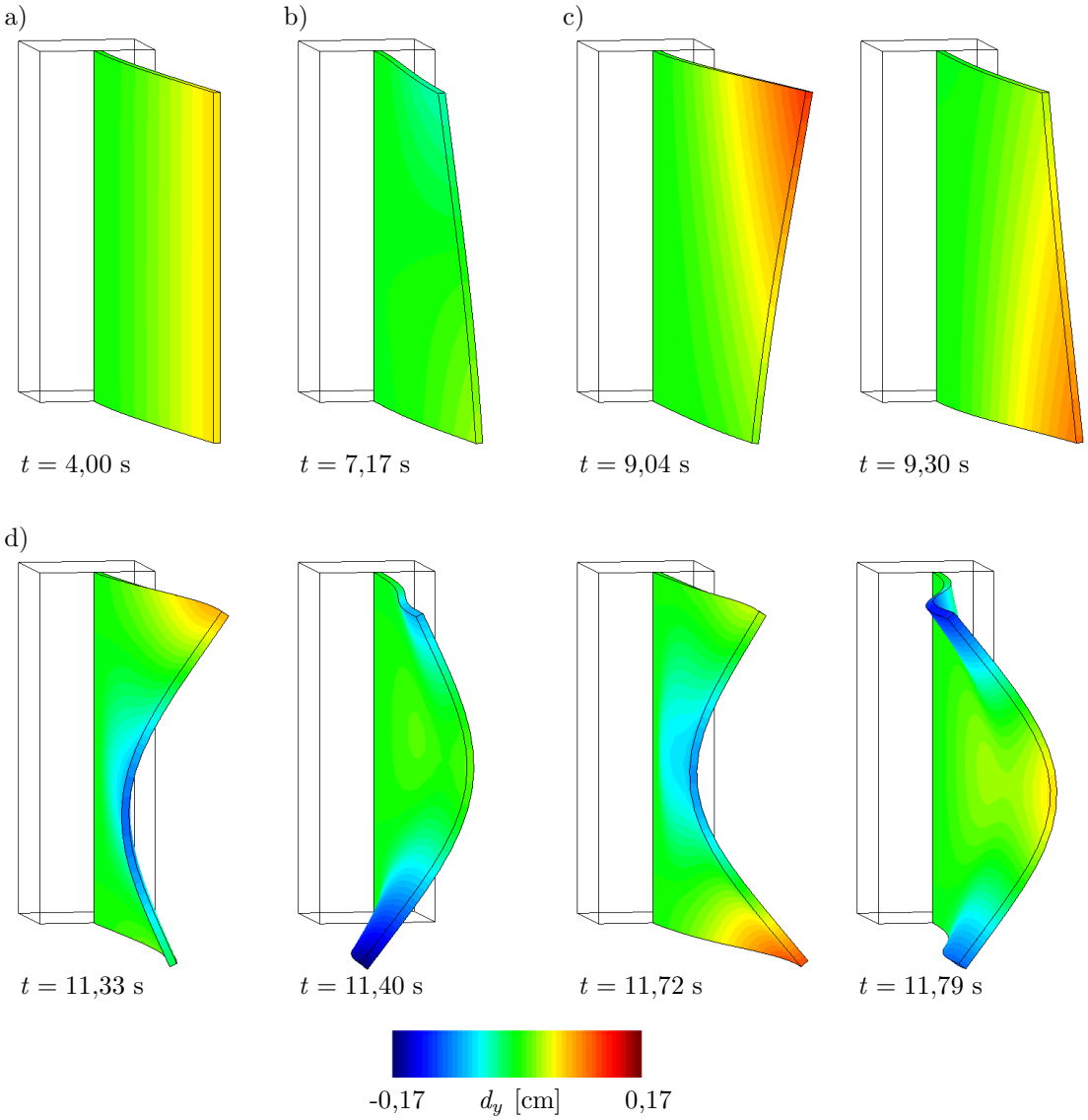


Abbildung 6.12: Fahne im Wind: Momentaufnahmen des Verformungszustands des Struktur mit Farbkontur der Veschiebungkomponente d_y (Verformung 5-fach überhöht).

sind (Abbildung 6.12 c)), erkennen. Im weiteren Verlauf der Simulation zeigen sich, wie in Abbildung 6.12 d) zu sehen ist, in der Verformung der Struktur sowohl in z- als auch in x-Richtung zusätzlich kurzwelligere Moden.

Dass trotz einer perfekten, symmetrischen Problembeschreibung unsymmetrische Systemantworten aus der Simulation hervorgehen, zeigt den hohen Grad an physikalischer Instabilität der Aufgabenstellung. Durch sehr kleine (numerische) Störungen wird das System von der symmetrischen Antwort abgelenkt.

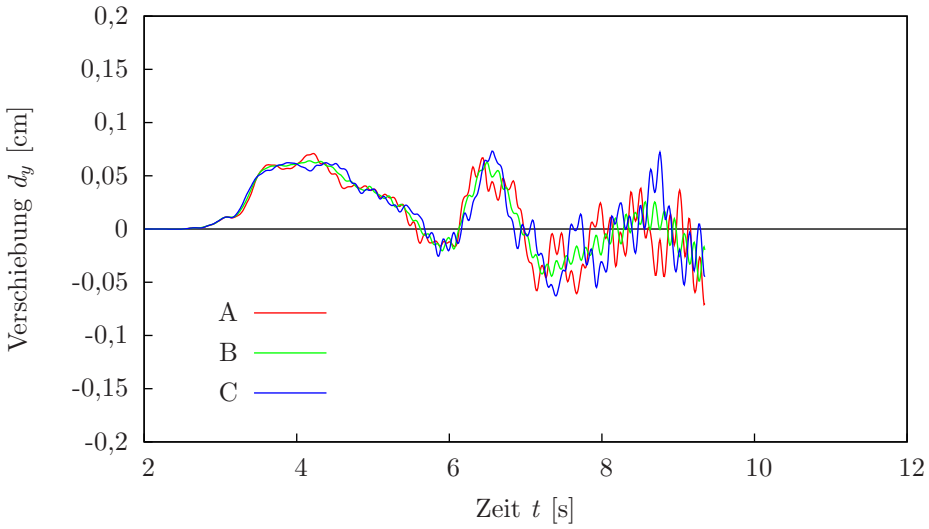


Abbildung 6.13: Fahne im Wind: Verschiebungen d_y in y-Richtung an drei Punkten an der hinteren Kante der Struktur (GMRES(120), BILU).

Wie stark die Systemantwort von den Eingangsparametern abhängt, zeigt der Vergleich der Abbildungen 6.11 und 6.13. Bei diesen beiden Berechnungen derselben Aufgabenstellung ist nur der lineare Löser des Fluidfelds verändert. Während für das Diagramm in Abbildung 6.11 ein BiCGSTAB-Verfahren mit Jacobi-Vorkonditionierung verwendet wird, werden in der Vergleichsrechnung (Abbildung 6.13) die linearen Gleichungssysteme bei gleicher Genauigkeitsanforderung mit einem GMRES-Verfahren mit BILU-Vorkonditionierung gelöst. Für die ersten sechs Sekunden zeigen beide Berechnungen eine identische Systemantwort. Wird aber die physikalische Instabilität erreicht, genügen kleine Unterschiede bei der Lösung der linearen Gleichungssysteme, um zwei verschiedene Systemantworten zu erzeugen.

7

Zusammenfassung und Ausblick

7.1 Zusammenfassung

Die numerische Simulation von komplexen dreidimensionalen Fluid-Struktur-Wechselwirkungs-Problemstellungen stellt immer noch sehr hohe Anforderungen an die verfügbare Computer-Hardware. Die Effizienz der Software ist daher für die Durchführbarkeit solcher Analysen von großer Bedeutung. In allen Vorgängerarbeiten zum Thema Fluid-Struktur-Wechselwirkung am Institut für Baustatik und Baudynamik (Wall 1999, Mok 2001, Förster 2007) wurde die Effizienzsteigerung als notwendige Weiterentwicklung aufgeführt, um die entwickelten Algorithmen, zum Beispiel zur Untersuchung des dynamischen Stabilitätsverhaltens von Schalenstrukturen in dreidimensionalen Strömungen, anwenden zu können.

Im Rahmen dieser Arbeit wurde der Aspekt der Effizienz von Fluid-Struktur-Wechselwirkungs-Simulationen aus verschiedenen Blickrichtungen analysiert und weiterentwickelt. Dabei wurden einerseits Aspekte der Einzelfeldlöser, wie Implementierung und Lösung der linearen Gleichungssysteme, betrachtet und andererseits die Frage des Kopplungsalgorithmus zwischen Fluid und Struktur untersucht.

Die Vernetzung von komplexen, dreidimensionalen Gebieten ist ein sehr rechenaufwändiger Prozess. Durch eine in dieser Arbeit beschriebene Vernetzung in zwei Stufen kann der Aufwand im Präprozessor deutlich verringert werden. Zudem ermöglicht die Verfeinerung der Präprozessor-Diskretisierung auf dem Hochleistungsrechner eine beliebige h- und p-Adaption auf der Basis eines einmal validierten Netzes. Durch sukzessive Anwendung der Verfeinerung lassen sich auch Netzhierarchien als Ausgangspunkt für Mehrgitter-Verfahren erzeugen.

Am Beispiel einer dreidimensionalen Strömung wurde eine effiziente Berechnung der Elementmatrizen auf Vektorrechnern vorgestellt. Durch eine Gruppierung der Elemente und Umstrukturierung der Implementierung konnte unter Beachtung der Grundregeln der Vektorisierung die Rechengeschwindigkeit für das Aufstellen der Elementmatrizen um den Faktor 30 gesteigert werden. Beim Vergleich des Einflusses verschiedener Programmiersprachen und der Größe der Elementgruppen auf die Rechengeschwindigkeit hat sich Fortran auf allen untersuchten Hardware-Architekturen als effizienteste Programmiersprache herausgestellt. Die optimale Größe der Elementgruppen hängt dagegen vom verwendeten Prozessor ab.

Neben der Berechnung der Elementmatrizen ist das Lösen des linearen Gleichungssystems der zeitaufwändigste Teil einer Finite-Element-Simulation. Die Untersuchung des Einflusses sowohl verschiedener Parameter der Lösungsverfahren und der Diskretisierung als auch der verwendeten Hardware-Architektur auf die Effizienz des Lösers zeigte deutliche Unterschiede in der benötigten Rechenzeit. Durch eine geschickte Wahl des Iterationsverfahrens und insbesondere des Vorkonditionierers konnte die Rechenzeit, die zur Lösung eines Gleichungssystems benötigt wird, erheblich reduziert werden.

Bei der partitionierten Lösung von gekoppelten Problemen hat neben den Effizienzfragen der Einzelfelder auch das Kopplungsverfahren einen erheblichen Einfluss auf die Gesamtrechenzeit. Die vorliegende Arbeit enthält einen Überblick über die wichtigsten partitionierten Lösungsverfahren in einer einheitlichen Darstellung. Weiterhin wurden Zwei-Level-Verfahren vorgestellt, die eine Lösung des gekoppelten Problems auf einer gröberen Diskretisierung nutzen, um die Erfüllung der Kopplungsbedingungen auf dem feinen Netz zu beschleunigen. Ein Vergleich für unterschiedlich stark gekoppelte Problemstellungen zeigte die Vor- und Nachteile der verschiedenen Verfahren.

