Lehrstuhl für Bauinformatik
Fakultät für Bauingenieur- und Vermessungswesen
Technische Universität München

Lattice-Boltzmann Strömungssimulationen auf Baumdatenstrukturen

Bernd Crouse

Vollständiger Abdruck der von der Fakultät für Bauingenieur- und Vermessungswesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr.-Ing. Th. Strobl

Prüfer der Dissertation:

1. Univ.-Prof. Dr.rer.nat. E. Rank
2. Univ.-Prof. Dr.-Ing. habil. M. Krafczyk
   Technische Universität Braunschweig

Die Dissertation wurde am 16.06.2003 bei der Technischen Universität München eingereicht und durch die Fakultät für Bauingenieur- und Vermessungswesen am 28.11.2003 angenommen.

Vorwort

Die vorliegende Arbeit entstand im Rahmen eines von der Firma SOFiSTiK AG gewährten Industriestipendiums während meiner Tätigkeit am Lehrstuhl für Bauinformatik an der Technischen Universität München (Dezember 1999 - März 2003).

An erster Stelle möchte ich mich bei meiner Mutter Cäcilia Crouse bedanken. Sie war mir stets eine wichtige Stütze auf meinem Weg.

Mein besonderer Dank gilt Herrn Professor Dr.rer.nat. E. Rank für die sehr gute Betreuung und großzügige Förderung dieser Arbeit. Seine Vorschläge und Anregungen haben zum Gelingen dieser Arbeit wesentlich beigetragen.

Ebenso möchte ich mich bei Herrn Professor Dr.-Ing. habil. M. Krafczyk bedanken. Die vielen Diskussionen und Ratschläge haben die Richtung dieser Arbeit maßgeblich beeinflusst und ebenfalls zum Gelingen dieser Arbeit beigetragen.

Für die stetige, fachliche Unterstützung möchte ich Herrn Dr.-Ing. Jonas Tölke meinen freundschaftlichen Dank aussprechen.

Allen Kollegen, die mich während meiner Zeit am Lehrstuhl begleitet haben, möchte ich meinen Dank gleichermaßen aussprechen. Das angenehme Arbeitsklima und der damit verbundene freundschaftliche Umgang haben sehr zum Gelingen dieser Arbeit beigetragen.

Bei meinem Kollegen Herrn Dipl.-Ing. Siegfried Kühner, mit dem ich das Büro geteilt habe, möchte ich mich für die lehrreiche Zeit in freundschaftlicher und humorvoller Atmosphäre bedanken.

Für die Leitung der Prüfungskommission und sein Interesse an dieser Arbeit möchte ich mich bei Herrn Professor Dr.-Ing. Th. Strobl herzlich bedanken.

Bei der SOFiSTiK AG bedanke ich mich für das mir durch das Promotionsstipendium entgegen gebrachte Vertrauen, ohne das diese Arbeit nicht zustande gekommen wäre. Insbesondere möchte ich mich hier bei den Herren Dr. C. Katz und Dr. J. Bellmann recht herzlich bedanken.

Zusammenfassung

In den Ingenieurwissenschaften, so auch im Bauwesen, gewinnen Fragestellungen der Strömungsmechanik und deren Lösung mithilfe von Strömungssimulationen zunehmend an Bedeutung. Eine wesentliche Aufgabe besteht darin, Methoden und Ansätze zu deren Anwendung zu erforschen und zu validieren.

Im ersten Teil dieser Arbeit wird ein Integrationskonzept vorgestellt, mit dem der Gesamtablauf von Strömungssimulationen effizienter gestaltet werden kann. Es wird dazu ein Rahmenwerk entwickelt, welches das Beziehungsgeflecht zwischen den geometrischen Modellen der Teilaufgaben von Strömungssimulationen darlegt und den Ablauf der Simulationen prinzipiell vorgibt. Als zentrale Datenstruktur wird ein Oktalbaum, ein dreidimensionaler Vertreter der Baumdatenstrukturen, verwendet. Dieser unterstützt einerseits das Abbilden der geometrischen Modelle und ermöglicht andererseits eine automatische Generierung kartesischer Gitter. Die angestrebte Kopplung von CAD und Strömungssimulation wird an zwei ausgewählten Beispielen demonstriert.

Der zweite Teil der Arbeit beschäftigt sich mit der Erweiterung der Lattice-Boltzmann Methode für Simulationen auf nicht-uniformen Gittern für zweidimensionale Probleme. Hierfür werden zunächst die Grundlagen der Methode dargestellt und die entsprechenden Erweiterungen für die Nichtuniformität im Detail erläutert. Von essentieller Bedeutung sind dabei die das Gitter repräsentierenden Datenstrukturen. Auch hier finden Baumdatenstrukturen, für zweidimensionale Anwendungen als Quadtrees bezeichnet, optimale Einsatzmöglichkeiten. Die Besonderheiten, die sich in der prototypischen Implementierung ergeben, werden ausführlich dargestellt. Da der Prototyp die Modifikation der Gitterkonfiguration erlaubt, werden erstmalig adaptive Lattice-Boltzmann Simulationen durchgeführt und deren Effizienz gezeigt.

Abstract

Fluid mechanics and its applications using methods of computational fluid dynamics (CFD) gain an increasing interest in the field of engineering technologies. One of the main issues of sciences in this area is the exploration of methods and solutions dealing with CFD.

In the first part of this work an integration concept, with the purpose of optimization of the application flow of fluid-flow simulations, is presented. Therefore, a framework, identifying the relationsships between the different geometric models and defining the sequence of the different tasks in flow simulations, is proposed. The central datastructure is based on an octree, a three-dimensional representative of the spacetrees. It is used to support the mapping of the different geometric models onto each other and to ensure the automation of the cartesian grid generation. The aspired coupling of CAD and flow simulation is

demonstrated using specific examples.

The second part of this work deals with the extension of the Lattice-Boltzmann method for simulations on non-uniform two-dimensional grids. Therefore, a brief introduction of the basics of the method is given and the corresponding enhancements for the non-uniform grids are explained in detail. The datastructure, representing the grid, is of essential interest. The two-dimensional version of the spacetrees, the quadtrees, find their optimal employment for this purpose. The main aspects, concerning the prototypic implementation, are described. As the prototyp allows a modification of the grid configuration during runtime, adaptive Lattice-Boltzmann simulations can be performed for the first time demonstrating their efficiency.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Strömungssimulationen im Ingenieurwesen

Die Erfassung der Strömungsverhältnisse in und um Bauwerke stellt heutzutage nach wie vor eine große Herausforderung für Ingenieure dar. Bei der Gebäudeumströmung erzeugt Wind sowohl statische als auch dynamische Belastungen an Bauwerken. Sie spielen insbesondere bei sensiblen Gebäuden, die dem Wind eine große Angriffsfläche bieten (z. B. Brücken, Türme und Hochhäuser), eine große Rolle. Standard-Tabellenwerke [91] gehen oft von quasi-stationären Strömungsverhältnissen aus oder beziehen sich auf vordefinierte Tragwerksformen. Dies kann zum einen zu suboptimalen Ergebnissen infolge übermäßig hoher Sicherheitsfaktoren führen und zum anderen Versagensformen wie Resonanzkatastrophen zur Folge haben, wenn aufgrund von bidirektionalen Fluid-Struktur-Wechselwirkungen die ihr innewohnende nichtlineare Dynamik unterschätzt wurde [18, 39]. Eine derartige Katastrophe ereignete sich beispielsweise 1965 in Ferrybridge, England, als drei von acht Kühltürmen infolge von periodisch ablösenden Wirbelballen einstürzten [85].

Die derzeit weitestverbreitete Möglichkeit, die Lastannahmen im Bauwesen festzulegen, stellt der Versuch im Windkanal dar. Mithilfe von maßstabgetreu gebauten Modellen wird versucht, reale Zustände zu simulieren. Aber zu welchen Kosten? Die Konstruktion und vor allem die Änderung der Modelle ist teuer und langwierig. Die Messung der Strömungsverhältnisse im Windkanalversuch kann nur an Orten geschehen, an denen Sonden angebracht sind, die ebenfalls teuer sind und meist das Strömungsfeld beeinflussen. Auf der anderen Seite ist man mit Windkanalversuchen zumindest in der Lage, unter Berücksichtigung der Ähnlichkeitsanforderungen [39, 79] Ergebnisse mit hoher Aussagekraft zu erlangen.

Computersimulationen zur Vorhersage von Strömungen in und um Bauwerke haben erst in den letzten Jahren Anwendung im Bauingenieurwesen erfahren. Die wichtigsten Gründe hierfür sind in den drei wesentlichen Aufgabenbereichen, in

die der Gesamtablauf von strömungsmechanischen Simulationen typischerweise unterteilt wird, zu finden [18, 19] (vgl. Abb. 2.2):

- Vorverarbeitungsphase (Preprocessing):
  Das Abbilden von CAD-Daten auf Berechnungsgitter- oder Berechnungsnetze ist gewöhnlich sehr aufwendig und zeitraubend.

- Berechnungsphase (Solving):
  Die Simulation von transienten, turbulenten Strömungen ist sehr schwierig und rechenzeitintensiv.

- Nachbereitungsphase (Postprocessing):
  Eine effiziente Visualisierung und Auswertung der dreidimensionalen Ergebnisdaten ist meist von der CAD Umgebung entkoppelt und nur selten in den Entwicklungszyklus integriert.

Durch die Entwicklung von leistungsfähigen und preiswerten Computern sowie der gleichzeitigen Umsetzung verbesserter numerischer Algorithmen in effiziente Software-Produkte hat sich die Anwendung von Simulationsverfahren zur näherungsweisen Lösung der Navier-Stokes Gleichungen in den letzten Jahren in vielen Ingenieurdisziplinen als echte Alternative, teilweise sogar als einziger praktikabler Lösungsansatz entwickelt. Zunächst sei an dieser Stelle die kommerzielle Strömunggsimulations-Software *CFX*[1] angeführt, welche sich vor allem im Bereich des Maschinenbaus in Einsatzgebieten wie der Turbinenströmung, aber auch im nahezu gesamten Spektrum der Strömungsphänome etabliert hat [68, 70, 95]. Ein noch sehr junges Software-Packet zur Simulation von Strömungen ist das Programm *PowerFLOW* der *Exa Corporation*[2]. Es unterscheidet sich von den anderen kommerziellen Programmen im numerischen Kern, der auf der sogenannten *Lattice-Boltzmann Methode* basiert. Durch Vorteile, die im Verlauf dieser Arbeit deutlich werden, hat dieses Produkt Einzug bei nahezu allen deutschen Automobil-Herstellern und in vielen weiteren Industriezweigen gefunden [67, 73, 75, 94]. Viele Firmen haben bereits die Anzahl der Windkanalversuche deutlich reduziert und planen sogar für einige Teilprobleme komplett auf diese zu Gunsten von Strömungssimulationen zu verzichten [13]. Dies zeigt, dass computergestützte Simulationen in vielen Bereichen sehr wohl als Alternative anzusehen sind, wenngleich man sicher noch lange auf Windkanalversuche nicht ganz verzichten wird.

1.2 Ziele der Arbeit

In der Baubranche sind typischerweise eine Vielzahl von interdisziplinären Projektpartnern bei der Planung und Konstruktion von Bauwerken beteiligt. Während

[1] http://www.cfx-germany.com
[2] http://www.exa.com

der Statiker die Annahme von Lasten sowie die Stabilität von Bauwerken sicherstellen muss, der Architekt sich um das Design und die Funktionalität kümmert, hat beispielsweise der Klima-Ingenieur die Aufgabe, sich um ausreichende Belüftung zu sorgen. Gleichzeitig sind alle Aufgabenbereiche von einander abhängig, sodass es nur natürlich erscheint, dass vor allem in frühen Planungsstadien immer wieder Fehlentscheidungen getroffen werden, weil eine Vorhersage beispielsweise von Belastungen durch Wind nur unzureichend möglich war. Es wird geschätzt, dass ca. 20% der gesamten Herstellungskosten eines Bauwerkes durch nachträgliche Korrekturen bedingt durch diese Fehlentscheidungen entstehen [81].

Ziel dieser Arbeit ist es, den Gesamtablauf einer Strömungssimulation unter Verwendung eines auf der Lattice-Boltzmann Methode basierten Berechnungskerns zu ermöglichen und zu optimieren, um somit den Planungsprozess zu unterstützen. Es wird ein möglichst hohes Maß an Automatisierung angestrebt, nicht nur um den Design-Zyklus zu verkürzen, sondern auch um die Methoden der Strömungssimulation in ein für Ingenieure vertrautes Umfeld zu integrieren. Hierfür ist eine detaillierte Analyse der Teilprozesse des Gesamtablaufs notwendig, um die erforderliche Transparenz der Problematik zu erhalten. Die Überlegungen und Implementierungen dieser Arbeit konzentrieren sich unter Anwendung objektorientierter Integrationstechniken des Software-Engineerings vor allem auf die Bereiche Preprocessing und Berechnungsphase, während der Bereich Postprocessing in [51] abgehandelt wird.

Weiterer wesentlicher Gegenstand dieser Arbeit ist die Erweiterung der Lattice-Boltzmann Methode für zweidimensionale Strömungen auf nicht-uniformen Gittern. Da sich weltweit bislang die Arbeiten in diesem Bereich hauptsächlich auf die Analyse und die Optimierung des Modells konzentriert haben, wurden in erster Linie uniforme und blockstrukturierte Raum-Zeit Gitter verwendet. Allerdings erfordert die Mehrzahl von Strömungsphänomenen aus dem Ingenieurwesen die Verwendung von nicht-uniformen Gittern, um lokale Effekte numerisch besser auflösen zu können. Dies wird durch die Wahl von Baumdatenstrukturen als Basis für das Berechnungsgitter begünstigt.