Durch die Steigerung der Effizienz in den verschiedenen Bereichen einer gekoppelten Fluid-Struktur-Wechselwirkungs-Simulation ist es nun möglich, das dynamische Verhalten von Schalenstrukturen in komplexen dreidimensionalen Strömungen zu untersuchen. Anwendungsmöglichkeiten der in dieser Arbeit verwendeten Algorithmen wurden anhand von numerischen Beispielen aufgezeigt.

7.2 Ausblick

Trotz der Anwendungsmöglichkeiten, die sich durch die effiziente Lösung von dreidimensionalen, gekoppelten Problemstellungen ergeben, besteht bis zur wirklichkeitsnahen Simulation realer Strukturen noch erheblicher Forschungs- und Entwicklungsbedarf. Dabei kann eine Spezialisierung des bisher verwendeten sehr allgemeinen Ansatzes auf eine bestimmte Problemklasse von Vorteil sein. So spielen zum Beispiel bei der Untersuchung

von Blutströmungen in flexiblen Adern andere Modellierungsaspekte eine wichtige Rolle als bei der Wechselwirkung zwischen leichten Bauwerken und Windströmungen.

Bei der Simulation von windumströmten Bauwerken ist z.B. die Berücksichtigung der Turbulenz entscheidend. Hier wäre die Implementierung eines entsprechenden Modells ein wichtiger Schritt zu realitätsnahen Simulationen. Das SST-Modell („shear stress transport“) nach Menter (1994) scheint für diesen Zweck gut geeignet. Auch eine Berücksichtigung der räumlichen und zeitlichen Schwankungen in der Windgeschwindigkeit (Böen) ist für realitätsnahe Untersuchungen von Gebäude-Wind-Interaktionen von großer Bedeutung.

Als Weiterentwicklung der in dieser Arbeit vorgestellten Methoden bietet sich auch die Untersuchung von Mehrgitter-Verfahren als linearer Löser bzw. Vorkonditionierer für die Einzelfelder im Vergleich zu den klassischen Iterationsverfahren an. Im Bereich der Netzerzeugung würden sich durch die algorithmische Behandlung von hängenden Knoten und nicht-konformen Netzen am Kopplungsrand die Möglichkeiten einer adaptiven Vernetzung eröffnen.

Aufbauend auf die vorgestellten Zwei-Level-Verfahren zur Interaktion von Fluid und Struktur ist eine Untersuchung der Anwendung von kompletten Mehr-Level-Verfahren auf die Kopplung auf jeden Fall erstrebenswert. In diesem Bereich liegt viel Potential zur weiteren Beschleunigung der Kopplungsiteration.

Vor allem ist jedoch eine Verifikation der Simulations-Ergebnisse der komplexen Fluid-Struktur-Wechselwirkung erforderlich. Dies kann einerseits durch den Vergleich von Ergebnissen unterschiedlicher numerischer Verfahren geschehen. Andererseits bieten sich dazu auch Vergleiche der Ergebnisse von numerischen Untersuchungen mit speziell dafür durchgeführten Experimenten oder Windkanaluntersuchungen und Messungen an realen Bauwerken an.

Literaturverzeichnis

Aitken 1937
: Aitken, A.C.: Studies in practical mathematics II. The evaluation of the latent roots and latent vectors of a matrix. In: *Proceedings of the Royal Society of Edinburgh* 57 (1937), S. 269–304

Altenbach und Altenbach 1994
: Altenbach, J.; Altenbach, H.: *Einführung in die Kontinuumsmechanik*. Teubner Studienbücher, Stuttgart, 1994

Argyris 1954
: Argyris, J.H.: Energy theorems and structural analysis: A generalized discourse with applications on energy principles of structural analysis including the effects of temperature and non-linear stress-strain relations. In: *Aircraft Engineering and Aerospace Technology* 26 (1954), S. 347–356

Badia und Codina 2007
: Badia, S.; Codina, R.: On some fluid-structure iterative algorithms using pressure segregation methods. Application to aeroelasticity. In: *International Journal for Numerical Methods in Engineering* 72 (2007), S. 46–71

Badia u. a. 2008
: Badia, S.; Nobile, F.; Vergara, Ch.: Fluid-structure partitioned procedures based on Robin transmission conditions. In: *Journal of Computational Physics* 227 (2008), S. 7027–7051

Bangert u. a. 2004
: Bangert, F.; Kuhl, D.; Meschke, G.: Chemo-hygro-mechanical modelling and nume-

rical simulation of concrete deterioration caused by alkalisilica reaction. In: *International Journal for Numerical and Analytical Methods in Geomechanics* 28 (2004), S. 689–714

Barcelos u. a. 2006
Barcelos, M.; Bavestrello, H.; Maute, K.: A Schur-Newton-Krylov solver for steady-state aeroelastic analysis and design sensitivity analysis. In: *Computer Methods in Applied Mechanics and Engineering* 195 (2006), S. 2050–2069

Barcelos und Maute 2008
Barcelos, M.; Maute, K.: Aeroelastic design optimization for laminar and turbulent flows. In: *Computer Methods in Applied Mechanics and Engineering* 197 (2008), S. 1813–1832

Barrenechea und Valentin 2002
Barrenechea, G.; Valentin, F.: An unusual stabilized finite element method for a generalized Stokes problem. In: *Numerische Mathematik* 92 (2002), S. 652–677

Behr u. a. 2000
Behr, M.; Pressel, D.M.; Sturek, W.B.: Comments on CFD code performance on scalable architectures. In: *Computer Methods in Applied Mechanics and Engineering* 190 (2000), S. 263–277

Belytschko u. a. 2000
Belytschko, T.; Liu, W.K.; Moran, B.: *Nonlinear finite elements for continua and structures*. John Wiley & Sons, Chichester, 2000

Berger 2005
Berger, H.: *NEC SX-8 at HLRS*. NEC High Performance Computing, Stuttgart, 2005

Bischoff 1999
Bischoff, M.: *Theorie und Numerik einer dreidimensionalen Schalenformulierung*, Bericht Nr. 30, Institut für Baustatik, Universität Stuttgart, Dissertation, 1999

Braess 1995
Braess, D.: Towards algebraic multigrid for elliptic problems of second order. In: *Computing* 55 (1995), S. 379–393

Brezzi und Fortin 1991
Brezzi, F.; Fortin, M.: *Mixed and hybrid finite element methods*. Computational Mathematics, Vol. 15, Springer-Verlag, Berlin, Heidelberg, 1991

Briggs u. a. 2000
Briggs, W.L.; Henson, V.E.; McCormick, S.F.: *A Multigrid Tutorial, Second Edition.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 2000

Bungartz u. a. 1998
Bungartz, H.-J.; Frank, A.; Meier, F.; Neunhoeffer, T.; Schulte, S.: Fluid structure interaction: 3D numerical simulation and visualization of a micropump. In: Friedrich, R.; Bontoux, P. (Hrsg.): *Computation and Visualization of Three-Dimensional Vortical and Turbulent Flow.* Vieweg, Braunschweig, 1998, S. 350–368

Bungartz und Schulte 1995
Bungartz, H.-J.; Schulte, S.: Coupled problems in microsystem technology. In: Hackbusch, W.; Wittum, G. (Hrsg.): *Numerical Treatment of Coupled Systems.* Vieweg, Braunschweig, 1995, S. 11–24

Bungartz und Schäfer 2006
Bungartz, Hans-Joachim; Schäfer, Michael (Hrsg.): *Fluid-structure interaction. Modelling, simulation, optimisation.* Series: Lecture Notes in Computational Science and Engineering, Vol. 53, Springer-Verlag, Berlin, Heidelberg, 2006

Büchter und Ramm 1992
Büchter, N.; Ramm, E.: 3d-extension of nonlinear shell equations based on the enhanced assumed strain concept. In: Hirsch, C.; Périaux, J.; Oñate, E. (Hrsg.): *Computational Methods in Applied Science.* Elsevier Science Publisher B.V., 1992, S. 55–62

Büchter u. a. 1994
Büchter, N.; Ramm, E.; Roehl, D.: Three-dimensional extension of nonlinear shell formulation based on the enhanced assumed strain concept. In: *International Journal for Numerical Methods in Engineering* 37 (1994), S. 2551–2568

Causin u. a. 2005
Causin, P.; Gerbeau, J.-F.; Nobile, F.: Added-mass effect in the design of partitioned algorithms for fluid-structure problems. In: *Computer Methods in Applied Mechanics and Engineering* 194 (2005), S. 2551–2568

Cervera u. a. 1996
Cervera, M.; Codina, R.; Galindo, M.: On the computational efficiency and implementation of block-iterative algorithms for nonlinear coupled problems. In: *Engineering Computations* 13 (1996), S. 4–30

Chen u. a. 2003
Chen, P.; Zheng, D.; Sun, S.; Yuan, M.: High performance sparse static solver in

finite element analysis with loop-unrolling. In: *Advances in Engineering Software* 34 (2003), S. 203–215

Chung und Hulbert 1993
Chung, J.; Hulbert, G.M.: A time integration algorithm for structural dynamics with improved numerical dissipation: The generalized-α method. In: *Journal of Applied Mathematics* 60 (1993), S. 371–375

Clough 1960
Clough, R.W.: The finite element method in plane stress analysis. In: *Proc. of the ASCE Conference on Electronic Computation, Pittsburg*, 1960

Codina 2002
Codina, R.: Stabilized finite element approximation of transient incompressible flows using orthogonal subscales. In: *Computer Methods in Applied Mechanics and Engineering* 190 (2002), S. 2681–2706

Cuvelier u. a. 1986
Cuvelier, C.; Sega, A.; van Steenhoven, A.A.: *Finite element methods and Navier-Stokes equations*. D. Reidel Publishing Company, 1986

Dahl und Wille 1992
Dahl, O.; Wille, S.O.: An ILU preconditioner with coupled node fill-in for iterative solution of the mixed finite element formulation of the 2D and 3D Navier-Stokes equations. In: *International Journal for Numerical Methods in Fluids* 15 (1992), S. 525–544

Degroote u. a. 2008
Degroote, J.; Bruggeman, P.; Haelterman, R.; Vierendeels, J.A.: Stability of a coupling technique for partitioned solvers in FSI applications. In: *Computers & Structures* (2008). – DOI: 10.1016/j.compstruc.2008.05.005

Deparis 2004
Deparis, S.: *Numerical analysis of axisymmetric flows and methods for fluid-structure interaction arising in blood flow simulation*, École polytechnique fédérale de Lausanne, Dissertation, 2004

Deparis u. a. 2006
Deparis, S.; Discacciati, M.; Quarteroni, A.: A domain decomposition framework for fluid-structure interaction problems. In: *Proceedings of the Third International Conference on Computational Fluid Dynamics (ICCFD3)*, 2006, S. 41–58

Dettmer 2004
Dettmer, W.G.: *Finite element modelling of fluid flow with moving free surfaces and*

interfaces including fluid-solid interaction, University of Wales Swansea, School of Engineering, Dissertation, 2004

Dettmer und Perić 2006
Dettmer, W.G.; Perić, D.: A computational framework for fluid-structure interaction: Finite element formulation and application. In: *Computer Methods in Applied Mechanics and Engineering* 195 (2006), S. 5754–5779

Diniz dos Santos u. a. 2008
Diniz dos Santos, N.; Gerbeau, J.-F.; Bourgat, J.-F.: A partitioned fluid-structure algorithm for elastic thin valves with contact. In: *Computer Methods in Applied Mechanics and Engineering* 197 (2008), S. 1750–1761

Dohrmann und Bochev 2004
Dohrmann, C.R.; Bochev, P.B.: A stabilized finite element method for the Stokes problem based on polynomial pressure projections. In: *International Journal for Numerical Methods in Fluids* 46 (2004), S. 183–201

Donea 1983
Donea, J.: Arbitrary Lagrangian-Eulerian finite element methods. In: Hughes, T.J.R.; Belytschko, T. (Hrsg.): *Computational Methods for Transient Analysis*. Elsevier Science Publishers B.V., 1983, S. 473–516

Donea und Huerta 1989
Donea, J.; Huerta, A.: *Finite element methods for flow problems*. John Wiley & Sons, Chichester, 1989

Donea u. a. 2004
Donea, J.; Huerta, A.; Ponthot, J.-P.; Rodriguez-Ferran, A.: Arbitrary Lagrangian-Eulerian methods. In: Stein, E.; de Borst, R.; Hughes, T.J.R. (Hrsg.): *Encyclopedia of Computational Methods, Volume 1: Fundamentals*. John Wiley & Sons, Chichester, 2004, S. 413–438

Ethier und Steinman 1994
Ethier, C.R.; Steinman, D.A.: Exact fully 3D Navier Stokes solution for benchmarking. In: *International Journal for Numerical Methods in Fluids* 19 (1994), S. 369–375

Farhat und Lesoinne 2000
Farhat, C.; Lesoinne, M.: Two efficient staggered algorithms for the serial and parallel solution of three-dimensional nonlinear transient aeroelastic problems. In: *Computer Methods in Applied Mechanics and Engineering* 182 (2000), S. 499–515

Farhat u. a. 1995
Farhat, C.; Lesoinne, M.; Maman, N.: Mixed explicit/implicit time integration of coupled aeroelastic problems: Three-field formulation, geometric conservation and distributed solution. In: *International Journal for Numerical Methods in Fluids* 21 (1995), S. 807–835

Farhat u. a. 2006
Farhat, C.; van der Zee, K.G.; Geuzaine, P.: Provably second-order time-accurate loosely-coupled solution algorithms for transient nonlinear computational aeroelasticity. In: *Computer Methods in Applied Mechanics and Engineering* 195 (2006), S. 1973–2001

Fedorenko 1964
Fedorenko, R.P.: The speed of convergence of one iterative process. In: *USSR Computational Mathematics and Mathematical Physics* 4 (1964), S. 227–235

Felippa und Park 1980
Felippa, C.A.; Park, K.C.: Staggered transient analysis procedures for coupled mechanical systems: Formulation. In: *Computer Methods in Applied Mechanics and Engineering* 24 (1980), S. 61–111

Felippa u. a. 1977
Felippa, C.A.; Park, K.C.; de Runtz, J.A.: Stabilization of staggered solution procedures for fluid-structure interaction analysis. In: Belytschko, T.; Geers, T.L. (Hrsg.): *Computational Methods for Fluid-Structure Interaction Problems, AMD Vol. 26, American Society of Mechanical Engineers, New York*, 1977, S. 95–124

Felippa u. a. 1998
Felippa, C.A.; Park, K.C.; Farhat, C.: Partitioned analysis of coupled systems. In: Idelsohn, S.R.; Oñate, E.; Dvorkin, E.N. (Hrsg.): *Proceedings of the WCCM IV, CIMNE, Barcelona*, 1998

Felippa u. a. 2001
Felippa, C.A.; Park, K.C.; Farhat, C.: Partitioned analysis of coupled mechanical systems. In: *Computer Methods in Applied Mechanics and Engineering* 190 (2001), S. 3247–3270

Fernández u. a. 2007
Fernández, M.A.; Gerbeau, J.-F.; Grandmont, C.: A projection semi-implicit scheme for the coupling of an elastic structure with an incompressible fluid. In: *International Journal for Numerical Methods in Engineering* 69 (2007), S. 794–821

Fernández und Moubachir 2005
Fernández, M.A.; Moubachir, M.: A Newton method using exact jacobians for solving fluid-structure coupling. In: *Computers & Structures* 83 (2005), S. 127–142

Ferziger und Peric 1997
Ferziger, J.H.; Peric, M.: *Computational techniques for fluid dynamics, Vol. I & II.* Springer-Verlag, Berlin, Heidelberg, 1997

Fletcher 1991
Fletcher, C.A.J.: *Computational methods for fluid dynamics.* Springer-Verlag, Berlin, Heidelberg, 1991

Franca und Farhat 1995
Franca, L.P.; Farhat, C.: Bubble functions prompt unusal stabilized finite element methods. In: *Computer Methods in Applied Mechanics and Engineering* 123 (1995), S. 299–308

Franca und Valentin 2000
Franca, L.P.; Valentin, F.: On an improved unusual stabilized finite element method for the advective-reactive-diffusive equation. In: *Computer Methods in Applied Mechanics and Engineering* 190 (2000), S. 1785–1800

Förster 2007
Förster, Ch.: *Robust methods for fluid-structure interaction with stabilised finite elements*, Bericht Nr. 51, Institut für Baustatik und Baudynamik, Universität Stuttgart, Dissertation, 2007

Förster u. a. 2006a
Förster, Ch.; Genkinger, S.; Neumann, M.; Wall, W.A.; Ramm, E.: Fluid-structure interaction of incompressible flows and slender structures. In: Helmig, R.; Mielke, A.; Wohlmuth, B.I. (Hrsg.): *Multifield Problems in Solid and Fluid Mechanics.* Lecture Notes in Applied and Computational Mechanics (LNACM), 28, Springer-Verlag, Berlin, Heidelberg, 2006, S. 187–218

Förster u. a. 2006b
Förster, Ch.; Wall, W.A.; Ramm, E.: On the geometric conservation law in transient flow calculations on deforming domains. In: *International Journal for Numerical Methods in Fluids* 50 (2006), S. 1369–1379

Förster u. a. 2007
Förster, Ch.; Wall, W.A.; Ramm, E.: Artificial added mass instabilities in sequential staggered coupling of nonlinear structures and incompressible flows. In: *Computer Methods in Applied Mechanics and Engineering* 196 (2007), S. 1278–1293

Galle und Schoenemeyer 2005
Galle, M.; Schoenemeyer, T.: *Basics on vectorization.* NEC High Performance Computing, Stuttgart, 2005

Gee 2004
Gee, M.W.: *Effiziente Lösungsstrategien in der nichtlinearen Schalenmechanik*, Bericht Nr. 43, Institut für Baustatik, Universität Stuttgart, Dissertation, 2004

Gee u. a. 2008
Gee, M.W.; Hu, J.J.; Tuminaro, R.S.: A new smoothed aggregation multigrid method for anisotropic problems. In: *Numerical Linear Algebra with Applications* (2008). – DOI: 10.1002/nla.593

Gee u. a. 2007
Gee, M.W.; Küttler, U.; Wall, W.A.: Advances in algebraic multigrid in fluid structure interaction simulations. In: *Proceedings of US National Congress on Computational Mechanics USNCCM9, July 2007, San Francisco*, 2007

Gee u. a. 2005
Gee, M.W.; Ramm, E.; Wall, W.A.: Parallel multilevel solution of nonlinear shell structures. In: *Computer Methods in Applied Mechanics and Engineering* 194 (2005), S. 2513–2533

Geller u. a. 2005
Geller, S.; Tölke, J.; Krafczyk, M.; Scholz, D.; A., Düster; Rank, E.: Simulation of bidirectional fluid-structure interaction based on explicit coupling approaches of Lattice Boltzmann and p-FEM solvers. In: *Proceedings of the International Conference on Computational Methods for Coupled Problems in Science and Engineering*, 2005

Gerbeau und Vidrascu 2003
Gerbeau, J.-F.; Vidrascu, M.: A quasi-Newton algorithm based on a reduced model for fluid-structure interaction problems in blood flows. In: *Mathematical Modelling and Numerical Analysis* 37 (2003), S. 631–647

GiD 2008
GiD, The personal pre and postprocessor: *International Center for Numerical Methods in Engineering (CIMNE).* 2008. – URL `http://gid.cimne.upc.es/`

Girault und Raviert 1986
Girault, V.; Raviert, P.-A.: *Finite element methods for Navier-Stokes equations – Theory and algorithms.* Springer-Verlag, Berlin, Heidelberg, 1986

Glück 2002
Glück, M.: *Ein Beitrag zur numerische Simulation von Fluid-Struktur-Interaktion – Grundlagenuntersuchungen und Anwendung auf Membrantragwerke*, Technische Fakultät der Universität Erlangen-Nürnberg, Dissertation, 2002

Glück u. a. 2003
Glück, M.; Breuer, M.; Durst, F.; Halfmann, A.; Rank, E.: Computation of wind-induced vibrations of flexible shells and membranous structures. In: *Journal of Fluids and Structures* 17 (2003), S. 739–765

Gordon und Hall 1973
Gordon, W.J.; Hall, C.A.: Construction of curvilinear co-ordinate systems and application to mesh generation. In: *International Journal for Numerical Methods in Engineering* 7 (1973), S. 461–477

Grasberger 2003
Grasberger, S.: *Gekoppelte hygro-mechanische Materialmodellierung und numerische Simulation langzeitiger Degradation von Betonstrukturen*, Institut für Konstruktiven Ingenieurbau, Ruhr-Universität Bochum, Dissertation, 2003

Grasberger u. a. 2003
Grasberger, S.; Neumann, M.; Meschke, G.: Numerische Dauerhaftigkeitsanalyse von Betonstrukturen am Beispiel einer Tunnelinnenschale. In: *Der Bauingenieur* 78 (2003), S. 411–421

Gravemeier 2003
Gravemeier, V.: *The variational multiscale method for laminar and turbulent incompressible flow*, Bericht Nr. 40, Institut für Baustatik, Universität Stuttgart, Dissertation, 2003

Greenshields und Weller 2005
Greenshields, C.J.; Weller, H.G.: A unified formulation for continuum mechanics applied to fluid-structure interaction in flexible tubes. In: *International Journal for Numerical Methods in Engineering* 64 (2005), S. 1575–1593

Gresho 1992
Gresho, P.M.: Some interesting issues in incompressible fluid dynamics, both in the continuum and in numerical simulation. In: *Advances in Applied Mechanics* 28 (1992), S. 45–140

Gresho und Sani 1998
Gresho, P.M.; Sani, R.L.: *Incompressible flow and the finite element method.* John Wiley & Sons, Chichester, 1998

Gunzburger 1989
Gunzburger, M.D.: *Finite element methods for viscous incompressible flows.* Academic Press Inc., San Diego, 1989

Hackbusch 1985
Hackbusch, W.: *Multi-Grid Methods and Applications.* Springer-Verlag, Berlin, Heidelberg, 1985

Hackbusch 1993
Hackbusch, W.: *Iterative Lösung großer schwachbesetzter Gleichungssysteme: mit Beispielen und Übungsaufgaben.* Teubner, Stuttgart, 1993

Halfmann u. a. 2000
Halfmann, A.; Rank, E.; Glück, M.; Breuer, M.; Durst, F.: A partitioned solution approach for the fluid-structure interaction of wind and thin-walled structures. In: *Proceedings of the International Conference on Application of Computer Science and Mathematics in Architecture and Civil Engineering*, 2000