1.3 Gliederung der Arbeit

Die vorliegende Arbeit befasst sich mit zwei Themenschwerpunkten, der Integration und Kopplung der Teilprozesse in Strömungssimulation und den multiskalen Lattice-Boltzmann Simulationen auf Baumdatenstrukturen.

Im **ersten Teil**, bestehend aus den Kapitel 2 bis 4, werden diese Teilprozesse weitgehend automatisiert und somit Problemstellungen der Ingenieurpraxis wie zum Beispiel Innenraumströmungssimulationen zugänglich gemacht. Der **zweite Teil** beinhaltet die Kapitel 5 bis 8 und beschäftigt sich mit den theoretischen Grundlagen der Lattice-Boltzmann Methode und deren Erweiterungen bezüglich

unstrukturierter und selbstorganisierender Gitter.

Im **zweiten Kapitel** wird nach einem ersten Überblick über Techniken des Software-Engineerings speziell auf den mit numerischen Strömungssimulationen unterstützten Produktentwicklungszyklus eingegangen. Weiter werden die kontextbezogenen Integrationsziele und im Anschluss daran einige Implementierungsanforderungen dargestellt.

Eine Übersicht über geometrische Darstellungsformen vermittelt das **dritte Kapitel**. Die Erläuterungen beschränken sich auf jene Geometrieformen, die im Gesamtablauf von Strömungssimulationen verwendet werden. Aufgrund der zentralen Bedeutung der raumpartitionierenden Baumdatenstrukturen werden diese im Gegensatz zu den anderen Geometrieformen detailliert vorgestellt.

Die Überlegungen und Ausführungen für ein Integrationskonzept, welches den Produktentwicklungsprozess von Bauwerken oder Bauteilen in der Planung unterstützen soll, werden in **Kapitel 4** ausführlich dargelegt. Mithilfe einer Ablaufkette werden die Abhängigkeiten zwischen den geometrischen Modellen verdeutlicht. Hierbei nehmen die Baumdatenstrukturen ihre zentrale Rolle im gesamten Rahmenwerk ein. Daher wird auf einige wichtige Algorithmen, welche zur Generierung der Baumdatenstrukturen angewandt werden, im Detail eingegangen. Zum Abschluss des Kapitels und des ersten Teils dieser Arbeit wird der Nutzen des vorgestellten Ansatzes anhand einer Innenraumströmung eines Bürozimmers mit komplexer Geometrie demonstriert.

Als Basis für den zweiten Teil dieser Arbeit werden in **Kapitel 5** die Grundlagen der Lattice-Boltzmann Methode erklärt und der Zusammenhang zu den Navier-Stokes Gleichungen verdeutlicht. Unter Verwendung von anschaulichen Erklärungen soll ein Einstieg in dieses numerische Verfahren ermöglicht werden.

Die methodischen Erweiterungen des Lattice-Boltzmann Verfahrens für Multiskalen Strömungssimulationen werden in **Kapitel 6** vorgestellt. Im Gegensatz zu vielen anderen numerischen Verfahren ändert sich bei Variation der Auflösung des Gitters nicht nur der Rechenaufwand und die Qualität der Lösung, sondern auch das numerische Grundsystem, der physikalische Phasenraum.

Im **Kapitel 7** wird das Konzept ebenso wie einige Implementierungsdetails eines prototypischen Lattice-Boltzmann Strömungslösers erläutert. Besonders hervorgehoben werden die Baumdatenstrukturen, da sie die grundlegende Datenstruktur des Berechnungsgitters sind. Wegen der Nähe zur Implementierung werden im gleichen Zuge sowohl die Randbedingungen als auch die Evaluierungsmethoden, z. B. zur Berechnung eines Widerstandbeiwerts, beschrieben. Das Kapitel wird mit der Verifizierung des Prototypen und der entsprechenden Grundlagen mithilfe einiger 2D-Benchmark-Strömungssimulationen abgeschlossen.

Gegenstand der Untersuchung des **achten Kapitels** ist die Erweiterung des Prototypen für adaptive Strömungsberechnungen. Hierzu wird das typische Vorgehen skizziert und die essentiellen Voraussetzungen, wie die Definition von heu-

ristischen Kriterien zur Verfeinerung aufgeführt. Die Bedeutung, die Effizienz und die Notwendigkeit von adaptiven Berechnungen im Ingenieurwesen wird an ausgewählten Beispielen gezeigt.

Abschließend erfolgt in **Kapitel 9** eine Zusammenfassung der wesentlichen Aussagen dieser Arbeit und ein Ausblick auf zukünftige Entwicklungen.

Teil I

Integrationskonzept für computergestützte, gitterbasierte Strömungssimulationen

Kapitel 2

Software-Engineering im CFD-basierten Planungsprozess

Moderne Software-Produkte spielen in nahezu allen wirtschaftlichen Zweigen eine wichtige Rolle. Im Ingenieurwesen steht neben der Unterstützung von Prozessabläufen (vernetzt-kooperative Planung) die Vorhersage von physikalischem Verhalten, sei es die Bauteilsubstanz (Strukturmechanik) selbst, deren fluide Umgebung (Strömungsmechanik) oder gar beide zusammen innerhalb einer Fluid-Struktur-Interaktion, im Vordergrund. Dieses und die folgenden zwei Kapitel beschäftigen sich vornehmlich mit computergestützten Strömungssimulationen zur Unterstützung von Produktentwicklungsprozessen.

2.1 Grundprinzipien des Software-Engineerings

Die Entwicklung von Software im Rahmen von Software-Projekten wird typischerweise in einem Vorgehensmodell, welches sich im übrigen prinzipiell auf nahezu alle Arten von Projekten anwenden lässt, definiert. Die Basis des Vorgehens wird in [77] ausführlich beschrieben und im Folgenden zusammengefasst dargestellt.

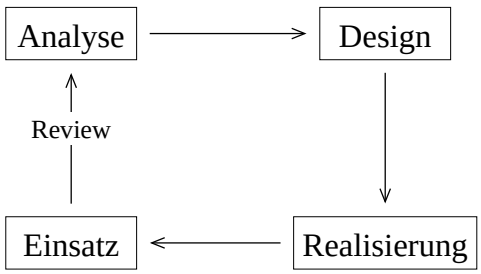

Abbildung 2.1: Produktionszyklus von Software

Der Zyklus des Vorgehensmodells wird in vier Haupteinheiten unterteilt:

Analyse: Die Entwicklung von Software beginnt immer mit der Analyse der Problemstellung der Anwendungswelt. IST-Zustände, die entweder durch schon vorhandene Software oder durch real existierende Abläufe dieser Anwendungswelt vorliegen, müssen möglichst genau erkannt und definiert werden. SOLL-Anwendungsfälle müssen entwickelt werden. Weiter sind alle Anforderungen und Zielvorstellungen bezüglich Hard- und Software festzustellen. In der Endphase der Analyse entsteht eine Prozesskette, die alle Anforderungen erfüllen soll.

Design: In der Design-Phase wird ein abstraktes Modell zur Rekonstruktion der Anwendungswelt mit Mitteln der Informatik entwickelt. Im Falle einer objektorientierten Programmiertechnik[1] entsteht dabei ein Klassenmodell meist in Kombination mit Sequenz- und Zustandsdiagrammen, in dem ein Beziehungsgeflecht die Zusammenhänge zwischen den Objekten der Klassen definiert. Die Klassen werden in einem Schichtenmodell angeordnet:

- Dialogschicht:
  Diese Schicht dient der Kommunikation zwischen der Software und dem Benutzer.
- Vorgangsschicht:
  In der Vorgangsschicht wird der Arbeitsablauf gesteuert. Sie ist für die Kommunikation der Dialog- und der Fachobjekte zuständig. Hierdurch wird weitestgehend eine Entkopplung der Fachklassen bewirkt.
- Fachklassenschicht:
  Hier werden die Attribute und Methoden der Anwendungswelt ebenso wie die Zusammenhänge und korrekten Beziehungen unter den Fachobjekten modelliert.

Dieses grundlegende Konzept ist prinzipiell anzustreben, aber nicht zwingend Voraussetzung für die Entwicklung von Software.

Realisierung: Der Konzeption folgt die Implementierung, basierend auf dem Design-Ergebnis. Die Klassen werden codiert und fortlaufend in ein Teil- bzw. Hauptsystem eingebettet. Vor allem die Wiederverwendbarkeit von Software muss essentieller Anspruch jedes Entwicklers sein, da sich somit der Implementierungsaufwand langfristig reduziert, der Wartungsaufwand ab- und die Zuverlässigkeit der Software zunimmt.

Bei Erstimplementierungen, wie es in der Forschung üblich ist, ist im Rahmen des *experimentellen Prototyping* zunächst die Tauglichkeit der modellierten Lösung nachzuweisen. Eine schrittweise Näherung an das optimierte

[1] In dieser Arbeit wird ausschließlich die objektorientierte Programmiersprache C++ verwendet.

Endprodukt ist beim Einsatz von objektorientierten Sprachen mithilfe eines evolutionären Vorgehens denkbar. Die Realisierung wird mithilfe von definierten SOLL-Anwendungsfällen, den sogenannten Benchmarks, abgeschlossen.

Je nach Umfang und Komplexität des Software-Projektes gewinnt ein professionelles Projektmanagement[2] an Bedeutung.

Einsatz/Review: In der Anwendung durch entsprechende Benutzer sind die Erfahrungen aus dem operativen Einsatz in Reviews zusammenzufassen. Diese dienen dann als Input für die Analyse eines neuen Software-Entwicklungszyklus.

2.2 Produktentwicklungszyklen mit CFD

Mithilfe von computergestützten Strömungssimulationen (CFD[3]) soll der Entwicklungszyklus von Produkten (z. B. Gebäuden oder Autos) verbessert werden. Abbildung 2.2 gliedert CFD in den Produktentwicklungsprozess ein. Mehrere

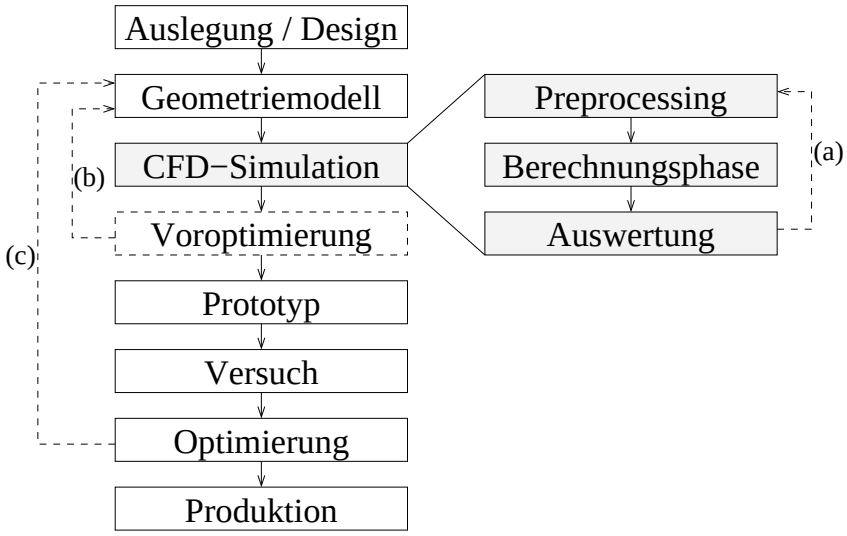

Abbildung 2.2: Produktentwicklungsprozess

Iterationsschleifen (gestrichelte Pfeile) sind erkennbar. Die *innerste* Schleife (a) dient der Anpassung der Simulationsparameter und der lokalen Auflösung durch selbstadaptive Gitter an die vorhandenen Strömungsverhältnisse (vgl. Kapitel 8). Idealerweise ist hierfür keine Interaktion mit dem Anwender notwendig. Die *mittlere* Iterationsschleife (b) stellt eine detaillierte Strömungsberechnung dar, in der

[2]Die Aufgaben des Projektmanagements sind nicht Inhalt dieser Arbeit.
[3]Computational Fluid Dynamics

mithilfe der erhaltenen Berechnungsergebnisse die zu umströmende Geometrie optimiert und die Berechnung erneut gestartet wird. Folgende Ziele werden im Vergleich zu einem korrespondierenden Produktentwicklungszyklus, in dem auf eine CFD-Simulation verzichtet wird, angestrebt:

- Voroptimierung mit CFD
- Geringere Anzahl von Prototypen
- Gezielte Optimierung.

Aus wirtschaftlicher Sicht ergeben sich somit einige weitere Vorteile:

- Höhere Qualität
- Kostenersparnis
- Zeitersparnis.

Während im Maschinenbau meist viele Produkte eines Typs (z. B. Automobile) hergestellt werden, steht im Bauwesen am Ende des Produktentwicklungsprozesses meist ein Unikat. Die *äußere* Iterationsschleife (c) entfällt hier. Daher ist hier eine Voroptimierung umso wichtiger, da der Prototyp gleichzeitig Endprodukt ist. Durch späte Entscheidungen, z. B. Verschieben einer Klimaanlage, entstehen Mehrkosten, die in der Praxis etwa 20 % der Gesamtproduktionskosten ausmachen können [81].

Die Entwicklung von mathematischen Modellen und entsprechender numerischer Verfahren zur Berechnung von Strömungen ermöglicht schon heute die Vorhersage vieler Strömungsphänomene. Aufgrund der Tatsache, dass Strömungssimulationen für Ingenieurprobleme größte Ansprüche an die Leistungsfähigkeit von Hardware und Software stellen, da selbst für Standardberechnungen von zahlreichen praxisrelevanten Anwendungsfällen mit extrem vielen Freiheitsgraden ($\mathcal{O}(10^6 - 10^9)$) gerechnet werden muss, besteht weiterhin ein Bedarf an Weiterentwicklungen und Optimierungen.

2.3 Integrationsziele

Die Hauptaufgabe des Software-Engineerings besteht in der Integration aller Teilaufgaben zu einem zusammengehörigen System, welches den gesamten Planungsprozess, beginnend mit dem Entwurf des Architekten über die Ingenieurdienstleistung bis hin zur Bauüberwachung unterstützt. Die in Kapitel 4 entwickelten und beschriebenen Integrationsansätze beschränken sich auf die drei Hauptphasen von Strömungssimulationen, dem Preprocessing, der Berechnungsphase und dem Postprocessing. Das Ziel besteht darin, einen Informationsaustausch zwischen den einzelnen Teilprozessen ohne signifikanten Datenverlust zu gewährleisten. Für kombinierte Verfahren zur Simulation von Fluid-Struktur-Interaktion

ist zudem ein bidirektionaler Datenaustausch zwischen den Teilprozessen und deren geometrischen Modelle entscheidend.

Die Konsistenz der Datenmodelle ist stets erforderlich und zu prüfen. Dies stellt ein großes Problem dar, da Software für die verschiedenen Teilprozesse in der Vergangenheit meist ohne gegenseitigen Bezug entwickelt wurden und unterschiedliche Ansprüche an die jeweilige Datenstruktur hatten. Um die Geometrie in ein sinnvolles Rahmenwerk, in dem prozessbedingt Beziehungen identifiziert werden können, zu packen, werden die einzelnen Objekte mit zusätzlichen Attributen versehen. Das sogenannte *IFC*[4]-Produktmodell ist ein vielversprechender Ansatz, Semantik und Geometrie im Bauwesen zu kombinieren [56]. In Kapitel 4 wird gezeigt, dass das geometrische Modell das Bindeglied aller Teilprozesse ist. Basierend auf dem IFC-Produktmodell wird ein umfangreicheres Produktmodell speziell für die Belange numerischer Simulationen konstruiert, in dem Spacetrees eine zentrale Rolle zur Kopplung der Teilprozesse einnehmen.

2.4 Anforderungen an die Implementierung

Im wissenschaftlichen Rechnen werden hauptsächlich die Programmiersprachen Fortran und C verwendet. Diese Sprachen eignen sich sehr gut für die Bearbeitung von mehrdimensionalen Feldern, wie sie typischerweise für numerische Simulationen eingesetzt werden. Mithilfe des gegebenen Sprachumfangs ist man teilweise in der Lage, objektorientierte Programme zu konzeptionieren und zu implementieren. Beispielsweise wurde das Finite-Elemente Programm *AdhoC*[5] unter Verwendung von Strukturen der Programmiersprache C teilweise objektorientiert implementiert. Dennoch bieten Sprachen wie C++, welche die Objektorientierung explizit unterstützen, die Möglichkeit die *Menschen-bezogene* Sichtweise deutlich besser nachzubilden. Einen Überblick über den Einsatz von Programmiersprachen wird in [2, 26, 98] gegeben.

Erst in den letzten Jahren hat die Programmiersprache C++ Einzug in das wissenschaftliche Rechnen gefunden. Ein Grund hierfür ist das Fehlen von Compilern, die mit denen der Sprachen Fortran und C mithalten können. Erst in den letzten Jahren wurden zunehmend Compiler zur Erzeugung von effizienten C++-basierten Programmen entwickelt. Zudem bedingen C++-Features wie Polymorphie, Überladen von Operatoren, Ableitung von Klassen deutliche Einbussen der Performance während der Laufzeit [97]. Für den Software-Ingenieur hat dies zur Folge, dass auf diese C++-Features nach Möglichkeit dann verzichtet werden sollte, wenn es sich um numerisch aufwendige Prozeduren handelt. Dagegen kann für Programmelemente, die nur selten abgearbeitet werden oder deren Ausführungszeit sehr gering ist, der volle Sprachumfang eingesetzt werden, um

[4]Industry Foundation Classes, http://www.iai-ev.de

[5]Am Lehrstuhl für Bauinformatik der TU München entwickelter FE-Forschungscode

die Übersichtlichkeit der Programmstruktur zu verbessern. Je näher man sich am Rechenkern befindet, umso weniger objektorientiert sollte ein Code sein.

Bei der Entwicklung von Programmen sollte weiterhin auf die Wiederverwendbarkeit, die Wartung und die Erweiterbarkeit von Code geachtet werden. Dies wird ebenfalls durch den Einsatz von C++ unterstützt.

Kapitel 3

Geometrische Modelle im Überblick

Alle Teilprozesse des in Kapitel 4 präsentierten Integrationskonzepts werden von geometrischen Modellen unterstützt und geprägt. Dieser Abschnitt verschafft dem Leser einen Überblick über die verwendeten geometrischen Modelle und die entsprechenden Datenstrukturen. Der Schwerpunkt wird insbesondere auf jene Modelle gelegt, die sich an den Achsen eines kartesischen Koordinatensystem orientieren. Der Grund hierfür liegt in der Natur des verwendeten Strömungssimulations-Verfahren, dessen Berechnungsgrundlage ein kartesisches Gitter ist (vgl. Kapitel 5). Die Baumdatenstrukturen, als Vertreter der raumpartitionierenden Datenstrukturen, werden wegen ihrer zentralen Rolle in der Modellkette detailliert vorgestellt.

3.1 Indirekte Darstellungsschemata

Indirekte Darstellungsschemata zeichnen sich dadurch aus, dass die Volumina der Körper *indirekt*, über die Oberflächen der Objekte, modelliert werden. Die Grundprimitive sind Punkte, Kanten und Flächen. Um die Topologie der Körperoberflächen zu beschreiben, werden in der Regel *vef*-Graphen[1] verwendet [11]. Sinnvollerweise ist die Speicherstruktur meist identisch mit der topologischen Struktur der Geometrie.

Ein Oberflächenmodell repräsentiert nur dann einen Körper, wenn folgende Gültigkeitsbedingungen erfüllt sind:

Geschlossene Volumina: Für eine eindeutige Aussage, ob sich ein beliebiger Punkt innerhalb oder außerhalb eines Körpers befindet, muss dieser eine geschlossene Oberfläche haben.

[1] vef = vertex edge face

Orientierbarkeit der Teilflächen: Die Orientierung der Flächen ist ebenfalls notwendig und meist durch Angabe des Normalenvektors gegeben.

Für das Preprocessing in Strömungssimulationen sind diese Bedingungen sowohl für die Modellierung der Geometrie im CAD-System, als auch für die Gittergenerierung von entscheidender Bedeutung (vgl. Kapitel 4.2.1).

3.1.1 Boundary-Representation-Model

Viele Geometriemodellierer und CAD-Systeme basieren auf den *Boundary-Representation* Datenstrukturen (B-rep). Der Vorteil liegt neben der Möglichkeit auf einzelne Teilflächen zuzugreifen und diese beispielsweise mit zusätzlichen Attributen zu versehen auf der Exaktheit[2], mit der die Form eines Körpers beschrieben wird. Hier hat der Einsatz von Freiformflächen die Mächtigkeit der Modellierer deutlich erhöht. Nachteilig wirkt sich die Notwendigkeit der Implementierung

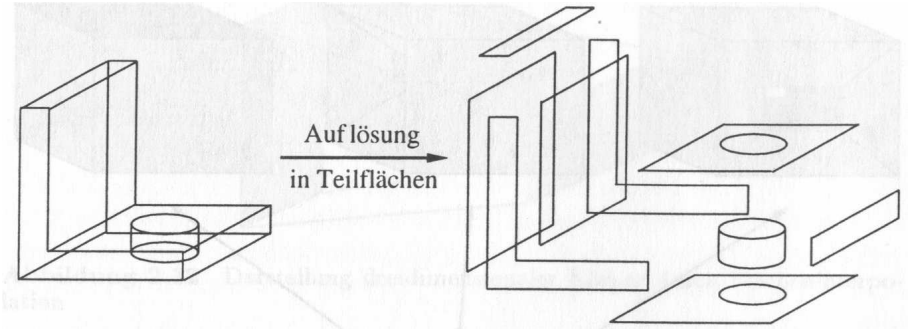

Abbildung 3.1: B-rep Modell aus [11], Seite 66

der umfangreichen und relativ komplexen Datenstrukturen, deren Grundstruktur auf vef-Graphen basiert, aus. Zudem müssen stets die in Kapitel 3.1 genannten Gültigkeitsbedingungen erfüllt bleiben.

3.1.2 Facettenmodelle

Bei Facettenmodellen handelt es sich um einen Spezialfall der B-rep-Modelle. Alle Oberflächen eines Körpers werden nicht mehr mit analytischen Beschreibungen der Geometrie, sondern nur mehr mit Dreiecken und/oder Vierecken[3] in Form von Oberflächennetzen dargestellt (vgl. Abb. 3.2). Daraus folgt ein Verlust der Genauigkeit der repräsentierten Geometrie und in vielen Fällen, vor allem bei gekrümmten Oberflächen ein deutlich höherer Speicherverbrauch. Der Genauigkeitsverlust kann durch Verfeinerung der Facetten beliebig reduziert werden.

[2]Die Exaktheit ist auf die Mächtigkeit der Beschreibungen der Teilflächen beschränkt.
[3]Diese Arbeit verwendet nur Dreieckselemente.

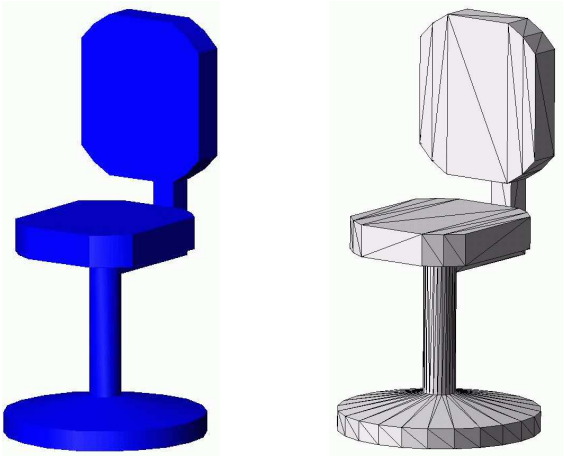

Abbildung 3.2: Schattiertes CAD- (links) und Facettenmodell (rechts) eines Bürostuhls

Lokale Modifikation sind nun noch einfacher umsetzbar. Die Komplexität der Datenhaltung, welche wieder auf vef-Graphen basiert, wird durch die Wahl einfacher Geometriebeschreibungen ebenfalls deutlich abgemindert.

3.2 Direkte Darstellungsschemata

Bei den direkten Darstellungsschemata wird der Körper durch die geometrischen Primitive selbst beschrieben, indem das Körpervolumen durch elementares Zusammensetzen der Teilvolumina erzeugt wird.

3.2.1 CSG-Schema

Das in den meisten CAD-Systemen vorhandene CSG[4]-Schema ist konstruktionsinspiriert und prinzipiell leicht interaktiv bedienbar. Wie bereits erwähnt, ist der Körper das Ergebnis sukzessiver Mengenoperationen mit einfachen Primitiven. Bei den Mengenoperationen handelt es sich um die Bool'schen Operationen

- Schnittmenge $\cap$
- Vereinigungsmenge $\cup$
- Differenz $\setminus$

Mithilfe eines Konstruktionsbaums können in Abhängigkeit von der Komplexität der Primitive nahezu beliebig geformte Körper erzeugt werden. Der Konstruk-

[4]Constructive Solid Geometry

tionsbaum ist ein *Binärbaum*, dessen Knoten die Bool'sche Operationen und dessen Kanten die korrespondierenden Operanden anzeigen (vgl. Abb. 3.3). Die Viel-

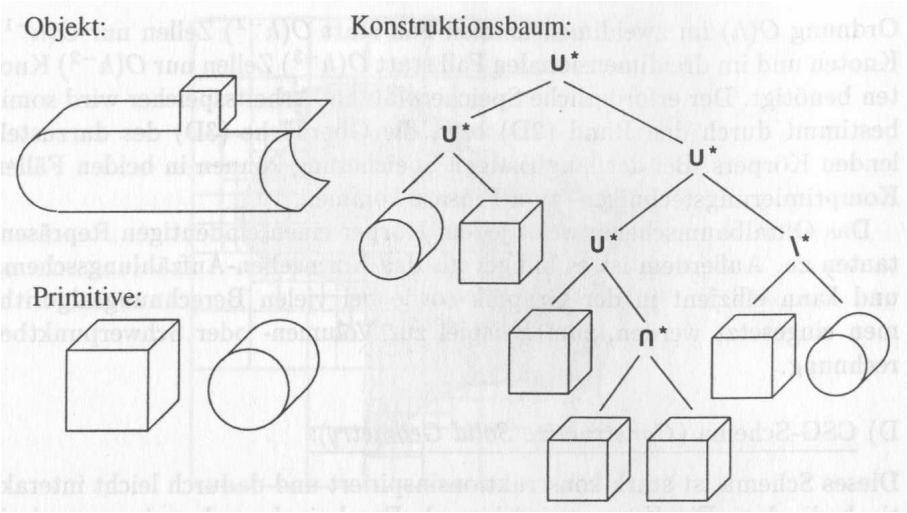

Abbildung 3.3: CSG-Schema mit Konstruktionsbaum aus [11], Seite 62

falt der Primitive kann erhöht werden, indem vorhandene Primitive durch die Transformationsvorschriften *Translation*, *Rotation* und *Skalierung* sowie durch *Extrusionstechniken* erweitert werden.