Halfmann u. a. 2001
Halfmann, A.; Rank, E.; Glück, M.; Breuer, M.; Durst, F.; Bellmann, J.; Katz, C.: Computational engineering for wind-exposed thin-walled structures. In: *Proceedings of the 3rd International Fortwihr Conference, Erlangen*, 2001

Hansbo und Szepessy 1990
Hansbo, P.; Szepessy, A.: A velocity-pressure streamline diffusion finite element method for the incompressible Navier-Stokes equations. In: *Computer Methods in Applied Mechanics and Engineering* 84 (1990), S. 175–192

Hartmann 2000
Hartmann, D.: *Theoretische Grundlagen des CAD.* Skripte aus dem Lehrstuhl für Ingenieurinformatik im Bauwesen, Ruhr-Universität Bochum, 2000

Hartmann 2007
Hartmann, S.: *Kontaktanalyse dünnwandiger Strukturen bei großen Deformationen*, Bericht Nr. 49, Institut für Baustatik und Baudynamik, Universität Stuttgart, Dissertation, 2007

Heil 2004
Heil, M.: An efficient solver for the fully coupled solution of large-displacement fluid-structure interaction problems. In: *Computer Methods in Applied Mechanics and Engineering* 193 (2004), S. 1–23

Hestenes und Stiefel 1952
Hestenes, M.R.; Stiefel, E.: Methods of conjugate gradients for solving linear systems. In: *NBS Journal of Research* 49 (1952), S. 409–436

Heywood u. a. 1996
Heywood, J.G.; Rannacher, R.; Turek, S.: Artificial boundaries and flux and pressure conditions for the incompressible Navier-Stokes equations. In: *International Journal for Numerical Methods in Fluids* 22 (1996), S. 325–352

Hofman 2003
Hofman, J.M.A.: Control-fluid interaction in air-conditioned aircraft cabins. A demonstration of stability analysis for partitioned dynamical systems. In: *Computer Methods in Applied Mechanics and Engineering* 192 (2003), S. 4947–4963

Holzapfel 2000
Holzapfel, G.A.: *Nonlinear solid mechanics – A continuum approach for engineering.* John Wiley & Sons, Chichester, 2000

Hughes 2000
Hughes, T.J.R.: *The finite element method.* Dover Publications, Mineola, 2000

Hughes und Brooks 1976
Hughes, T.J.R.; Brooks, A.: A multi-dimensional upwind scheme with no cross wind diffusion. In: Hughes, T.J.R. (Hrsg.): *Finite element for convection dominated flows.* Vol. 34, ASME, AMD, 1976, S. 19–35

Hughes und Brooks 1982
Hughes, T.J.R.; Brooks, A.: A theoretical framework for Petrov-Galerkin methods with discontinuous weighting functions. In: Gallagher, R.H. (Hrsg.): *Finite elements in fluids, Vol. 4.* John Wiley & Sons, Chichester, 1982, S. 47–65

Hughes u. a. 1986
Hughes, T.J.R.; Franca, P.; Balestra, M.: A new finite element formulation for computational fluid dynamics: V. Circumventing the Babuska-Brezzi condition: A stable Petrov-Galerkin formulation of the Stokes problem accommodating equal-order interpolation. In: *Computer Methods in Applied Mechanics and Engineering* 59 (1986), S. 85–99

Hughes u. a. 1989
Hughes, T.J.R.; Franca, P.; Hulbert, G.M.: A new finite element formulation for computational fluid dynamics: VIII. The Galerkin/Least-Squares method for advective-diffusive equations. In: *Computer Methods in Applied Mechanics and Engineering* 73 (1989), S. 173–189

Hughes u. a. 1981
Hughes, T.J.R.; Liu, W.K.; Zimmermann, T.K.: Lagrangian-Eulerian finite element formulation. In: *Computer Methods in Applied Mechanics and Engineering* 178 (1981), S. 343–366

Hübner 2003
Hübner, B.: *Simultane Analyse von Bauwerks-Wind-Wechselwirkungen*, Institut für Statik, Technische Universität Braunschweig, Dissertation, 2003

Hübner und Dinkler 2005
Hübner, B.; Dinkler, D.: A simultaneous solution procedure for strong interactions of generalized Newtonian fluids and viscoelastic solids at large strains. In: *International Journal for Numerical Methods in Engineering* 64 (2005), S. 920–939

Hübner u. a. 2004
Hübner, B.; Walhorn, E.; Dinkler, D.: A monolithic approach to fluid-structure interaction using space-time finite elements. In: *Computer Methods in Applied Mechanics and Engineering* 193 (2004), S. 2087–2104

Irons und Tuck 1969
Irons, B.; Tuck, R.C.: A version of the Aitken accelerator for computer implementation. In: *International Journal for Numerical Methods in Engineering* 1 (1969), S. 275–277

Kalro und Tezduyar 2000
Kalro, V:; Tezduyar, T.E.: A parallel 3D computational method for fluid-structure interactions in parachute systems. In: *Computer Methods in Applied Mechanics and Engineering* 190 (2000), S. 321–332

Katz u. a. 2000
Katz, C.; Mayr-Knoch, W.; Durst, F.; Rank, E.; Zenger, C.: Numerische Berechnung der Fluid-Struktur-Wechselwirkung auf Vektor-Parallelrechnern mit verteiltem Speicher. FORTWIHR-Abschlussbericht / Lehrstuhl für Bauinformatik, Technische Universität München. 2000. – Forschungsbericht

Knabner und Angermann 2000
Knabner, P.; Angermann, L.: *Numerik partieller Differentialgleichungen. Eine anwendungsorientierte Einführung.* Springer-Verlag, Berlin, Heidelberg, 2000

Knoll und Keyes 2004
Knoll, D. A.; Keyes, D. E.: Jacobian-free Newton-Krylov methods: A survey of approaches and applications. In: *Journal of Computational Physics* 193 (2004), S. 357–397

Kollmannsberger u. a. 2006
Kollmannsberger, S.; Scholz, D.; Düster, A.; Rank, E.: FSI based on bidirectional coupling of high order solids to a Lattice-Boltzmann method. In: *Proceedings of the ASME Pressure Vessels and Piping Division Conference*, 2006

Krause und Rank 2003
Krause, R.; Rank, E.: Multiscale computations with a combination of the h- and p-version of the finite-element method. In: *Computer Methods in Applied Mechanics and Engineering* 192 (2003), S. 3959–3983

Kuhl 1996
Kuhl, D.: *Stabile Zeitintegrationsalgorithmen in der nichtlinearen Elastodynamik dünnwandiger Tragwerke*, Bericht Nr. 22, Institut für Baustatik, Universität Stuttgart, Dissertation, 1996

Kuhl und Crisfield 1999
Kuhl, D.; Crisfield, M.A.: Energy-conserving and decaying algorithms in non-linear structural dynamics. In: *International Journal for Numerical Methods in Engineering* 45 (1999), S. 569–599

Lanczos 1952
Lanczos, C.: Solution of systems of linear equations by minimized iterations. In: *Journal of the National Bureau of Standards* 49 (1952), S. 33–53

Le Tallec und Mouro 1998
Le Tallec, P.; Mouro, J.: Fluid structure interaction with large structural displacements. In: Papailiou, K.D. (Hrsg.): *Proceedings of the 4th ECCOMAS Computational Fluid Dynamic Conference, Athens*, John Wiley & Sons, Chichester, 1998, S. 1032–1040

Le Tallec und Mouro 2001
Le Tallec, P.; Mouro, J.: Fluid structure interaction with large structural displacements. In: *Computer Methods in Applied Mechanics and Engineering* 190 (2001), S. 3039–3067

Löhner und Galle 2002
Löhner, R.; Galle, M.: Minimization of indirect addressing for edge-based field solvers. In: *Communications in Numerical Methods in Engineering* 18 (2002), S. 335–343

Marques und Pereira 2004
Marques, N.P.C.; Pereira, J.C.F: Comparison of matrix-free acceleration techniques in compressible Navier-Stokes calculations. In: *International Journal for Numerical Methods in Engineering* 61 (2004), S. 455–474

Marsden und Hughes 1983
Marsden, J.E.; Hughes, T.J.R.: *Mathematical foundations of elasticity.* Prentice-Hall, Englewood Cliffs, 1983

Matthies und Steindorf 2003
Matthies, H.G.; Steindorf, J.: Partitioned strong coupling algorithms for fluid-structure interaction. In: *Computers & Structures* 81 (2003), S. 805–812

Meister 1999
Meister, A.: *Numerik linearer Gleichungssysteme.* Vieweg, Braunschweig, Wiesbaden, 1999

Menter 1994
Menter, F.R.: Two-equation eddy-viscosity turbulence models for engineering applications. In: *AIAA Journal* 32 (1994), S. 1598–1605

Michler 2005
Michler, Ch.: *Efficient numerical methods for fluid-structure interaction*, Technische Universiteit Delft, Dissertation, 2005

Michler u. a. 2005
Michler, Ch.; van Brummelen, E.H.; de Borst, R.: An interface Newton-Krylov solver for fluid-structure interaction. In: *International Journal for Numerical Methods in Fluids* 47 (2005), S. 1189–1195

Michler u. a. 2006
Michler, Ch.; van Brummelen, E.H.; de Borst, R.: Error-amplification analysis of subiteration-preconditioned GMRES for fluid-structure interaction. In: *Computer Methods in Applied Mechanics and Engineering* 195 (2006), S. 2124–2148

Mok 2001
Mok, D.P.: *Partitionierte Lösungsansätze in der Strukturdynamik und der Fluid-Struktur-Interaktion*, Bericht Nr. 36, Institut für Baustatik, Universität Stuttgart, Dissertation, 2001

Mok und Wall 2001
Mok, D.P.; Wall, W.A.: Partitioned analysis schemes for the transient interaction of incompressible flows and nonlinear flexible structures. In: Wall, W.A.; Bletzinger, K.-U.; Schweitzerhof, K. (Hrsg.): *Proceedings of Trends in Computational Structural Mechanics*, 2001, S. 689–698

Nassi und Shneiderman 1973
Nassi, I; Shneiderman, B: Flowchart techniques for structured programming. In: *ACM SIGPLAN Notices* 8 (1973), S. 12–26

Neumann u. a. 2006
Neumann, M.; Tiyyagura, S.R.; Wall, W.A.; Ramm, E.: Robustness and efficiency aspects for computational fluid structure interaction. In: Krause, E.; Shokin, Y.I.; Resch, M.; Shokina, N. (Hrsg.): *Computational Science and High Performance Computing II*. Notes on Numerical Fluid Mechanics and Multidisciplinary Design (NNFM), 91, Springer-Verlag, Berlin, Heidelberg, 2006, S. 99–114

Newmark 1959
Newmark, N.M.: A method of computation for structural dynamics. In: *Journal of the Engineering Mechanics Division* 85 (1959), S. 67–94

Ohayon und Felippa 2001
Ohayon, R.; Felippa, C.A.: Special Issue: Advances in computational methods for fluid-structure interaction and coupled problems. In: *Computer Methods in Applied Mechanics and Engineering* 190 (2001)

Ohayon und Kvamsdal 2006
Ohayon, R.; Kvamsdal, T.: Special Issue on Fluid-Structure Interaction. In: *Computer Methods in Applied Mechanics and Engineering* 195 (2006)

Oliker u. a. 2003
Oliker, L.; Canning, A.; Carter, J.; Shalf, J.; Skinner, D.; Ethier, S.; Biswas, R.; Djomehri, J.; van der Wijngaart, R.: Evaluation of cache-based superscalar and cacheless vector architectures for scientific computations. In: *Proceedings of the ACM/IEEE Supercomputing Conference 2003, Phoenix, Arizona, USA*, 2003

Oñate 1998
Oñate, E.: Derivation of stabilized equations for numerical solution of advective-diffusive transport and fluid flow problems. In: *Computer Methods in Applied Mechanics and Engineering* 151 (1998), S. 233–265

Papadrakakis und Allix 2008
Papadrakakis, M.; Allix, O.: Special Issue on Computational Methods in Fluid-Structure Interaction. In: *Computer Methods in Applied Mechanics and Engineering* 197 (2008)

Park und Felippa 1983
Park, K.C.; Felippa, C.A.: Partitioned analysis of coupled systems. In: Belytschko, T.; Hughes, T.J.R. (Hrsg.): *Computational Methods for Transient Analysis*. Elsevier Science Publishers B.V., 1983, S. 157–219

Piperno 1997
Piperno, S.: Explicit/implicit fluid/structure staggered procedures with a structural

predictor and fluid subcycling for 2D inviscid aeroelastic simulations. In: *International Journal for Numerical Methods in Fluids* 25 (1997), S. 1207–1226

Piperno u. a. 1995
Piperno, S.; Farhat, C.; Larrouturou, B.: Partitioned procedures for the transient solution of coupled aeroelastic problems. Part I: Model problem, theory and two-dimensional application. In: *Computer Methods in Applied Mechanics and Engineering* 124 (1995), S. 79–112

Plato 2000
Plato, R.: *Numerische Mathematik kompakt.* Vieweg, Braunschweig, Wiesbaden, 2000

Pohl u. a. 2004
Pohl, T.; Deserno, F.; Thürey, N.; Rüde, U.; Lammers, P.; Wellein, G.; Zeiser, T.: Performance evaluation of parallel large-scale Lattice Boltzmann applications on three supercomputing architectures. In: *Proceedings of the ACM/IEEE Supercomputing Conference 2004, Pittsburgh, USA*, 2004

Quarteroni 1991
Quarteroni, A.: Domain decomposition methods and parallel processing for the numerical solution of partial differential equations. In: *Surveys on Mathematics for Industry* 1 (1991), S. 75–118

Quarteroni u. a. 2002
Quarteroni, A.; Sacco, R.; Saleri, F.: *Numerische Mathematik 1.* Springer-Verlag, Berlin, Heidelberg, 2002

Ramm u. a. 2008
Ramm, E.; von Scheven, M.; Förster, Ch.; Wall, W.A.: Interaction of incompressible flows and thin-walled structures. In: *ECCOMAS Multidisciplinary Jubilee Symposium. New Computational Challenges in Materials, Structures and Fluids.* Computational Methods in Applied Sciences, 14, Springer-Verlag, Berlin, Heidelberg, 2008

Rank 1992
Rank, E.: Adaptive remeshing and *h-p* domain decomposition. In: *Computer Methods in Applied Mechanics and Engineering* 101 (1992), S. 299–313

Rank u. a. 1993
Rank, E.; Schweingruber, M.; Sommer, M.: Adaptive mesh generation and tranformation of triangular to quadrilateral meshes. In: *Communications in Numerical Methods in Engineering* 9 (1993), S. 121–129

Rugonyi und Bathe 2000
Rugonyi, S.; Bathe, K.-J.: On the analysis of fully coupled fluid flows with structural interactions: A coupling and condensation procedure. In: *International Journal for Computional Civil and Structural Engineering* 1 (2000), S. 29–41

Saad und Schulz 1986
Saad, Y.; Schulz, M.H.: GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. In: *SIAM Journal on Scientific and Statistical Computing* 7 (1986), S. 856–869

Saad und van der Vorst 2000
Saad, Y.; van der Vorst, H.: Iterative solution of linear systems in the 20th century. In: *Journal of Computational and Applied Mathematics* 123 (2000), S. 1–33

Schmidberger 2006
Schmidberger, M.: *Partitionierte Verfahren zur Lösung von Fluid-Struktur-Interaktion*, Lehrstuhl für Numerische Mechanik, Technische Universität München, Diplomarbeit, 2006

Schulz u. a. 2008
Schulz, K.; Klinkel, S.; Wagner, W.: A geometrically nonlinear finite shell element for the analysis of piezoelectric smart structures. In: *Proceedings of 8th World Congress on Computational Mechanics WCCM8, Venedig*, 2008

Schäfer und Turek 1996
Schäfer, M.; Turek, S.: Benchmark computations of laminar flow around a cylinder. In: Hirschel, E.H. (Hrsg.): *Flow Simulation with High-Performance Computers II.* Notes on Numerical Fluid Mechanics, Vol. 52, Vieweg, 1996, S. 547–566

Sonneveld 1989
Sonneveld, P.: CGS, a fast Lanczos-type solver for nonsymmetric linear systems. In: *SIAM Journal on Scientific and Statistical Computing* 10 (1989), S. 36–52

Souläimani u. a. 2002
Souläimani, A.; Salah, N.B.; Saad, Y.: Enhanced GMRES acceleration techniques for some CFD problems. In: *International Journal of Computational Fluid Dynamics* 16 (2002), S. 1–20

Spieth 1999
Spieth, H.C.: *Iterative Löser für unsymmetrische Gleichungssysteme aus FE-Simulationen inkompressibler Strömungen.* Institut für Baustatik, Universität Stuttgart, Seminararbeit, 1999

Stein und Barthold 1996
Stein, E.; Barthold, F.-J.: Elastizitätstheorie. In: Mehlhorn, G. (Hrsg.): *Der Ingenieurbau, Grundwissen: Werkstoffe, Elastizitätstheorie*. Ernst & Sohn, Berlin, 1996, S. 165–428

Stein u. a. 1998
Stein, K.; Benney, R.; Kalro, V.; Tezduyar, T.E.; Leonard, J.; Accorsi, M.: *Parachute fluid-structure interactions: Coupling issues*. University of Minnesota Supercomputer Institute Research Report Nr. 98/70, 1998

Stein u. a. 2000
Stein, K.; Benney, R.; Kalro, V.; Tezduyar, T.E.; Leonard, J.; Accorsi, M.: Parachute fluid-structure interactions: 3-D computation. In: *Computer Methods in Applied Mechanics and Engineering* 190 (2000), S. 373–386

Steindorf 2002
Steindorf, J.: *Partitionierte Verfahren für Probleme der Fluid-Struktur-Wechselwirkung*, Fachbereich für Mathematik und Informatik der Technischen Universität Braunschweig, Dissertation, 2002

Sternel u. a. 2008
Sternel, D.C.; Schäfer, M.; Heck, M.; Yigit, S.: Efficiency and accuracy of fluid-structure interaction simulations using an implicit partitioned approach. In: *Computational Mechanics* (2008). – DOI: 10.1007/s00466-008-0278-y

Stüben 2001
Stüben, K.: A review of algebraic multigrid. In: *Journal of Computational and Applied Mathematics* 128 (2001), S. 281–309

SX-8 Homepage 2008
SX-8 Homepage: *Höchstleistungsrechenzentrum Stuttgart (HLRS)*. 2008. – URL `http://www.hlrs.de/hw-access/platforms/sx8/`

Teraflop Workbench Homepage 2008
Teraflop Workbench Homepage: *Höchstleistungsrechenzentrum Stuttgart (HLRS)*. 2008. – URL `http://www.teraflop-workbench.de/`

Tezduyar 2004
Tezduyar, T.E.: Finite element methods for fluid dynamics with moving boundaries and interfaces. In: Stein, E.; de Borst, R.; Hughes, T.J.R. (Hrsg.): *Encyclopedia of Computational Mechanics, Volume 3: Fluids*. John Wiley & Sons, Chichester, 2004, S. 545–578

Tezduyar u. a. 1996
Tezduyar, T.E.; Aliabadi, S.; Behr, M.; Johnson, A.; Kalro, V.; Litke, M.: Flow simulation and high performance computing. In: *Computational Mechanics* 18 (1996), S. 397–412

Tezduyar u. a. 1992
Tezduyar, T.E.; Mittal, S.; Ray, S.E.; Shih, R.: Incompressible flow computations with stabilized bilinear and linear equal-order-interpolation velocity-pressure elements. In: *Computer Methods in Applied Mechanics and Engineering* 95 (1992), S. 221–242

Tiyyagura u. a. 2006
Tiyyagura, S.R.; Küster, U.; Borowski, S.: Performance Improvement of Sparse Matrix Vector Product on Vector Machines. In: Alexandrov, V.; van Albada, D.G.; Sloot, P.M.A.; Dongarra, J. (Hrsg.): *Proceedings of the Sixth International Conference on Computational Science (ICCS 2006).* Reading, UK : Springer-Verlag, Berlin, Heidelberg, 2006, S. 196–203

Tiyyagura und Küster 2007
Tiyyagura, S.R.; Küster, U.: Block-based approach to solving linear systems. In: Shi, Y.; van Albada, G.D.; Dongarra, J.; Sloot, P.M.A. (Hrsg.): *Computational Science - ICCS 2007. 7th International Conference, Beijing China, May 27-30, 2007, Proceedings, Part I*, Lecture Notes in Computer Science, Vol 4487, Springer-Verlag, Berlin, Heidelberg, 2007, S. 128–135

Tiyyagura und von Scheven 2007
Tiyyagura, S.R.; von Scheven, M.: FSI simulations on vector systems - Development of a linear iterative solver (BLIS). In: Resch, M.; Roller, S.; Lammers, P.; Furui, T.; Galle, M.; Bez, W. (Hrsg.): *High Performance Computing on Vector Systems 2007. Proceedings of the 6th Teraflop Workshop, Stuttgart, Germany.* Springer-Verlag, Berlin, Heidelberg, 2007, S. 167–177

Tonti 1975
Tonti, E.: *On the formal structure of physical theories.* Monograph of the Italian Research Council, 1975

Turek 1998
Turek, S.: *Konsequenzen eines numerischen „Elch Tests“ für Computersimulationen.* Universität Heidelberg, 1998. – Preprints SFB 359, Nummer 98-46

Turek 1999
Turek, S.: *Trends in processor technology and their impact on Numerics for PDE‘s.* Universität Heidelberg, 1999. – Preprints SFB 359, Nummer 99-31

Turner u. a. 1956
Turner, M.J.; Clough, R.W.; Martin, H.C.; Topp, L.J.: Stiffness and deflection analysis of complex structures. In: *J Aero Sci* 23 (1956), S. 805–823

van der Vorst 1992
van der Vorst, H.A.: Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric systems. In: *SIAM Journal on Scientific and Statistical Computing* 13 (1992), S. 631–644

van Zuijlen u. a. 2007
van Zuijlen, A.H.; Bosscher, S.; Bijl, H.: Two level algorithms for partitioned fluid-structure interaction computations. In: *Computer Methods in Applied Mechanics and Engineering* 196 (2007), S. 1458–1470

Veldhuizen 1997
Veldhuizen, T.L.: Scientific computing: C++ versus Fortran: C++ has more than caught up. In: *Dr. Dobb's Journal of Software Tools* 22 (1997), S. 36–38

Veldhuizen und Jernigan 1997
Veldhuizen, T.L.; Jernigan, M.E.: Will C++ be faster than Fortran? In: *Proceedings of the 1st International Scientific Computing in Object-Oriented Parallel Environments (ISCOPE'97)*, Springer-Verlag, Berlin, Heidelberg, 1997