3.2.2 Normzellen-Aufzählungsschema

Ein Raum wird in ein Gitter, bestehend aus gleich großen Zellen, zerlegt. Für 3D-Kontinua sind dies meist Würfel, für den 2D-Fall Quadrate. Die Zellen befinden sich entweder außerhalb eines Körpers oder nicht. Somit reicht eine Bit-Matrix zur Speicherung des Normzellen-Aufzählungsschemas aus (vgl. Abb. 3.4). Alternativ wäre es auch denkbar, jene Zellen, die den Rand der Geometrie schneiden, gesondert zu markieren. Mit abnehmender Kantenlänge der Zellen nimmt die Approximationsgüte zu, womit sich bei iterativer, homogener Zellverfeinerung eine beliebig genaue Annäherung der wahren Geometrie erreichen läßt. Da die Speicherkomplexität zweiter Ordnung für 2D und dritter Ordnung für 3D-Darstellungen ist, vervierfacht bzw. verachtfacht sich der Speicherverbrauch pro Verfeinerungsiteration.

Dieses Schema eignet sich hervorragend zur Darstellung von strukturierten und uniformen Gittern. Durch die sich regelmäßig wiederholende Topologie ist die Implementierung von Funktionen, die auf diese Gitter angewendet werden, relativ einfach und die Entwicklung von Applikationen wird somit erleichtert.

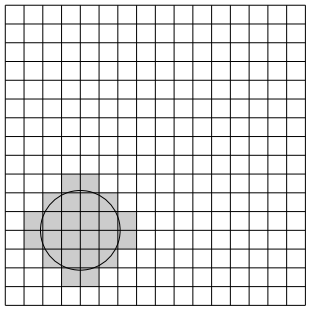

Abbildung 3.4: Uniform aufgelöster Kreis

3.2.3 Blockstrukturiertes Normzellen-Aufzählungsschema

Blockstrukturierte Gitter lassen sich als Verbund von strukturierten Gittern (meist rechteckige Blöcke) charakterisieren (vgl. Abb. 3.5). Sie erben teilweise die positi-

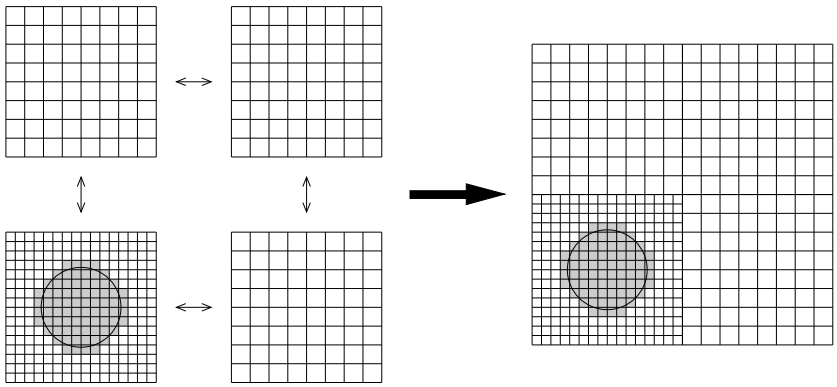

Abbildung 3.5: Blockstrukturiertes Gitter

ven Eigenschaften der strukturierten Gitter bezüglich der Effizienz der erzeugenden Berechnungsalgorithmen, benötigen allerdings einen großen Rechenaufwand zur Kommunikation zwischen den einzelnen Blöcken. Durch die Möglichkeit, verschiedene Blöcke nahezu unabhängig voneinander zu modellieren und zu diskretisieren, lassen sich komplexe geometrische Objekte einfacher vernetzen. Typischerweise werden blockstrukturierte Gitter für numerisch höher aufzulösende Regionen verwendet [27].

3.2.4 Hybride Schemata

Oft werden die Eigenschaften der CSG-Schemata und der B-rep-Schemata für die Modellierung von Körpern benötigt. Viele Modellierer und CAD-Systeme,

wie beispielsweise der ACIS-Kernel[5] und AutoCAD[6], halten die Geometrien in beiden Formen durch Transformation des CSG-Schemas auf das B-rep-Schema vor. Die Abbildung wird häufig unter Verwendung von Oktalbäumen (Octree), den dreidimensionalen Vertretern der Baumdatenstrukturen, als Hilfskonstruktionen unterstützt [49, 62]. Ein solches Schema wird in [11] vorgeschlagen und in Kapitel 4 kontextbezogen entwickelt und umgesetzt.

3.3 Baumdatenstrukturen - Spacetrees

Die zu den direkten Darstellungsschemata zählenden Baumdatenstrukturen nehmen in dieser Arbeit eine zentrale Rolle ein und werden daher in diesem Kapitel detailliert beschrieben. Wegen der besseren Darstellbarkeit in den Abbildungen werden alle visuellen Erklärung anhand ihrer zweidimensionaler Vertreter, den Quadralbäumen (Quadtree), vorgenommen.

3.3.1 Terminologie

Baumdatenstrukturen, wegen des ihnen immanenten raumorganisierenden Prinzips auch Spacetrees genannt, sind hierarchisch adaptive Datenstrukturen und eignen sich hervorragend zur Darstellung von Objekten im Raum [16, 107]. Die Elemente der Spacetrees zeichnen sich dadurch aus, dass sie mit Ausnahme der Wurzel genau ein Vorgängerelement und 2^D (D = Anzahl der Raumdimensionen) Folgeelemente, deren lokale Position im Elternelement genau bestimmt ist, besitzen. Blätter werden definitionsgemäß nicht weiter aufgeteilt. Die Unterteilung ist uniform und selbstähnlich, so dass die Längenverhältnisse der Kind-Zellen mit

$$\frac{l_{k_1}}{l_{k_2}} = \left(\frac{1}{2}\right)^{\Delta k} \quad \text{mit} \quad \Delta k = k_1 - k_2 \quad \text{wobei} \quad k_i = \text{Baumtiefe} \tag{3.1}$$

gegeben ist. Somit wird der Auflösungsgrad der dargestellten Körper durch die Wahl der Baumtiefe vorgegeben. Die Nomenklatur der bekanntesten Baumdatenstrukturen ist in tabellarischer Form gegeben:

Bezeichnung	Liste	Binary Tree	Quadtree	Octree
Unterteilungen	$1 = 2^0$	$2 = 2^1$	$4 = 2^2$	$8 = 2^3$

Tabelle 3.1: Nomenklatur der Baumdatenstrukturen

Weiter müssen folgende Voraussetzungen erfüllt sein, wenn man von einer Baumdatenstruktur spricht:

[5] www.spatial.com
[6] www.autodesk.com

1. Die Elemente müssen vom gleichen Datentyp sein. Diese Voraussetzung ist extrem wichtig für die Anwendung von rekursiven Algorithmen.

2. Es muss genau einen Knoten geben, von dem aus *jeder* andere gefunden werden kann. Dieser Knoten wird auch *Wurzel* genannt. Die Wurzel repräsentiert das Gesamtuniversum (Ebene, Raum), das dann je nach Belieben rekursiv in Teilräume unterteilt wird. Alternativ können auch mehrere Bäume einen Gesamtraum erzeugen. In diesem Fall gibt es mehrere Wurzelelemente.

3. Jedes Element darf nur **ein** Vorgängerelement haben.

Die *Höhe* oder *Tiefe* eines Baumes ist bestimmt durch den maximalen Abstand eines Blattes von der Wurzel oder durch das größte Level. Die Elemente einer

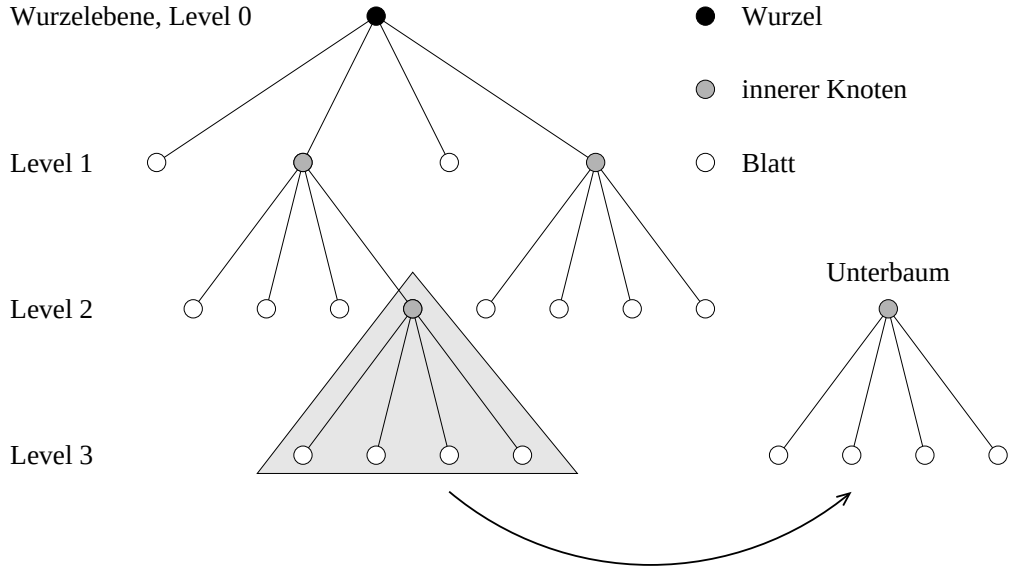

Abbildung 3.6: Terminologie von Baumdatenstrukturen

Baumdatenstruktur werden auch Knoten genannt. Zur Darstellung von Körpern werden die Knoten durch Quadranten (2D) oder durch Oktanten (3D) ersetzt. Neben den geometrischen und topologischen Daten können die Elemente noch weitere Informationen, wie zum Beispiel Ergebniswerte, Systemvariablen, die Lage (innen/außen), usw. enthalten [15].

Ist der Größenunterschied zweier benachbarter Elemente (im geometrischen Sinne) stets kleiner oder gleich als eine fest vorgegebene Schranke (üblicherweise ein Faktor 2), so spricht man von einem geglätteten oder balancierten Baum (vgl. Kapitel 4.4.3.1).

3.3.2 Eigenschaften

Ein wichtiges Merkmal von Baumdatenstrukturen ist die Möglichkeit der Unterteilung des Baumes in Unterbäume. Dabei wird ein bestimmter Knoten (mit allen Folgeelementen) aus einem Baum geschnitten, wodurch er selbst zur Wurzel des somit erzeugten Unterbaumes wird (vgl. Abb. 3.6). Da diese Abtrennung prinzipiell an jedem Knoten erfolgen kann, stellt jeder Knoten potentiell eine Wurzel dar und ist in entsprechender Weise von jeder Funktion gleich zu behandeln. Dies ist Voraussetzung für die Anwendung von rekursiven Algorithmen, welche eine der größten Stärken der Baumdatenstruktur darstellen.

Bezüglich der Speicherkomplexität ist ein Spacetree anderen Darstellungsschemata, wie zum Beispiel dem Normzellenschema, überlegen. Während das Normzellenschema eine Speicherkomplexität von $\mathcal{O}(n^D)$ (D = Anzahl der Raumdimensionen) hat, wird beim Spacetree nur noch eine Speicherkomplexität von $\mathcal{O}(n^{D-1})$ erforderlich. Der Grund hierfür liegt darin, dass nur noch die Berandung des Körpers bearbeitet und geometrisch diskretisiert werden muss (*tree- complexity bound theorem*). Dies kann in den Abbildungen 3.5 und 3.8 beobachtet werden. Die Auflösung der Kreise ist in beiden Fällen identisch.

Für weitere Eigenschaften wird an dieser Stelle auf die Dissertation von Anton Frank [33] verwiesen.

3.3.3 Graphentheorie - Speichermodelle

Jeder Spacetree lässt sich durch einen gerichteten, azyklischen Graphen repräsentieren. Wegen der strengen Hierarchie existiert genau ein Pfad, der den Wurzel-Knoten mit einem beliebigen Ziel-Knoten verbindet. Alle Kanten des Graphen werden mit der Adjazenzrelation

$$vv \subseteq V \times V \tag{3.2}$$

beschrieben. v stellt dabei einen Knoten aus der Menge aller Knoten V dar. Die Menge vv enthält nur Knotenpaare, die entsprechend der Hierarchie in Richtung der größeren Baumtiefe zeigen. Werden auch Knotenpaare zugelassen, die auf die Vorgängerelemente zeigen, wird die graphentheoretische Definition eines Baumes zwar gebrochen, die Flexibilität des Spacetrees aber deutlich erhöht, da nun jeder Knoten mit jedem anderen beliebigen Knoten über einen eindeutigen Pfad miteinander verbunden werden kann. Der Graph ist nun ungerichtet, aber nach wie vor zyklenfrei.

Für Spacetrees, die während der Laufzeit eines Programms substantiell benötigt werden, eignet sich die Implementierung eines bidirektionalen Zeigergeflechts. Nachteilig wirkt sich der hohe Speicherverbrauch für die Verzeigerung aus. Von Vorteil ist jedoch, dass der Spacetree zur Laufzeit leicht manipuliert werden kann. Mithilfe der in Kapitel 4.4.3.1 vorgestellten Techniken der Nachbarsuche (bidirektionale Baumtraversierung), kann in der Zeit $\mathcal{O}(2h)$ (h = Höhe des Baumes)

ausgehend von jedem beliebigen Knoten auf jeden Knoten zugegriffen werden. Zudem ist es die naheliegendste Methode zur Implementierung des Graphen, da die Hierarchie vollständig wiedergegeben wird.

3.3.4 Erzeugen der Spacetrees

Wegen der Gleichbehandlung der Dimensionen, also durch sukzessives Halbieren der Zellen in alle Raumrichtungen, lassen sich elegante, skaleninvariante und effiziente Algorithmen durch Verwendung von rekursiven Funktionen implementieren. Hierbei wird die Divide-and-Conquer-Strategie verfolgt und durch die rekursiven Eigenschaften der Baumdatenstrukturen unterstützt. In den Abbildungen 3.7 und 3.8 wird dieser Vorgang illustriert.