Vierendeels 2005
Vierendeels, J.A.: Implicit coupling of partitioned fluid-structure interaction solvers using reduced order models. In: *35th AIAA Fluid Dynamics Conference and Exhibit, Toronto, Canada*, 2005

Vierendeels 2006
Vierendeels, J.A.: Strong coupling of partitioned fluid-structure interaction problems with reduced order models. In: *European Conference on Computational Fluid Dynamics, ECCOMAS CFD*, 2006

Vierendeels u. a. 2007
Vierendeels, J.A.; Lanoye, L.; Degroote, J.; Verdonck, P.: Implicit coupling of partitioned fluid-structure interaction problems with reduced order models. In: *Computers & Structures* 85 (2007), S. 970–976

von Scheven u. a. 2007
von Scheven, M.; Tiyyagura, S.R.; Ramm, E.; Bischoff, M.: Efficiency issues of partitioned solution in fluid-structure interaction. In: *Proceedings of the Int. Conf. on Computational Methods for Coupled Problems in Science and Engineering, Ibiza*, 2007

Vrahatis u. a. 2003
Vrahatis, M.N.; Magoulas, G.D.; Plagianakos, V.P.: From linear to nonlinear iterative methods. In: *Applied Numerical Mathematics* 45 (2003), S. 59–77

Vuik u. a. 2001
Vuik, C.; Frank, J.; Segal, A.: A parallel block-preconditioned GCR method for incompressible flow problems. In: *Future Generation Computer Systems* 18 (2001), S. 31–40

Vuik und Saghir 2002
Vuik, C.; Saghir, A.: *The Krylov accelerated SIMPLE(R) method for incompressible flow*. Technische Universiteit Delft, Report 02-01, 2002

Vuik u. a. 2000
Vuik, C.; Saghir, A.; Boerstoel, G.P.: The Krylov accelerated SIMPLE(R) methods for flow problems in industrial furnaces. In: *International Journal for Numerical Methods in Fluids* 33 (2000), S. 1027–1040

Wagner 1998
Wagner, C.: *Introduction to algebraic multigrid*. Interdisziplinäres Zentrum für wissenschaftliches Rechnen, Universität Heidelberg, 1998

Walhorn u. a. 2003
Walhorn, E.; Hübner, B.; Kölke, A.; Dinkler, D.: Fluid-structure coupling within a monolithic model involving free surface flows. In: Bathe, K.-J. (Hrsg.): *Computational Fluid and Solid Mechanics* Bd. 2, 2003, S. 1560–1563

Wall 1999
Wall, W.A.: *Fluid-Struktur-Interaktion mit stabilisierten Finiten Elementen*, Bericht Nr. 31, Institut für Baustatik, Universität Stuttgart, Dissertation, 1999

Wall u. a. 1999
Wall, W.A.; Mok, D.P.; Ramm, E.: Partitioned analysis approach of the transient coupled response of viscous fluids and flexible structures. In: Wunderlich, W. (Hrsg.): *Solids, Structures and Coupled Problems in Engineering, Proceedings of the European Conference on Computational Mechanics ECCM '99, Munich*, 1999

Wall und Ramm 1998
Wall, W.A.; Ramm, E.: Fluid-structure interaction based upon a stabilized (ALE) finite element method. In: Oñate, E.; Idelsohn, S. (Hrsg.): *Computational Mechanics, Proceedings of the Fourth World Congress on Computational Mechanics WCCM IV, Buenos Aires, Argentina*, 1998

Warsi 1993
Warsi, Z.U.A.: *Fluid dynamics – Theoretical and computational approaches*. CRC Press, Boca Raton, 1993

Wenisch u. a. 2007
Wenisch, P.; van Treeck, Ch.; Borrmann, A.; Rank, E.; Wenisch, O.: Computational steering on distributed systems: Indoor comfort simulations as a case study of interactive CFD on supercomputers. In: *The International Journal of Parallel, Emergent and Distributed Systems* 22 (2007), S. 275–291

Wikipedia 2008
Wikipedia: *Die freie Enzyklopädie*. 2008. – URL `http://www.wikipedia.de/`

Wriggers 2001
Wriggers, P.: *Nichtlineare Finite-Element-Methoden*. Springer-Verlag, Berlin, Heidelberg, 2001

Wüchner 2007
Wüchner, R.: *Mechanik und Numerik der Formfindung und Fluid-Struktur-Interaktion von Membrantragwerken*, Lehrstuhl für Statik, Technische Universität München, Dissertation, 2007

Zhang 2000
Zhang, J.: Preconditioned Krylov subspace methods for solving nonsymmetric matrices from CFD applications. In: *Computer Methods in Applied Mechanics and Engineering* 189 (2000), S. 825–840

Zhang und Hisada 2004
Zhang, Q.; Hisada, T.: Studies of the strong coupling and weak coupling methods in FSI analysis. In: *International Journal for Numerical Methods in Engineering* 60 (2004), S. 2013–2029

Zheng und Liou 2003
Zheng, Y.; Liou, M.-S.: A novel approach of three-dimensional hybrid grid methodology: Part 1. Grid generation. In: *Computer Methods in Applied Mechanics and Engineering* 192 (2003), S. 4147–4171

Zienkiewicz und Cheung 1967
Zienkiewicz, O.C.; Cheung, Y.K.: *The finite element method in structural and continuum mechanics*. McGraw-Hill, London, 1967

Zienkiewicz und Phillips 1971
Zienkiewicz, O.C.; Phillips, D.V.: An automatic mesh generation scheme for plane

and curved surfaces by isoparametric coordinates. In: *International Journal for Numerical Methods in Engineering* 3 (1971), S. 519–528

Zienkiewicz u. a. 2005
Zienkiewicz, O.C.; Taylor, R.L.; Zhu, J.Z.: *The finite element method, Band I, II & III*. Elsevier Butterworth-Heinemann, Oxford, 2005

Index

A

Abbruchkriterium 119, 129, 154, 161
„added mass“-Effekt 115, 139, 141
Adressierung, indirekte 54, 83, 86
Advektions-Diffusions-Gleichung 25
Aitken, Alexander Craig 120
Aitken-Verfahren 120, **123**, 125, 131 f., 134, 143
ALE-Formulierung 17, 19 f., 23, 32, 153, 159
Anfangs-Randwert-Problem 8, **9**, 10, 12, 18, **20**, 22, 24
Anfangsbedingung **10**, 20, **21**, 29, 63
Ansatzfunktion 12, 24, 42, 45, 48, 57 ff., 129 f., 149
Arbeit, virtuelle 11
Arbitrary Lagrangean Eulerian 17
Array 54, 59 f., 62 f.
„artificial added mass“-Effekt 114, **115**, 116, 123, 142
„artificial added mass“-Instabilität 141
Assemblierung **12**, 57 f.

B

B-Spline 47, **47**
  Basisfunktion 48 ff.
    rationale 50
  non-uniform rational 47
  uniform 49
Bézier, Pierre Étienne 46
Bézierkurve 47 ff.
Babuška, Ivo M. 24
Backward Differentiation Formulae 30
Bandbreite 45, 82
BDF2 **30**, 115, 149, 159
Beltrami, Eugenio 63
Bernstein, Sergei Natanowitsch 47
Bernstein-Polynom **47**, 49
Betrachtungsweise
  materielle 6
  räumliche 17
Bi-Lanczos-Algorithmus 78
BiCG-Verfahren 78
BiCGSTAB-Verfahren **77**, 78 f., 92, 94
Bisektionsverfahren 51 f.
BLIS 86
Block-Gauß-Seidel-Verfahren 109, **117**, 119 f., 122 f., 127, 131, 137 f., 140 f., 143 f., 146, 154, 161
Block-Iterations-Verfahren **117**, 121, 141 ff.
Block-Jacobi-Verfahren 83, 119, **121**, 123

Block-SOR-Verfahren 118 f.
Boltzmann, Ludwig 121
Brezzi, Franco 24
Bubnow, I.G. 12
Bubnow-Galerkin-Diskretisierung 12, 24

C

Casteljau, Paul de 46
Cauchy, Augustin Louis 8
Cauchy-Spannungen 18, 106
Cauchy Bewegungsgleichung 8
CG-Verfahren **74**, 80, 86
  mit Neustart 77
  zyklisches 77
CGS-Verfahren 78
„checkerboard modes“ 24
Cluster 35, 86
Compiler 54 f., 60
  Direktive 55
  Option 55

D

Datenparallelität 39, 54 ff., 59 f.
Deformationen, große 6 f., 9
Deformationsgeschwindigkeit 18
Deformationsgradient 6, 8
Diagonalmatrix 70, 81
Diffusion 27
Direkte Steifigkeitsmethode 12
Dirichlet, Johann Peter 10
Dirichlet-Neumann-Partitionierung 105, 141
Dirichlet-Neumann-Substruktur-Verfahren 117, 120
Diskretisierung **12**, 13, **24**, 29, 42, 73, 81, 83, 85, 94, 108, 110, 126 f., 135, 148, 153, 159 f.
Dissipation, numerische 13 f., 30
Dreiecksmatrix 70, 72, 81 f.
Druck 21 f., 24 f., 106, 141, 149, 152
  hydrostatischer 18
  kinematischer 18

E

Eigenwert 73, 77, 80 f., 123
Einschritt-θ-Verfahren 29
Einzelschritt-Verfahren 71
Elastizitätsmodul 9
Elastizitätstensor 9
Elastodynamik 6 ff., 10, 12 f.
Elementlänge, charakteristische 27
Energieerhalt 103, 105, 111, 114, 116, 120
Euler, Leonhard 17
Euler-Rückwärts-Verfahren **29**, 105, 115

F

Feld-Eliminations-Verfahren 103
Feldgleichung 8, 10, 20
  dynamische 8, 11
  kinematische 7, 18
  konstitutive 9
fill-in 68, 82
Finite-Differenzen-Methode 126
Finite-Element-Formulierung 84
Finite-Element-Methode 5, 12, 41
Finite-Element-Programm 53, 56
Finite-Element-Simulation 40, 63, 67
Finite Differenzen 134, **138**, 146
Finite Elemente 12 f., 17, 24, 74, 83, 85, 115
Finite Increment Calculus 25
Fixpunkt 69, 107, 119
Fixpunkt-Iteration 117, 119, 138
FLOPS 37, 39, 86
Fluid **16**, 105, 147

inkompressibles 5, 17 f., 20 f., 24, 102, 114, 143
Newton'sches 18, 106, 148, 159
Fluid-Operator **32**, 104, 118, 121, 125, 130, 133, 136, 138
erweiterter 104
Fluid-Struktur-Interaktion 20, 22, 102 f., 106 f., 116, 122
Fluid-Struktur-Kopplung 75
Fluid-Struktur-Kopplungs-Abbildung 107
Fluid-Struktur-Operator 139
Fluid-Struktur-Problem 140
Fluid-Struktur-Wechselwirkung 6, 16 f., 31, 37, **101**, 104, 107, 110, 117, 119 f., 126 f., 147 f., 158
Fluiddynamik 5, **16**, 104
Fluidfeld 37
Fluidformulierung 83
Fluidgebiet 17, 19 ff., 24, 32, 104 ff., 130, 133, 136, 139, 147 f., 150, 153, 159
Fluidnetz 33
Formänderungsarbeit 9
Fortran 60 ff.
Fourier, Jean Baptiste Joseph 116
Freiformfläche 46, 50
Freiformkurve 46

G

Galerkin, Boris Grigorjewitsch 12
Galerkin-Approximation 27
Galerkin-Bedingung 74
Galerkin-Formulierung 29
Galerkin/Least-Squares 25 f.
Gauß'scher Integralsatz 11, 23
Gauß, Johann Carl Friedrich 11, 67
Gauß-Elimination 67
Gauß-Seidel-Verfahren 67, **71**, 117
rückwärts 72
relaxiertes 72
symmetrisches 72
Generalisiertes-α-Verfahren 12, **13**, 154, 160
„geometric conservation laws" 20
geometrische Bilanzgleichungen **20**, 105, 111
Gesamtschritt-Verfahren 70
Geschwindigkeit, konvektive 20
Geschwindigkeitsfeld 10, 18, 21
Gleichgewicht **8**, 11 f.
Gleichungssystem 80, 83, 147, 154
gekoppeltes 106, 108
lineares 15, 56, 63, 67 f., 74, 77 f., 86, 121, 135, 139
nichtlineares 15, 31, 106, 117, 123
Gleitkommaoperation 37
Gleitkommazahl 39 f.
GLS 25, **26**, 95, 97
GMRES-Verfahren 74, 78, **78**, 84, 86, 140
mit Neustart 79
Gradienten-Operator
materieller 7
räumlicher 18
Gradienten-Verfahren **75**, 124, 133
Green'sche Elastizität 9
Green, George 7
Green-Lagrange-Verzerrungstensor 7 ff.
Grenzschicht 24, 32
Grob-Gitter-Gradient 133
Grob-Gitter-Korrektur 127
Grob-Gitter-Prädiktor **127**, 131, 133, 142 ff.