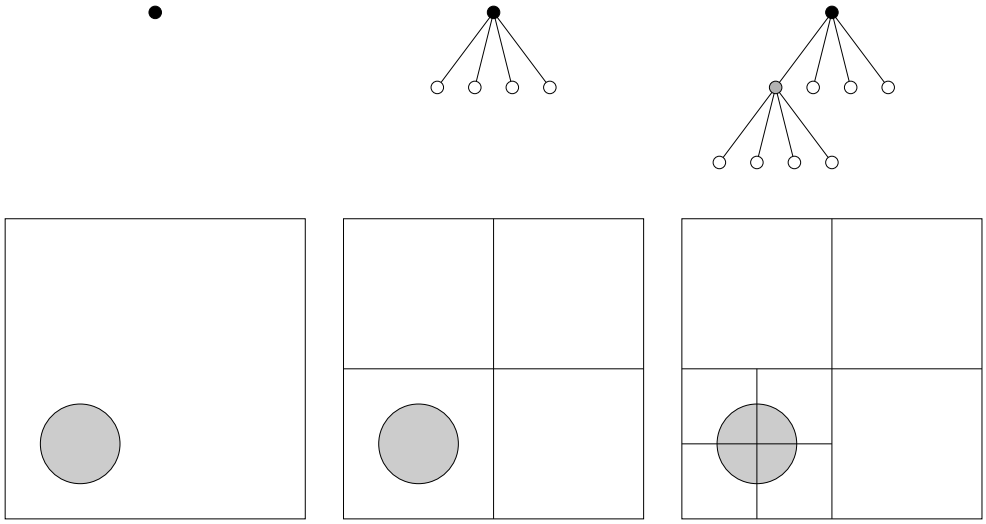

Abbildung 3.7: Quadtreeaufbau - Gitterverfeinerung

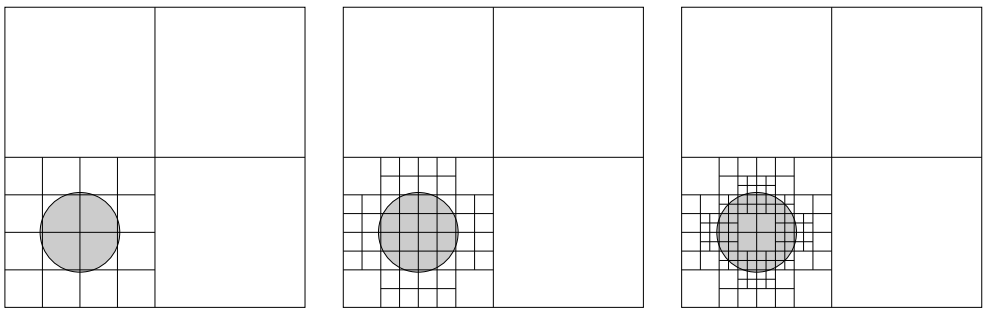

Abbildung 3.8: Weitere Gitterverfeinerungsschritte

Ausgehend von einem geometrischen Modell werden in der *geometrischen Diskretisierung* nur die Ränder der Teilobjekte approximiert. Hierbei werden jene Elemente des Spacetrees verfeinert, also durch vier Quadranten in 2D und acht Oktanten in 3D ersetzt, die eine nicht-leere Schnittmenge mit dem Rand eines geometrischen Objektes aufweisen. Die Reduktion der algorithmischen Komplexität der Schnittmengenoperationen wird deutlich, wenn man bedenkt, dass anstelle eines Fläche-Fläche-Vergleichs bzw. eines Volumen-Volumen-Vergleichs nur noch ein Fläche-Kante-Vergleich (Quadrat-Kante) bzw. ein Volumen-Fläche-Vergleich (Würfel-Fläche) notwendig wird. Ein weiterer Vorteil dieses Vorgehens besteht darin, dass lediglich der Rand geometrisch diskretisiert wird und damit für deutlich weniger Zellen eine Schnittmengenoperation berechnet werden muss, als dies für ein Normzellenschema der Fall wäre. Hierdurch wird der Aufwand um eine Dimensionsordnung reduziert.

Kapitel 4

Kopplung von CAD und Strömungssimulation

Nachdem in den vorangegangenen Kapitel einige elementare Grundlagen des Software-Engineerings und von geometrischen Modelle besprochen wurden, sollen nun Techniken zur Kopplung eines CAD-Systems mit dem gitterbasierten Strömungssimulator *VirtualFluids* (vgl. Kapitel 4.3.3.2) betrachtet werden. Der Fokus wird vor allem auf die geometrische Anbindung unter Verwendung der Baumdatenstrukturen sowie auf deren algorithmische und strukturelle Anpassung gelegt. Die Erweiterung des Rahmenwerkes für die Visualisierung und Auswertung wird ebenfalls kurz erläutert. Eine detaillierte Diskussion hierüber ist in [51] zu finden.

Mithilfe der in Kapitel 2 vorgestellten Techniken des Software-Engineerings soll anhand des Vorgehensmodells ein Integrationskonzept zur Optimierung des Ablaufs von Strömungssimulationen entwickelt und implementiert werden.

4.1 Überblick: Gesamtablauf einer Strömungssimulation

Der Gesamtablauf einer strömungsmechanischen Simulation lässt sich in drei wesentliche Aufgabenbereiche unterteilen [18, 19] (vgl. Abb. 2.2):

- Preprocessing
- Solving
- Postprocessing

Beim Preprocessing wird das Strömungsgebiet sowie die zu umströmenden Körper typischerweise mit einem CAD-System geometrisch modelliert. Als nächstes muss dieses Modell transformiert werden, d.h. die CAD-Objekte müssen auf Objekte abgebildet werden, die mit den Objekten des eigentlichen Berechnungskerns,

meist Knoten oder Elemente, kompatibel sind. Dieser Vorgang wird gewöhnlich als Vernetzung bezeichnet. Hierbei werden sinnvollerweise die strömungsmechanischen Randbedingungen den Knoten oder Elementen als Attribute zugewiesen. Diese Informationen werden dann z. B. über eine Eingabedatei dem Berechnungskern zur Verfügung gestellt.

In der Berechnungsphase wird das diskretisierte Strömungsproblem mit einem numerischen Verfahren, in diesem Projekt der sogenannten *Lattice-Boltzmann Methode*, berechnet. Die durch diesen Vorgang erzeugten Ergebnisdateien enthalten typischerweise die Komponenten der Geschwindigkeitsvektoren sowie den Druck an allen Knoten für eine endliche Menge von Zeitpunkten.

Beim Postprocessing werden diese Ergebnisdaten eingelesen und mit effizienten Algorithmen aufbereitet, um eine aussagekräftige Visualisierung der Primärvariablen und weiterer abgeleiteter Eigenschaften (Vortizität, Schubspannungen) sowie Komfortgrößen [53] zu erreichen. Aus diesen Größen werden je nach Visualisierungsart wiederum geometrische Objekte erzeugt (z. B. Geschwindigkeitsvektoren oder Stromlinien) und anschließend auf dem Bildschirm dargestellt. Für diesen Zweck kommen Werkzeuge, wie zum Beispiel das Visualisierungs- und Entwicklungspaket AVS/Express [1, 17], zum Einsatz. Eine noch sehr neue Variante der Darstellung von 3D-Ergebnisdaten mit weitergehenden Möglichkeiten bietet eine Virtual-Reality-Umgebung [19, 51].

4.2 Analyse: Anforderungen an die Teilprozesse

Entscheidende Grundeigenschaft der eben beschriebenen Ablaufkette (Modellierung → numerische Berechnung → Auswertung/Visualisierung) sollte darin bestehen, das Gesamtkonzept so zu gestalten, dass mit einem Minimum an Schnittstellen zwischen den Einzelprozessen ein Maximum an Flexibilität erreicht wird. Nach Möglichkeit wird ein hohes Maß an Automatisierung bei der Transformation zwischen den spezifischen Objekttypen angestrebt. Da die drei eben erwähnten Aufgabenbereiche unterschiedliche Ansprüche an die Hardware stellen, ist eine alle Teilprozesse integrierende Einzelapplikation für quantitative ingenieurrelevante Fragestellungen gegenwärtig noch nicht realistisch. Für Voruntersuchungen, in denen vornehmlich qualitative Aussagen geliefert werden sollen, wurde in [3, 51] eine Einzelapplikation in Form eines Metacomputing Systems[1] implementiert.

Neben den Integrationszielen (vgl. Kapitel 2.3) und den Implementierungsanforderung (vgl. Kapitel 2.4) werden in den drei Aufgabenbereichen weitere Kriterien erwünscht.

[1] Metacomputing = Kopplung von unterschiedlichen Plattformen

4.2.1 Preprocessing

Die Hauptaufgaben eines Preprocessors sind die geometrische Modellierung, die Definition von physikalischen Randbedingungen, die Diskretisierung des Berechnungsgebietes sowie die Erzeugung und Übergabe der Eingabedaten an den Strömungssimulator.

Die geometrische Modellierung wird im Ingenieurwesen üblicherweise unter Verwendung von CAD-Systemen mithilfe eines gewöhnlichen Arbeitsplatzrechners durchgeführt. Die Modellierer sollten es erlauben, möglichst frei gestaltete, räumliche Strukturen und somit realitätsnahe Modelle [10] zu erzeugen. Die physikalischen Randbedingungen werden idealerweise in der selben CAD-Umgebung modelliert. Der Vorteil ergibt sich aus der Möglichkeit der interaktiven Attributierung der Objekte und der Konsistenzprüfung während der Modellierung [40, 41].

Für viele Simulationsverfahren (z. B. FEM, FVM) ist die Gittergenerierung immer noch der Flaschenhals in applikationsorientierten CFD-Simulationen [22], da es oft nicht möglich ist, gute, nur aus Hexaeder-Elementen bestehende, Berechnungsnetze automatisch zu erzeugen [100, 101]. Durch die Forderung komplexe, nicht mehr durch einfache geometrische Objekte darstellbare Geometrieformen vernetzen zu wollen, ist eine semiautomatische Gittergenerierung entscheidend geworden. Folgende Anforderung ergeben sich:

Zuverlässigkeit: Ein fehlerfreies Gitter ist eine Voraussetzung dafür, dass die Erwartungen, die aufgrund von theoretischen Überlegungen an die numerische Berechnung gestellt werden, überhaupt erfüllt werden können.

Berechnungszeit: Speziell bei numerischen Berechnungen im Bereich der Strömungsmechanik sind hier hohe Anforderungen zu stellen, da je nach strömungsmechanischer Problemstellung extrem viele Gitterknoten erforderlich werden.

Speicherverbrauch: Die bei der Diskretisierung erzeugten Elemente sollten möglichst effizient gespeichert werden. Eine strikte Minimierung des Speicherplatzes ist allerdings nicht zwingend anzustreben, da sich dies wiederum negativ auf die Berechnungszeit auswirken kann [15].

Flexibilität: Bei allen Berechnungsmethoden sollte die Gitterknotendichte in der Nähe von umströmten Objekten automatisch erhöht[2] bzw. eine adaptive Verfeinerung in Bereichen von großen Lösungsgradienten (oder Vergröberung in Bereichen von kleinen Gradienten) angestrebt werden (vgl. Kapitel 8).

Das Erzeugen der Eingabedaten für den Berechnungskern ist der letzte Schritt des Preprocessings. An dieser Stelle kann die Modellierungshardware verlassen

[2]Typischerweise erfolgt eine geometrisch motivierte Verfeinerung bezüglich der Hindernisobjekte.

und somit unter Umständen Eingabedateien für eine völlig neue Umgebung bezüglich Rechenarchitektur und Betriebssystem erzeugt werden. Es ist daher darauf zu achten, die erstellten Informationen für das neue System in richtiger Syntax (z.B. little/big Endian) mit einem möglichst geringen Verlust an Information und der Möglichkeit des Rücktransfers an die Modellierungsumgebung zu übergeben.

4.2.2 Berechnungsphase

Die Berechungsphase ist auch unter der Voraussetzung einer Programmierfehlerfreien Software mit einigen Fehlerquellen behaftet:

- Modellierungsfehler der physikalischen Gesetzmäßigkeiten (physikalische Realität $\Leftrightarrow$ mathematisches Modell)
- Diskretisierungsfehler des numerischen Verfahrens (bedingt durch die Gitterweite, die Zeitschrittweite)
- Abbruchfehler, nicht vollständig konvergierte Lösungen

Diese Fehler in der Modellbildung sollten auf ein Minimum reduziert werden. Der Berechnungskern sollte weiterhin ein hohes Maß an Stabilität bieten und mit zunehmender numerischer Auflösung gegen die exakte Lösung der Navier-Stokes Gleichungen konvergieren.

Da heute selbst auf Hochleistungsrechnern nach wie vor die Berechnungsdauer *der* limitierende Faktor für die Simulation großer Systeme ist, muss das Berechnungsverfahren effizient und parallelisierbar sein. Es ist davon auszugehen, dass die Berechnungsphase auch in absehbarer Zukunft auf parallelen Rechnerarchitekturen stattfinden wird.

Gleichzeitig sollte das Verfahren und die Implementierung bezüglich weiterer Modelle, wie zum Beispiel der Turbulenzmodellierung, der Kopplung mit zusätzlichen Transportgleichungen (Temperatur, Spezies) oder der Kopplung mit anderen Simulationsverfahren (Struktursimulation) integrationsfähig sein.

4.2.3 Postprocessing

Die klassische Vorgehensweise in der Visualisierung setzt sich aus dem Einlesen, der Reduktion und der Aufbereitung der Ergebnisdaten, der Generierung von grafischen Objekten und deren Darstellung auf dem Bildschirm zusammen. Dieser Schritt wird üblicherweise, nachdem ein entsprechender Datensatz vom Simulationskernel erzeugt wurde, durchgeführt. Während der Simulation werden zudem oft weitere Ergebniswerte zur quantitativen Bewertung der Simulation berechnet (vgl. Kapitel 7.5). Neben den grundsätzlichen Anforderung, zu denen unter anderen die echtzeitfähige, interaktive Visualisierung mit leistungsfähigen

Hardware-Komponenten, wie z. B. einer Holobench[3] gehört, wird zunehmend auch die quantitative und automatische Analyse der Daten gefordert (vgl. Kapitel 8). Im Zuge eines Optimierungszyklus soll die Auswertung Aussage darüber treffen, inwieweit beispielsweise ein geometrisches Modell angepasst werden muss (äußere Interaktionsschleife, Abb. 2.2).