H

Hardware 36, 142
Hardware-Architektur 53, 61, 84, **97**, 113
Hardware-Ressourcen 40, 99
HLRS 53, 86

Hochleistungsrechnen **35**, 36 ff., 147
Hochleistungsrechner 35–38, 40 f., 53, 86, 149, 160

I

ILU 82
Implementierung 36, 42, 53, 56, 60, 62, 81, 83 f., 103, 108, 137, 139, 143, 147
Impulsbilanz **8**, **18**, 20 f., 23, 25, 27, 30
Impulserhalt 103, 105, 111, 114, 116
Inkompressibilitätsbedingung **19**, 21
Instabilität 25, 114 ff., 141 f.
Integrationspunkt 44, 57
Interface 105, 111, 116
Interface-Kompatibilitäts-Operator 107, 135 f.
Interface-Verschiebungen 106, 119, 123 f.
Iterationsmatrix **69**, 72 f.
Iterationsverfahren **67**, 83 f., 112
  instationäres 69, 119
  Klassifizierung 68
  Konvergenz 70
  lineares 69, 117
  nichtlineares 69 f., 107, 119
  Ordnung 69
  stationäres 69
Iterationsvorschrift 69, 107

J

Jacobi, Carl Gustav Jacob 58, 67
Jacobi-Matrix 58, 134 ff., 138 ff.
Jacobi-Verfahren 68, **70**, 73, 83, 121
Jacobi-Vorkonditionierung **82**, 84, 86, 89 f., 94

K

Kinematik 6, 18
Kirchhoff, Gustav Robert 8
Kompatibilität 105, 107, 136
Konditionierung 87, 90, 95, 154
Konditionszahl 77, 80
Konjugierte-Gradienten-Verfahren 74, 76 f.
Konstitutivgesetz 8, 18
Kontinuität 48 f., 103
  C^1 47
  dynamische 103, 105, **106**, 114
  kinematische 103, **105**, 114 ff., 143
Kontinuitätsgleichung **19**, 21, 25 f., 29 f.
Kontinuumsmechanik 5 f., 18
Kontrollpunkt 47, 50
Kontrollpunkt-Polygon 47
Konvektion 24, 27, 136
Konvergenz 70, 103, 140 f.
  lineare 70
  quadratische 31, 108, 136
Konvergenzaussage 79 f.
Konvergenzbeschleunigung 122, 124
Konvergenzeigenschaft 116, 139
Konvergenzgeschwindigkeit 72, 80, 84, 88, 95, 142
Konvergenzverhalten 78, 80, 96
Kopplungsalgorithmus 16, 31, 33
Kopplungsbedingung 22, 102, 104, **105**, 106, 109, 116
Kopplungskraft 16, 32, 104, 106, 113
Kopplungsrand 16, 31 ff., 104 f., 112, 116, 118
Kopplungsverfahren 103, 127
  asynchrones 111, 114
  einfach gestaffeltes 109 f.
  explizites 109
  implizites 109
  iterativ gestaffeltes 109, **116**
  parallel gestaffeltes **111**, 122
  schwach koppelndes 109
  semi-implizites 141
  sequenziell gestaffeltes **113**, 120, 142

stark koppelndes 109
Korrektormatrix **70**, 71, 81
Krylow, Alexei Nikolajewitsch 74
Krylow-Unterraum 74, 76, 79, 85, **91**
Krylow-Unterraum-Verfahren **73**, 78 ff., 84, 86, 120, 136

L

Löser
direkter 68, 79
iterativer **67**
linearer 45
Lösungsverfahren
einfach gestaffeltes 110
iterativ gestaffeltes 116
monolithisches 103, **108**, 116
partitioniertes 103, 107, **108**
Ladyschenskaja, Olga Alexandrowna 24
Ladyschenskaja-Babuška-Brezzi-Bedingung 24
Lagrange, Joseph Louis 6
Lagrange-Formulierung 6, 106
Lamé, Gabriel 9
Lamé-Konstanten 9
Lanczos, Cornelius 78
Lattice-Boltzmann 121
LBB-Bedingung 24
„loop“
„division“ 56
„expansion“ 56
„fusion“ 56
„interchange“ 56 f.
„unrolling“ 56, 86
LU-Zerlegung 82

M

Mach, Ernst 19
Mach-Zahl 19
„mapping“-Methode 41 f.
Massenbilanz **18**, 20, 23
Massenerhalt 103, 105, 111, 114, 116
Massenmatrix 12, 27, 29
Materialgebiet 6
Materialgesetz **8**, 57, 108
Materialparameter 18, 148, 153, 158, 160
Materialtensor 9
Matrix-Vektor-Produkt 78 f., 81 ff., 86, 136, 138, 146
Mehrfeldproblem 102
Mehrgitter-Verfahren 42, 126, 129
Methode der gewichteten Residuen 11
Modellreduktion 103
Momentankonfiguration 6
„momentum balance equation“ 25

N

Navier, Claude Louis Marie Henri 20
Navier-Stokes-Gleichungen **21**, 25 f., 29, 74, 84, 115
NEC SX-8 **38**, 54, 86, 97 f., 147, 151, 155
Netz-Operator **33**, 104
Netzbewegung 32
Netzdynamik 32
Netzerzeugung **40**, 41
Netzgeschwindigkeit 20, 32
Netzqualität 32
Netzverfeinerung 24, 42 f.
Netzverschiebung **32**, 104, 136, 153, 159
Neumann, Carl Gottfried 10
Newmark, Nathan M. 13
Newmark-Parameter 13 f.
Newton, Isaac 15
Newton-Krylow-Verfahren 109, 117, 120, 127, 134, **135**, 138, 141 f., 146
Jacobi-freies 140

Newton-Raphson-Verfahren **15**, 31, 108, 112, 135 f.
Norm 15, 119, 135
  euklidische 79, 119
Normalform 69
NURBS **46**, 49, 51

O

Optimalitätseigenschaft 76, 125, 134
Orthogonalitätsbedingung 74, 78
Ortsvektor 6, 17
Oszillation 20, 24, 29, 78, 105

P

Parallelisierung 37, 41, 71, 103, 111 f., 122
„performance“
  „peak“ 37
  „sustained“ **37**, 65, 86
Petrow, Georgi Iwanowitsch 25
Petrow-Galerkin-Bedingung 74, 78 f.
Piola, Gabrio 8
Piola-Kirchhoff-Spannungstensor
  Erster 8
  Zweiter **8**, 9 f., 106
Pipeline 39, 54, 56
Poisson, Siméon-Denis 9
Poisson-Gleichung 115, 139
Poissonzahl 9
Polynomgrad 42, 49
Postprozessor 40 f., 86
Prädiktor 112 f., **114**, 115, 118–121, 127, 129, 131, 135, 143, 154, 161
Präprozessor 40 ff., 46, 86, 149, 159
Pressure Stabilized Petrow-Galerkin 25
Prinzip der virtuellen Verschiebungen 11, 13
Produkt, inneres 16, **23**
Projektions-Verfahren 141
Projektionsmethode 73
  orthogonale 74 f.
  schiefe 74, 79
Prolongation 126, 129, **129**, 130, 133
Pseudo-Struktur-Ansatz **32**, 153, 159

Q

Quasi-Newton-Verfahren 117
Querdehnzahl 9

R

Rückwärtseinsetzen 33, 81 f.
Randbedingung **10**, 16, **20**, 32, 36, 41, 84, 105, 120, 125, 133
  Ausfluss 22
  Dirichlet 10, 21, 33, 105, 141
  Druck 22
  Einström 150, 161
  geometrische 10 f.
  Gleit 21, 105, 150, 161
  Haft 21, 85, 105, 161
  natürliche 10
  Neumann 10, 21, 112 f., 141
  Robin 105
  statische 10 f.
  wesentliche 10
Randschicht 17, 147
Randwertproblem 8 ff., 12, 18, 20, 22, 24, **32**, 73
Raphson, Joseph 15
Raumgebiet 6, 22 f.
Rechenaufwand 41, 68, 78 f., 87, 103, 123 f.
Rechenkapazität 36
Rechenleistung 35
Rechenzeit 37, 40, 56, 60, 63, 67 f., 84, 86 f., 127, 132, 134, 137, 141 f.

Rechnerarchitektur 35, 37
Referenzgebiet 17, 19
Referenzkonfiguration 6–9
Rekursionsformel 48
Relaxation 73, 116, 118, 120, **122**, 129, 131, 141
Relaxations-Parameter 72 f., 85, 87, 89, 94, 118, 123 ff., 133, 143, 154, 161
Relaxations-Verfahren 72
Residuum 15, 22, 25, 68, 70, 77 ff., 116, 119, 126, 129, 135, 139 f.
Restriktion 128, **129**, 133
Reynolds, Osborne 17
Reynoldszahl 17, 86, 95 f., 150, 160
Richardson, Lewis Fry 109
Richardson-Iteration 109, 117
Robin, Victor Gustave 105
Robin-Kopplungs-Bedingung 141
Robin-Partitionierung 141

S

Saint-Venant, Adhémar Jean Claude de 9
Saint-Venant-Kirchhoff-Material 9
Schachbrettmoden 24
Schubspannungen, viskose 18
Schur, Issai 120
Schurkomplement-Matrizen 125
Schurkomplement-Verfahren 120
Schwache Form 11, 22
  semidiskrete 25, 27
Segmentierung 39
Serendipity-Element 45
Shift-Parameter 14
Sobolew, Sergei Lwowitsch 23
Sobolew-Raum 23
SOR **73**, 118 f.
Spannungs-Dehnungs-Beziehung 9
Spannungsmaß 8
Spannungszustand, hydrostatischer 18
Sparse-Matrix-Format 45
Speicherbedarf 40, 57, 79, 82, 91 ff., 120
Spektralradius 14, 70–73
Stabilisierung 14, 16, **25**, 27, 97, 136
  „augmented mass“ 116
  GLS **26**, 97
  SUPG/PSPG **26**, 86, 96, 149
Stabilisierungs-Operator 25
Stabilisierungs-Parameter 26, **27**, 31, 95, 97
Stabilisierungs-Verfahren 25
Starke Form 10, 20, 22
Steifigkeitsmatrix 15, 33, 45, 56
Stokes, George Gabriel 20
Stokes-Matrix 84
Stokes-Problem 25 f.
Strömung 17
  inkompressible 5, 18, 20 f.
  instationäre 22
  kompressible 84
  konvektionsdominante 24
  laminare 85
  stationäre 85
  viskose 5, 18, 20, 105
Strömungseffekte 147
Strömungseigenschaft 27
Strömungsgleichungen 17
Strömungsmechanik 36
Strömungssimulation 32, 35 ff., 40
Streamline Upwind Petrow-Galerkin 25
„stride one“ 54, 56, 59
Struktur **6**
Struktur-Operator **16**, 104, 118, 136
Struktur-Prädiktor 111, **114**
Strukturdynamik 5, **6**, 7, 10, 102
Strukturmechanik 120
Strukturmodell 6
Substruktur-Methoden, iterative 120
SUPG/PSPG-Verfahren 25, **26**

T

Taylor, Brook 15
Taylorreihe 15, 135
Tera-FLOPS 86
Testfunktion 11 f., 22 ff.
Tonti, Enzo 10
Tonti-Diagramm 10, 22
Trapezregel 29, 105

U

Unusual Stabilized Finite Element Method 25 f.
USFEM 25 f.