4.3 Design: Modellkette vom Produktmodell zur Visualisierung

Aufgrund der teilweise grundsätzlichen Verschiedenheit der Modelle, die im Entwicklungszyklus vorkommen, ist es vermutlich nur schwer möglich, *ein Hypermodell*, das den verschiedenen Ansprüche genügt, zu identifizieren. In Abbildung 4.1 sind die Modellketten und der Datenfluss einer gitterbasierten Strömungssimulation (links) und einer FE-Strukturanalyse (rechts) dargestellt. Der Ablauf

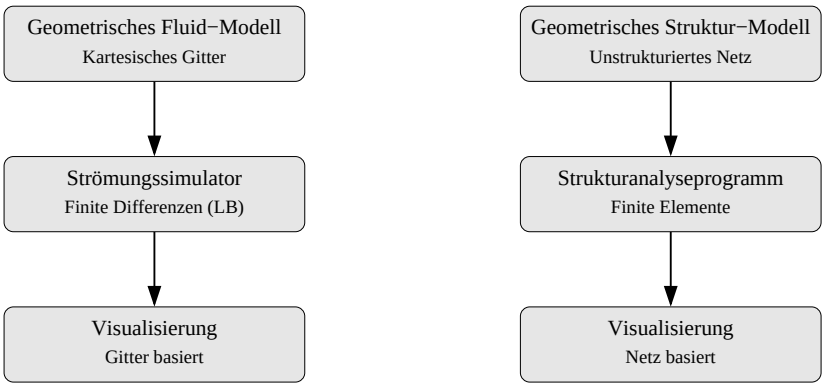

Abbildung 4.1: Modellkette und Datenfluss klassischer Simulationsverfahren

beider Simulationsmethoden basiert auf völlig unterschiedlichen geometrischen Darstellungsformen, dem Gitter für die Strömungssimulation und den Netzen für die Struktursimulation. Zur Unterstützung des Entwicklungsprozesses und des Gesamtablaufs werden folgende Ansätze vorgeschlagen:

- Gemeinsame Untermodelle, die in möglichst vielen Bereichen eingesetzt werden können, werden identifiziert.
- Es werden Techniken zur Verfügung gestellt, damit die verschiedenen Modelle möglichst effizient und verlustfrei aufeinander abgebildet werden können.
- Modelle werden von allgemeineren Basismodellen abgeleitet.

[3]Zwei Projektionsflächen ermöglichen die Darstellungen von vollwertigen 3D-Szenen.

Eine mögliche Lösung des Problems der Unvereinbarkeit der verschiedenen Darstellungsformen liegt darin, die Gemeinsamkeiten der verschiedenen Ablaufketten nicht in den numerischen Modellen, sondern eine Ebene darüber, in der Modellierung (geometrische und physikalische Ebene) zu suchen. Somit erhielte man für alle Beteiligten eine neutrale, nur aus geometrischen und topologischen Informationen bestehende Beschreibung des Bauteils oder des Gebäudes. In [53, 81] wird daher ein Rahmenwerk vorgestellt, in dem vier Modellebenen identifiziert werden:

- Geometrische Ebene
- Physikalische Ebene
- Numerische Ebene
- Präsentations- und Auswertungsebene.

Abbildung 4.2 zeigt das Gesamtkonzept, bestehend aus den Ablaufketten der Strömungs- und der Struktursimulation sowie existierender oder gewünschter Verbindungen beider Modelle. Die verschiedenen Ebenen stellen nicht nur die Kette und die Abhängigkeiten der geometrischen Modelle dar, sondern auch die Prozesskette und den Datenfluss.

4.3.1 Geometrische Ebene

Das geometrische Modell erhält man über das IFC-Produktmodell oder durch Modellierung in einem CAD-System (hier: AutoCAD2000). Neben den in Kapitel 4.2 aufgeführten Anforderungen ist vor allem auf die korrekte Modellierung unter Verwendung von 3D-Solid-Körpern (CSG-Schema) zu achten, um später die Möglichkeit zu haben, eindeutig außen von innen zu unterscheiden. Das B-rep-Modell wird automatisch von einem CAD-System (z. B. AutoCAD) erzeugt[4].

4.3.2 Physikalische Ebene

Das Geometriemodell wird in der physikalischen Ebene in eine praxisnahe physikalische Umgebung eingebettet. Hierfür wurde für den Bereich der Strömungssimulationen ein computergestützter Windkanal (WTM-Modul[5]), basierend auf AutoCAD2000 und der ObjectARX[6], entwickelt [40]. Die geometrischen Objekte werden interaktiv und GUI[7]-unterstützt (vgl. Abb. 4.3) mit Attributen zur

[4]Umgekehrt kann im Allgemeinen ein Volumenmodell basierend auf einem B-rep-Modell nicht erzeugt werden.

[5]WTM = Wind Tunnel Modeller

[6]C++-Programmierschnittstelle für AutoCAD [15]

[7]GUI = Graphical User Interface

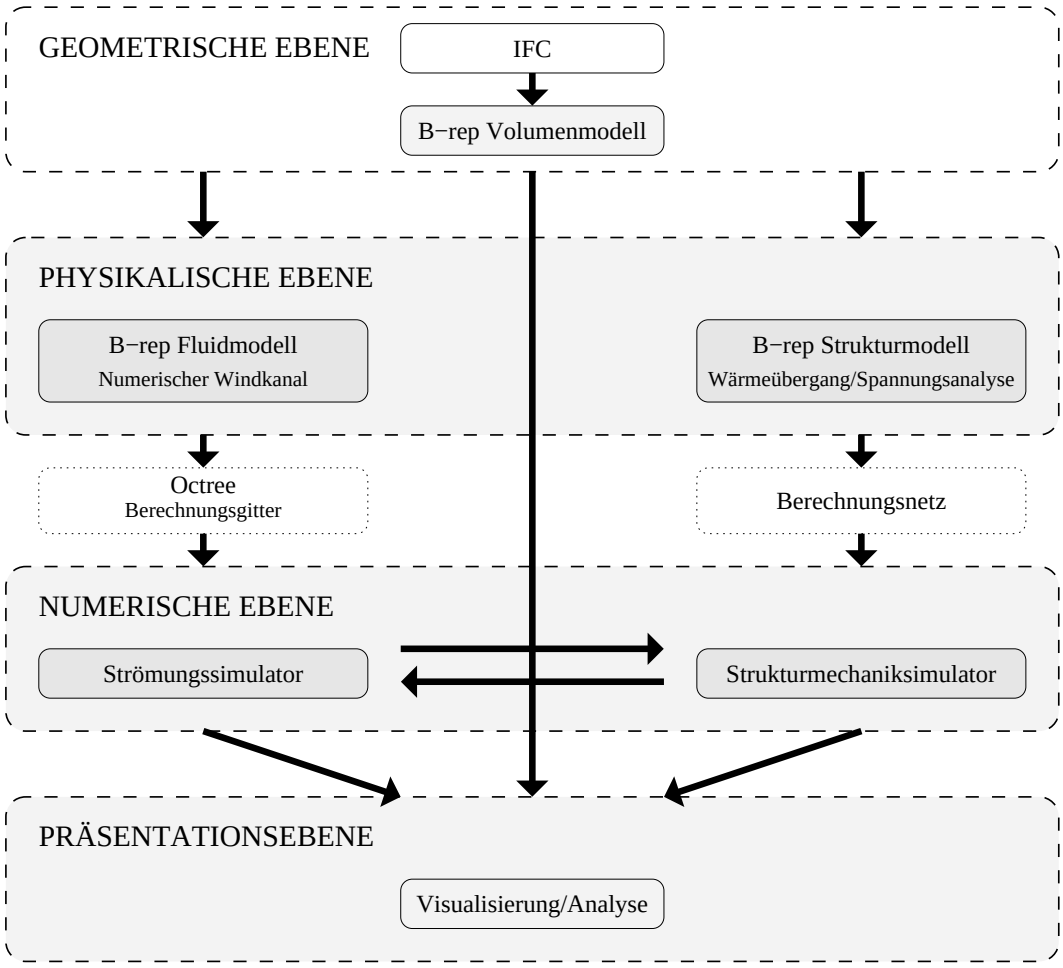

Abbildung 4.2: Modellkette

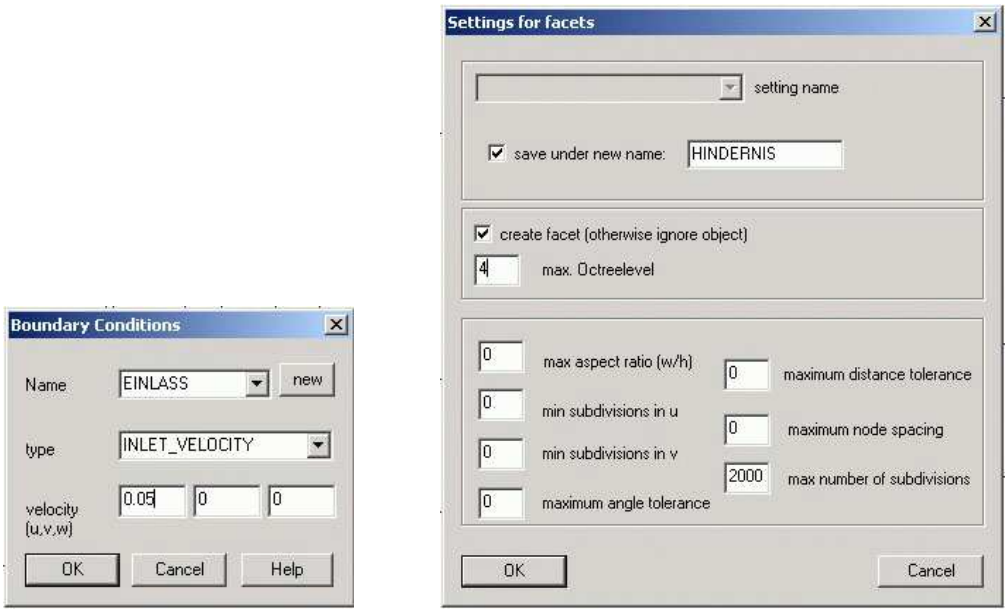

Abbildung 4.3: Dialoge für die Definition von Randbedingungen (links) und die Facettierung (rechts)

Definition von Randbedingungen, Materialparameter usw. belegt und somit immer noch unabhängig von der Simulationsmethode für die numerische Ebene vorbereitet [41]. Diese strikte Trennung der physikalischen Definitionen von den numerischen Techniken ermöglicht erst den Austausch der Werkzeuge und der geometrischen Modelle bis zu dieser Ebene. Es ist denkbar einen modifizierten Windkanal als Präprozessor für FE-Analysen zu nutzen.

Das WTM-Modul erzeugt Dateien, die in der numerischen Ebene automatisch weiter verarbeitet werden.

4.3.3 Numerische Ebene

In dieser Ebene wird das eigentliche physikalische Problem mithilfe einer numerischen Diskretisierung der mathematischen Modelle der vorhandenen Physik näherungsweise berechnet. Dieser Vorgang ist nicht nur der wichtigste Prozess der Ablauf- und Modellkette, sondern auch die existentielle Grundlage der Simulation. Meist wird hierfür der Großteil der Resourcen bezüglich Rechenleistung benötigt. Daher ist vor allem der Berechnungsmethode und entsprechend auch der Berechnungsgrundlage, also dem Gitter oder Netz, die größte Bedeutung beizumessen. Als Konsequenz wird der numerischen Ebene und den korrespondierenden geometrischen Modellen die zentrale Stellung im Integrationskonzept (vgl. Abb. 4.2) eingeräumt, sodass alle anderen Ebenen und deren Modelle sich dieser Ebene anpassen müssen.

4.3.3.1 Gittergenerierung

Zunächst muss das geometrische Modell in Abhängigkeit vom numerischen Verfahren diskretisiert werden. Da der verwendete 3D-Simulator auf äquidistanten, kartesischen Gittern rechnet, muss zunächst ein solches generiert werden. Dieser Schritt entspricht der Kopplung der verschiedenen Modelle zwischen dem CAD-System und dem Strömungssimulator und wird wegen der Thematik dieser Arbeit ausführlich in den Kapiteln 4.3.5 und 4.4 behandelt.

4.3.3.2 Berechnungskern: VirtualFluids

Um die Navier-Stokes Gleichungen (vgl. Kapitel 5.5.3) zu lösen, wird ein auf dem Lattice-Boltzmann Verfahren basierter Strömungssimulator (*VirtualFluids*) eingesetzt. Er wurde im Rahmen mehrerer Arbeiten [59, 92, 102] entwickelt und bezüglich der Rechenzeit optimiert.

Im Wesentlichen basiert die Lattice-Boltzmann Methode auf der Beschreibung des statistisch gemittelten Verhaltens aller Partikel in einem Kontrollvolumen. Das Spektrum der möglichen Partikel-Geschwindigkeiten wird auf diskrete Geschwindigkeitsvektoren reduziert. Durch Anwendung der diskreten kinetischen Theorie der molekularen Dynamik (z. B. [82]) wird die Lattice-Boltzmann Gleichung so abgeleitet, dass Erhaltungssätze der Strömungsmechanik erfüllt werden. Das Verfahren ist vollständig explizit und gibt instationäre Vorgänge wieder.

Die Grundlagen und Erweiterungen der Methode werden ausführlich in den Kapitel 5 und 6 beschrieben. An dieser Stelle sei darauf hingewiesen, dass ebenso andere gitterbasierte Simulationsverfahren den Lattice-Boltzmann Kernel substituieren könnten.

4.3.4 Präsentationsebene

Im Anschluss an die Berechnungsphase werden die erhaltenen Datensätze in geeigneter Weise ausgewertet und visualisiert. Ziel der Auswertung ist die quantitative Bestimmung geeigneter, meist algebraischer Ausdrücke (z.B. c_D-Wert, siehe auch Kapitel 7.5) oder die Verifikation der Konsistenz des numerischen Verfahrens, indem beispielsweise die Divergenzfreiheit eines inkompressiblen Strömungsfeldes geprüft wird. Die Visualisierung beschäftigt sich dagegen mit der grafischen Darstellung von Datenmengen großen Umfangs, wie sie typischerweise aus Strömungssimulationen resultieren. Neben der Möglichkeit der qualitativen Analyse der Daten steht oft das Verstehen und die Klärung physikalischer Sachverhalte im Vordergrund [50, 54, 60].

Grundsätzlich sind im Postprocessing einer numerischen Simulation von der Ergebnisdatei bis zur gewünschten Abbildung folgende Arbeitsschritte durchzuführen [18]:

- Übernahme der Simulationsergebnisse des Strömungssimulators sowie der Geometrie aus einem CAD-Modell:
  Wegen der bereits erwähnten Trennung der drei Aufgabenbereiche Pre-, Main- und Postprocessing wird dies hier über ein File-Interface realisiert.

- Reduktion der Datenmengen durch Filterung von Gitterpunkten:
  Um den Konflikt zwischen Geschwindigkeit der weiteren Verarbeitung inklusive des anschließenden Renderings und der Qualität der Ergebnisausgabe zu minimieren, wird eine adaptive Gitterausdünnung im Bereich kleiner Lösungsgradienten durchgeführt.

- Abbildung der Ergebnisse:
  Dazu gehören z. B. die numerische Integration des Geschwindigkeitsfeldes zur Generierung von Stromlinien, das Erzeugen von Isoflächen und Vektorsymbolen oder die Interpolation von Ergebnissen auf beliebigen Schnittebenen.

- Echtzeitfähige Visualisierung mit hoher Ausgabequalität:
  Die Antwortzeit des Visualisierungssystems sollte sehr klein sein, um dem Ingenieur die Möglichkeit zu geben, Datensätze schnell und möglichst interaktiv zu analysieren ohne qualitative Einbußen in Kauf nehmen zu müssen.

- Zur Nachbearbeitung zählen das Erstellen von Bildern oder Videofilmen sowie Ausgabedateien zur Weiterbearbeitung in verschiedenen Softwarepaketen oder Datenformaten.

Durch das hier vorgestellte Integrationskonzept wird zum einen der Datentransfer vom Simulationsprogramm zum Visualisierungswerkzeug und zum anderen die Reduktion der Datenmenge optimiert (vgl. Kapitel 4.3.6).

4.3.5 Transition von der physikalischen zur numerischen Ebene

Die Ableitung eines Berechnungsgitters von einem mit Randbedingungen behafteten CAD-Modell erfolgt in drei wesentlichen Schritten, wobei der Oktalbaum das zentrale Bindeglied darstellt (vgl. Abb. 4.4):

- Zunächst wird ein Facettenmodell der Oberfläche des B-rep-Modells generiert. Die zugewiesenen Randbedingungen werden gleichzeitig auf das Facettenmodell, welches somit völlig unabhängig vom CAD-Modell oder einem Produktmodell vorliegt, übertragen. Darüber hinaus reduziert sich die Komplexität der Oberflächenbeschreibung und somit die damit verbundene Rechenzeit im Vergleich zu Freiformoberflächen.

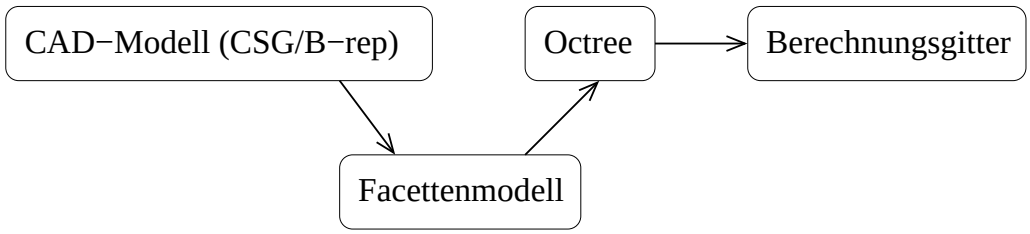

Abbildung 4.4: Modellkette: Vom CAD-Modell zum Rechengitter

- Ein Oktalbaum wird erzeugt, indem alle Oktanten, die mit dem Facettenmodell eine Schnittmenge haben, bis zu einem bestimmten Ziellevel verfeinert werden. Die Attribute *innen*, *außen* oder *Berandung* werden den Blättern des Spacetrees zugewiesen.

- Das Berechnungsgitter für das Lattice-Boltzmann Programm VirtualFluids wird in Form eines 1D-Vektors[8] übergeben. Hierfür müssen jene Oktanten, die nicht die geforderte Auflösung aufweisen, während der Erzeugung des 1D-Vektors gefüllt werden.

Diese oben genannten Punkte stellen im Wesentlichen das Konzept zur geometrischen Kopplung der Aufgabenbereiche Preprocessing und Berechnungsphase dar. In [15] wird gezeigt, dass die der Octree-Erzeugung vorgeschaltete Facettierung des Geometriemodells nicht zwingend nötig ist. Es werden aber gleichzeitig Schwächen angedeutet, welche durch das Facettenmodell reduziert bzw. gelöst werden:

- Durch die Entkopplung entsteht die Möglichkeit, Facettenmodelle aus anderen Applikationen zu übernehmen und weiter zu verarbeiten.

- AutoCAD wird derzeit nur für Windows Plattformen entwickelt und vertrieben. Für die Generierung sehr großer Gitter kann es erforderlich werden, Hochleistungsrechner einzusetzen, die überlicherweise andere, meist Unix-basierte Betriebssysteme verwenden.

- Bei Verwendung eines Facettenmodells zur Repräsentierung der Geometrie, wird die Operation $3D - Objekt \bigcap Oktant$ durch mehrere Operationen $Dreieck \bigcap Oktant$ ersetzt. Neben der effizienteren Implementierung sind Geschwindigkeitsvorteile bei der Octree-Generierung zu erwarten.

Die Klassenstruktur, die Algorithmen und die Implementierungsdetails werden in Kapitel 4.4 ausführlich beschrieben. Weitere Ansätze zur Netzgenerierung für Finite-Elemente Diskretisierungen können z. B. in [36, 57, 66, 108, 109] nachgelesen werden.

[8] Da die Raumdimensionen des 3D-Feldes bekannt sind, ist VirtualFluids in der Lage, jede Position im 1D-Vektor im Raum eindeutig zuzuordnen.

4.3.6 Transition von der numerischen Ebene zur Präsentationsebene

4.3.6.1 Anbindung der Visualisierung

Mithilfe der Baumdatenstrukturen lässt sich die Visualisierung sehr gut in das Gesamtkonzept integrieren [8]. Die vom Berechnungskernel erzeugte Ergebnisdatei enthält alle Blattzellen und die entsprechenden Ergebniswerte eines noch nicht vorhandenen Oktalbaumes. Der Aufbau des Spacetrees wird nun von den Blattzellen selbst gesteuert, indem ausgehend von einer a priori bestimmten Wurzelzelle rekursiv Zellen verfeinert werden, solange die Blattzelle mit ihr eine Schnittmenge aufweist (vgl. Kapitel 3.3.4)[52]. Der zeitaufwendigste Schritt, die Berechnung der Schnittmenge, kann hier entfallen und man erhält in diesem Fall einen *vollständigen* Spacetree, d. h. alle Blätter haben die gleiche, maximale Baumtiefe.

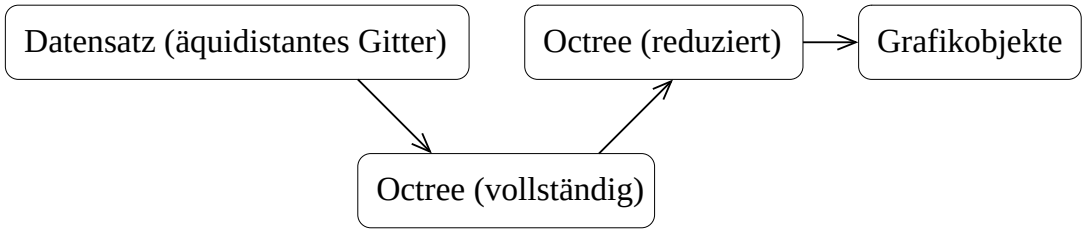

Abbildung 4.5: Modellkette: Vom Berechungsgitter zur Visualisierung

4.3.6.2 Datenreduktion

Die Datenreduktion entspricht dem inversen Vorgang einer Gitterverfeinerung (vgl. Kapitel 8). Ein Vergröberungskriterium, beispielsweise die Begrenzung des Gradienten des Druckfeldes, wird auf alle Blätter, die ein gemeinsames Vorgängerelement haben, angewendet. Nur wenn alle zusammengehörigen Blätter für die Vergröberung freigegeben werden, darf die Löschung der entsprechenden Zellen erfolgen (vgl. Abb. 4.6):
In Abbildung 4.7 ist das reduzierte Gitter einer 2D-Simulation der von Karmanstraße abgebildet. Bei dieser Datenreduktion wurde die Anzahl der Gitterpunkte ausgehend von einem uniformen Gitter von 41409 auf 3394 ($\approx 8.2\%$ der ursprünglichen Gitterpunkte) bei nahezu gleichbleibender Qualität verringert.

4.3.6.3 Datenvisualisierung

Mit dem reduzierten Datensatz können die Daten nun effizienter dargestellt werden. In Abbildung 4.8 wird das Geschwindigkeitsfeld mithilfe von Stromlinien visualisiert.

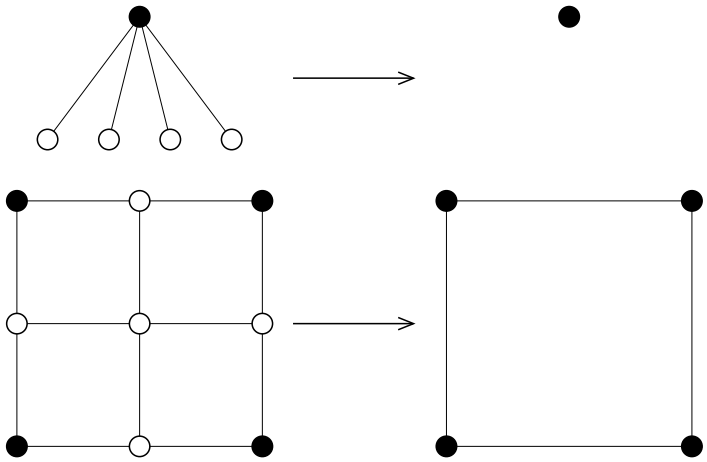

Abbildung 4.6: Zellvergröberung

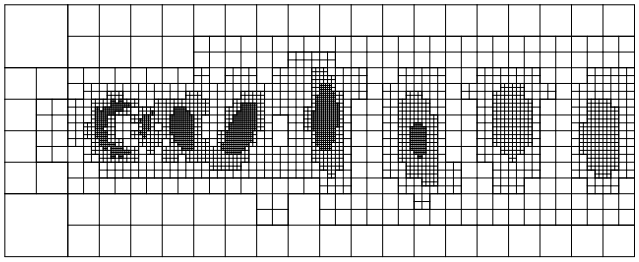

Abbildung 4.7: Beispiel: von Karmannstraße bei Re=100

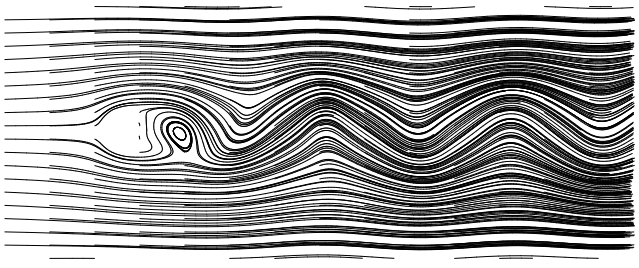

Abbildung 4.8: Visualisierung: Geschwindigkeitsfeld mit Stromlinien

Die weiter oben vorgestellte Modellkette wurde in [38] und [55] entwickelt und aus Gründen der Vollständigkeit des Konzepts an dieser Stelle kurz vorgestellt. Eine ausführlichere Darstellung ist in [51] nachzulesen.

4.4 Realisierung: Transition von der physikalischen Ebene zur numerischen Ebene

4.4.1 Facettenmodell

4.4.1.1 Klassenkonzept: Facettenmodell

Die in Abbildung 4.9 dargestellte Klasse *TriFaceSet* dient der Datenhaltung der Facettenmodelle. Für die Facetten kommen nur Dreiecke in Frage, deren Nummer in der Elementliste und deren Koordinaten aufeinanderfolgend in der Knotenliste gespeichert werden. Bestünde ein Facettenmodell aus nur einem Dreieck, würden in der Knotenliste die Koordinaten der drei Punkte A,B und C in der Form

$$Knotenliste = \{A_x, A_y, A_z, B_x, B_y, B_z, C_x, C_y, C_z\}$$

stehen. Alle physikalischen und geometrischen Parameter werden in einem Objekt der Klasse *MeshValues* gehalten. Bei den Attributen *WTM_Boundary*, *WTM_Material* handelt es sich um Zeichenketten, anhand derer die Randbedingungen und Materialkennwerte aus einer Datenbank gelesen werden können [40]. Das Attribut *octrLvl* gibt die vorgegebene Baumtiefe an. Die restlichen Attribute steuern die Facettierung. Mit *distTol* wird beispielsweise der maximale Abstand zwischen dem Dreiecksnetz und der ursprünglichen Geometrie begrenzt (vgl. Abb. 4.3).