V

Vektor-Pipeline 39
Vektorcomputer 38, 147, 151
Vektorisierung **53**, 55 f., 98, 103
Vektorlänge 54, 65, 86, 98
Vektoroperationen 54, 65
Vektorprozessor 37, **38**, 58, 61, 65, 94, 98
Vektorrechner 35, 38, 53, 86, 97
Vektorregister 53 f., 63
Verfahren des steilsten Abstiegs 75
Vernetzung 40
Verschiebungsfeld 6, 10
Verschiebungsformulierung 11 f.
Verzerrungen 6, 9, 11
Verzerrungs-Verschiebungs-Beziehung 7
Verzerrungsarbeit 9
Verzerrungsmaß 7
virtuelle Verschiebung 11, 13
Viskosität 17, 106
  dynamische 18
  kinematische 18, 86
Vorkonditionierer 67 f., 72, 77 f., **80**, 84 f., 87, 93, 96, 108, 140
  algebraischer 81
  Block 83
  expliziter 81
  funktionaler 81
  impliziter 81
  linker, rechter 80
  splitting-assoziierter 81
Vorkonditionierung **80**, 82, 84
Vorwärtseinsetzen 81 f.

Z

Zeigervariable 54 f., 60
Zeitableitung 8
  materielle 19 f.
Zeitdiskretisierung 13, 29
Zeitintegrationsalgorithmus 14
Zeitintegrationsansatz 13, 15
Zeitintegrationsverfahren 13 f., 20, 29 f., 111, 115
Zeitschritt **13**, 15, 29, 31 f., 36, 40 f., 63, 85 f., 95 f., 104, 111, 115, 149, 151, 153 f., 159 f.
Zerlegung
  additive 70
  der Eins 47
  multiplikative 82
Zwei-Level-Verfahren 110, **126**, 127, 134

Lebenslauf

Persönliche Angaben:

Name:	Malte von Scheven, geb. Neumann
Geburtsdatum:	30. Juli 1976
Geburtsort:	Bochum
Familienstand:	verheiratet
Konfession:	evangelisch

Schulbildung:

08/1983 - 07/1987	Gemeinschaftsgrundschule Hufelandstraße, Bochum Querenburg
08/1987 - 06/1996	Gymnasium Schiller-Schule, Bochum
01/1993 - 12/1993	Auslandsaufenthalt an der Melville Highschool, Hamilton / Neuseeland

Schulabschlüsse:

12/1993	Neuseeländische Hochschulreife (New Zealand A-Bursary)
06/1996	Deutsche Allgemeine Hochschulreife

Zivildienst:

10/1996 - 10/1997	Freizeit- und Jugendbildungsstätte Oeverdiek der Evangelisch-methodistischen Kirche in Timmendorfer Strand

Studium:

10/1997 - 03/2002	Studium des Bauingenieurwesens an der Ruhr-Universität Bochum
03/2002	Diplomprüfung (mit Auszeichnung)
11/2002	Auszeichnung der Diplomarbeit als herausragende wissenschaftliche Arbeit mit dem Preis an Studierende 2002 der Ruhr-Universität Bochum

Beruflicher Werdegang:

04/2002 - 07/2006	wissenschaftlicher Assistent am Institut für Baustatik der Universität Stuttgart
seit 07/2006	Akademischer Rat am Institut für Baustatik und Baudynamik der Universität Stuttgart

Berichte des Instituts für Baustatik und Baudynamik der Universität Stuttgart

1 (1983) P. Osterrieder:
Traglastberechnung von räumlichen Stabwerken bei großen Verformungen mit finiten Elementen.

2 (1983) T.A. Kompfner:
Ein finites Elementmodell für die geometrisch und physikalisch nichtlineare Berechnung von Stahlbetonschalen.

3 (1983) A. Diack:
Beitrag zur Stabilität längsversteifter Kreiszylinderschalen unter Axialdruck.

4 (1984) A. Burmeister, F.W. Bornscheuer, E. Ramm:
Traglasten von Kugelbehältern mit Stutzen und Formabweichungen unter Innendruck und Stützenlängskraft.

5 (1985) H. Stegmüller:
Grenzlastberechnungen flüssigkeitsgefüllter Schalen mit „degenerierten" Schalenelementen.

6 (1987) A. Burmeister:
Dynamische Stabilität nach der Methode der finiten Elemente mit Anwendung auf Kugelschalen.

7 (1987) G. Kammler:
Ein finites Elementmodell zur Berechnung von Trägern und Stützen mit offenem, dünnwandigem Querschnitt unter Berücksichtigung der Interaktion zwischen globalem und lokalem Versagen.

8 (1988) A. Matzenmiller:
Ein rationales Lösungskonzept für geometrisch und physikalisch nichtlineare Strukturberechnungen.

9 (1989) D. Tao:
Die Technik der reduzierten Basis bei nichtlinearen finiten Element-Berechnungen.

10 (1989) K. Weimar:
Ein nichtlineares Balkenelement mit Anwendung als Längssteifen axialbelasteter Kreiszylinder.

11 (1990) K.-U. Bletzinger:
Formoptimierung von Flächentragwerken.

12 (1990) S. Kimmich:
Strukturoptimierung und Sensibilitätsanalyse mit finiten Elementen.

13 (1991) U. Andelfinger:
Untersuchungen zur Zuverlässigkeit hybrid-gemischter finiter Elemente für Flächentragwerke.

14 (1992) N. Büchter:
Zusammenführung von Degenerationskonzept und Schalentheorie bei endlichen Rotationen.

15 (1992) T.J. Hofmann:
Beitrag zur verfeinerten Balkentheorie.

16 (1994) D. Roehl:
Zur Berechnung von großen elastoplastischen Deformationen bei Flächentragwerken und Kontinua.

17 (1994) R. Reitinger:
Stabilität und Optimierung imperfektionsempfindlicher Tragwerke.

18 (1995) R. Suanno:
Ein dreidimensionales Simulationsmodell für Stahlbeton mit Plastizität und Schädigung.

19 (1995) M. Braun:
Nichtlineare Analysen von geschichteten, elastischen Flächentragwerken.

20 (1996) N. Rehle:
Adaptive Finite Element Verfahren bei der Analyse von Flächentragwerken.

21 (1996) C. Haußer:
Effiziente Dreieckselemente für Flächentragwerke.

22 (1996) D. Kuhl:
Stabile Zeitintegrationsalgorithmen in der nichtlinearen Elastodynamik dünnwandiger Tragwerke.

23 (1998) H. Schmidts:
Zur effizienten Modellierung und Analyse von Hochhaustragwerken.

24 (1998) H. Wang:
Interaktion des lokalen und globalen Stabilitätsverhaltens dünnwandiger Stäbe.

25 (1998) K. Maute:
Topologie- und Formoptimierung von dünnwandigen Flächentragwerken.

26 (1998) B. Maurer:
Karl Culmann und die graphische Statik.

27 (1998) F. Cirak:
Adaptive Finite-Element-Methoden bei der nichtlinearen Analyse von Flächentragwerken.

28 (1998) M. Trautz:
Zur Entwicklung von Form und Struktur historischer Gewölbe aus der Sicht der Statik.

29 (1999) H. Menrath:
Numerische Simulation des nichtlinearen Tragverhaltens von Stahlverbundträgern.

30 (1999) M. Bischoff:
Theorie und Numerik einer dreidimensionalen Schalenformulierung.

31 (1999) W.A. Wall:
Fluid-Struktur-Interaktion mit stabilisierten Finiten Elementen.

32 (2000) E. Kuhl:
Numerische Modelle für kohäsive Reibungsmaterialien.

33 (2001) A. Maute:
Adaptive Finite-Element-Methoden in der Strukturdynamik.

34 (2001) S. Schwarz:
Sensitivitätsanalyse und Optimierung bei nichtlinearem Strukturverhalten.

35 (2001) A. Haufe:
Dreidimensionale Simulation bewehrter Flächentragwerke aus Beton mit der Plastizitätstheorie.

36 (2001) D.P. Mok:
Partitionierte Lösungsverfahren in der Strukturdynamik und der Fluid-Struktur-Interaktion.

37 (2002) H. Steeb:
Fehlerschätzer für FE-Berechnungen bei entfestigenden Materialien.

38 (2002) K. Krausz:
Tragverhalten gemauerter Tonnengewölbe mit Stichkappen.

39 (2002) M. Hörmann:
Nichtlineare Versagensanalyse von Faserverbundstrukturen.

40 (2003) V. Gravemeier:
The Variational Multiscale Method for Laminar and Turbulent Incompressible Flow.

41 (2004) R. Kemmler:
Stabilität und große Verschiebungen in der Topologie- und Formoptimierung.

42 (2004) G.A. D'Addetta:
Discrete Models for Cohesive Frictional Materials.

43 (2004) M. Gee:
Effiziente Lösungsstrategien in der nichtlinearen Schalenmechanik.

44 (2004) T. Erhart:
Strategien zur numerischen Modellierung transienter Impaktvorgänge bei nichtlinearem Materialverhalten.

45 (2005) M. Leukart:
Kombinierte anisotrope Schädigung und Plastizität bei kohäsiven Reibungsmaterialien.

46 (2006) F. Huber:
Nichtlineare dreidimensionale Modellierung von Beton- und Stahlbetontragwerken.

47 (2007) A. Lipka:
Verbesserter Materialeinsatz innovativer Werkstoffe durch die Topologieoptimierung.

48 (2007) A.S. Hund:
Hierarchische Mehrskalenmodellierung des Versagens von Werkstoffen mit Mikrostruktur.

49 (2007) S. Hartmann:
Kontaktanalyse dünnwandiger Strukturen bei großen Deformationen.

50 (2007) T.M. Hettich:
Diskontinuierliche Modellierung zur Versagensanalyse von Verbundmaterialien.

51 (2007) Ch. Förster:
Robust methods for fluid-structure interaction with stabilised finite elements.

Universität Stuttgart

ISBN 978-3-00-027330-8