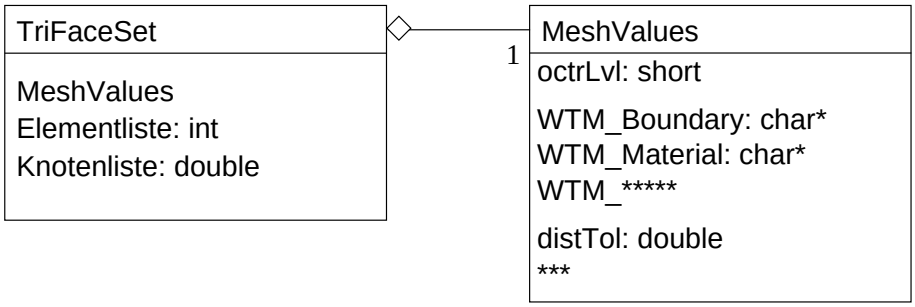

Abbildung 4.9: Facettenmodell: Klassen mit Attributen

4.4.1.2 Implementierung: Facettenmodell

Zur Darstellung der konstruierten Objekte hält AutoCAD neben dem CSG-Baum und Extrusionsmodellen auch ein B-rep-Modell, auf das mit der Klassen-

bibliothek *ObjectARX* zugegriffen werden kann, vor. Für verschiedene Render-Funktionen benötigt AutoCAD selbst eine Facettierung der Oberflächenmodelle. Die ObjectARX stellt hierzu spezielle Netzobjekte zur Verfügung, um automatisch orientierte Facettenmodelle extrahieren zu können. Die Orientierung der Facetten ist durch den Normalenvektor gegeben. Das Facettenmodell wird über eine Datei-Schnittstelle an den Octree-Generator übergeben [47].

4.4.2 Oktalbaum

4.4.2.1 Klassenkonzept: Oktalbaum

Die Klasse *Octree* ist im Wesentlichen ein Container für alle den Octree betreffenden Attribute mit Methoden zu dessen Generierung. Ein Octree-Objekt enthält ein oder mehrere[9] Wurzel-Elemente, Listen für die Blätter, Listen für die Gitterknoten und weitere Attribute. Der Bezug zu den Facettenmodellen wird schon bei der Erzeugung der Oktanten hergestellt. Während der Berechnung der

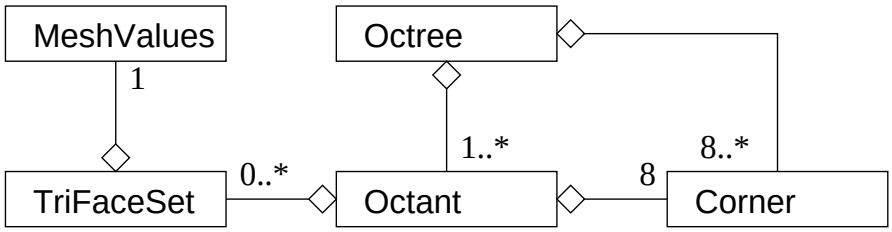

Abbildung 4.10: Oktalbaumgenerator: Klassenbaum

Schnittmenge ($Dreieck \bigcap Oktant$) werden die Identifizierungsnummern (ID) der Facettenmodelle in den Oktanten selbst gespeichert. Auf diese Art und Weise hat jeder Oktant über das Facettenmodell mit der höchsten Priorität[10] einen Bezug zu einem geometrischen Objekt, welches wiederum mit physikalischen Randbedingungen behaftet ist. Diese Information kann dann von den Oktanten an dessen Gitterknoten weiter vererbt werden [47].

4.4.2.2 Implementierung: Oktalbaum

Die Octree-Datenstruktur wurde unter Verwendung des in Kapitel 4.4.2.1 dargestellten Konzepts mit der Programmiersprache C++ implementiert. Die Implemenierung wurde bereits in [15] und in [47] ausführlich beschrieben und wird daher nur zusammenfassend dargestellt.

[9] Um Gebiete, die nicht Würfel-förmig sind, besser diskretisieren zu können, werden mehrere Wurzel-Elemente zugelassen.

[10] Oktanten, die von mehreren Facettenmodellen geschnitten werden, erhalten die ID jenes Facettenmodells, das in seiner Priorität höher eingestuft wurde.

Im ersten Schritt findet die sogenannte *geometrische Diskretisierung* statt, in der ein Octree in Form eines ungerichteten, zyklenfreien Graphen aufgebaut wird. Die Baumtiefe in Bereichen der Berandung der geometrischen Objekte wird vom Nutzer vorgegeben (vgl. Kapitel 3.3.4). Zu diesem Zeitpunkt ist nur der Rand der Hindernisobjekte diskretisiert. Die Zuordnung der Zellen (Oktanten) zu den geometrischen Objekten des physikalischen Modells erfolgt über die IDs der Facettenmodelle. Generell wissen die Zellen aber noch nicht, ob sie innerhalb oder außerhalb einer Struktur liegen.

Als nächstes wird der Octree *geglättet* und *vernetzt*. Im Anschluss daran wird jede Zelle eindeutig bezüglich der Lage relativ zu den Hindernisobjekten zugeordnet. Diese Schritte weichen in ihrer Implementierung von anderen Vorgehensweisen [78, 86, 87, 107], welche meist auf die Glättung oder die Vernetzung oder beides verzichten, ab und werden daher ausführlich in Kapitel 4.4.3 beschrieben.

Im letzten Schritt werden die Attribute (ID der Facettenmodelle), mit denen die Randbedingungen korrelieren, an die Gitterknoten weitergegeben. Hierbei muss auf die Priorität der geometrischen Objekten geachtet werden, da in Bereichen, in denen sich diese überlappen oder sehr nahe zusammenliegen eine eindeutige Zuordnung oft nicht möglich ist.

4.4.3 Modifikationen der Oktalbaum-Datenstruktur

Um die Effizienz diverser Berechnungsalgorithmen sowie die Möglichkeiten der Octrees auszubauen, ist eine Erweiterung der Zellen der Baumdatenstruktur, vor allem der Blätter, um einige Attribute notwendig. Zu diesen Attributen zählen neben den Ergebniswerten selbst vor allem die Zeiger auf Nachbarzellen.

4.4.3.1 Glättung und Vernetzung

Die Suche benachbarter Zellen zwingt jeden spacetreebasierten Algorithmus, der nur auf eine direktionale Verzeigerung zugreifen kann, zu ständigen, unter Umständen zeitaufwendigen Baumtraversierungen. Für den Fall, dass die Nachbarschaftsbeziehungen öfter benötigt werden, wird die bereits vorhandene bidirektionale, hierarchische Verzeigerung der Zellen um eine flache Verzeigerung zwischen den Blatt-Zellen erweitert [15]. Mithilfe der somit generierten *hybriden Netz-Baumdatenstruktur* sind alle folgenden Algorithmen in der Lage, Zellen direkt und sehr schnell auf deren Nachbarn zugreifen zu lassen. Dies führt zu einer Datenstruktur, die sowohl die Eigenschaften eines hierarchischen Gitters, als auch die Eigenschaften eines Oberflächen-Netzes, um beispielsweise Informationen schnell zu propagieren (vgl. Kapitel 4.4.3.2), besitzt. Zur Reduktion der Anzahl der möglichen Nachbarschaftsverhältnisse zwischen den Zellen und somit zur Reduktion des Aufwandes der Implementierung wird vorab eine Glättung des Spacetrees vorgeschlagen, sodass sich orthogonale Nachbarn um maximal eine Baumtiefe unterscheiden.

Zunächst wird ein Algorithmus vorgestellt, mit dem Nachbarn durch gezielte Baumtraversierung gefunden werden können. Hierzu wird folgende Bit-Codierung der Position von Elementen innerhalb ihres Eltern-Elementes angegeben (vgl. Abb. 4.11):

$$Position = \{x_{Bit}, y_{Bit}\}$$

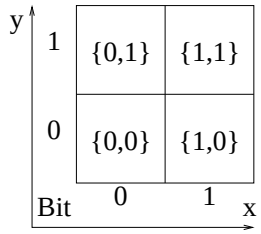

Abbildung 4.11: Bit-Codierung der Position von Elementen

Ein Pfad wird als Folge von Positionsangaben definiert:

$$Pfad = \left\{ \{x_{Bit}, y_{Bit}\}_{Level} \right\}$$

Die Baumtraversierung, die Glättung und die Vernetzung ist in den Abbildungen 4.12 und 4.13 skizziert. Ausgehend von der markierten Zelle in Bild 4.12 (linkes Gitter) soll der Nachbar in positiver x-Richtung gefunden werden. Die gesuchte Zelle liegt nicht innerhalb des Elternelementes der suchenden Zelle. Der Pfad ausgehend von der suchenden Zelle ist $\{1, 1\}_3$. Der Baum wird jetzt in den Schritten 1 und 2 in Richtung der Wurzel traversiert, wobei stets das x-Bit der gröbsten Ebene betrachtet wird. Der Pfad lautet nun (Bild 4.12, rechtes Gitter):

$$Pfad = \left\{ \{\mathbf{1}, 1\}_3, \{\mathbf{1}, 0\}_2, \{\mathbf{0}, 1\}_1 \right\}$$

Da das x-Bit in der Ebene 1 von 1 auf 0 gewechselt hat, ist somit bekannt, dass das Elternelement der Position $\{0, 1\}_1$ Wurzel (Bild 4.13, markierte Zelle des linken Gitters) eines Unterbaumes ist, der sowohl die gesuchte als auch die suchende Zelle enthält. Der Pfad wird nun bezüglich der Bit-Folge in x-Richtung invertiert und man erhält den neuen Pfad (von rechts nach links gelesen), der den Weg im Baum zum Nachbarelement anzeigt:

$$Pfad^{inv} = \left\{ \{\mathbf{0}, 1\}_3, \{\mathbf{0}, 0\}_2, \{\mathbf{1}, 1\}_1 \right\}$$

In den Schritten 3 und 4 wird dieser inverse Pfad beginnend bei der neuen Wurzel in Richtung der zunehmenden Baumtiefe traversiert, bis ein Blatt erreicht wird.

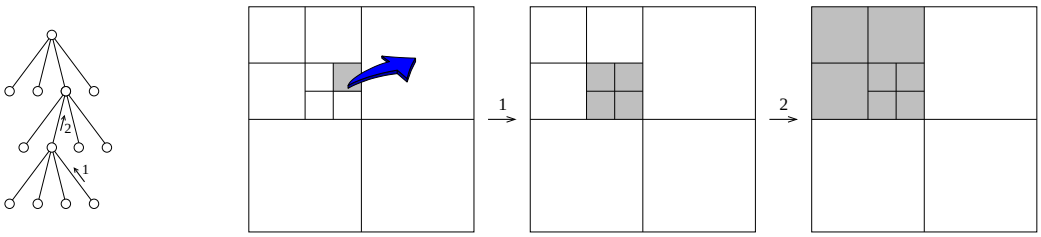

Abbildung 4.12: Glättung und Vernetzung - Teil 1

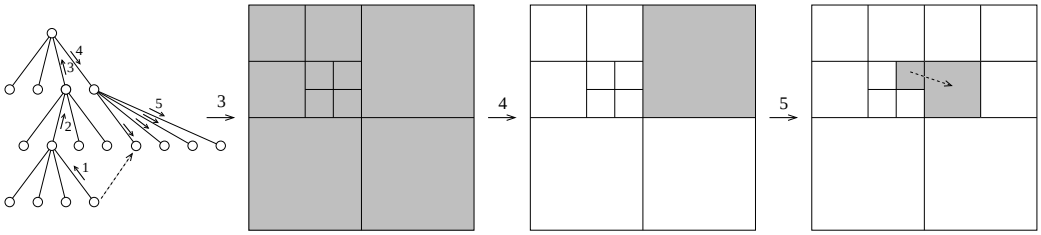

Abbildung 4.13: Glättung und Vernetzung - Teil 2

Wenn sich die Baumtiefe des gefundenen Nachbarn um mehr als eins unterscheidet (Bild 4.13, mittleres Gitter) wird die Zelle verfeinert und der Pfad weiter verfolgt, bis das Kriterium der Glättung erfüllt ist. Anschließend wird ein Zeiger auf das gefundene Element gesetzt, womit der Spacetree vernetzt wird (Bild 4.13, Pfeil, rechtes Gitter).

4.4.3.2 Überflutung

Wie bereits erwähnt, erlaubt die hybride Netz-Baumdatenstruktur einen schnellen Informationsaustausch in horizontaler Ebene. Dieser Vorteil kann innerhalb der Octree-Erzeugung bei der Zuordnung der Lage der Blätter genutzt werden. Anstatt mit jedem Blatt einen zeitintensiven Vergleich mit dem Facettenmodell zu berechnen, wird für jedes geometrische Objekt ausgehend von dem jeweiligen Teil-Facettenmodell ein inneres Blatt gesucht. Es wird entsprechend markiert und im nächsten Schritt erhalten alle orthogonalen Nachbarn mithilfe der schnellen Zugriffsmöglichkeit durch die Vernetzung ebenfalls das Attribut *innen*. Dieser Schritt wird solange rekursiv wiederholt (in Wellen) bis die Information '*Blatt ist innen*' alle inneren Blätter der Hindernis-Objekte markiert hat (vgl. Abb. 4.14). Die Überflutung wird lokal abgebrochen, wenn ein Blatt erreicht wird, das als Berandung markiert wurde. Somit können mit nur einer Berechnung der Lage eines Blattes, alle anderen Blätter des gleichen Objekts zugewiesen werden [15].

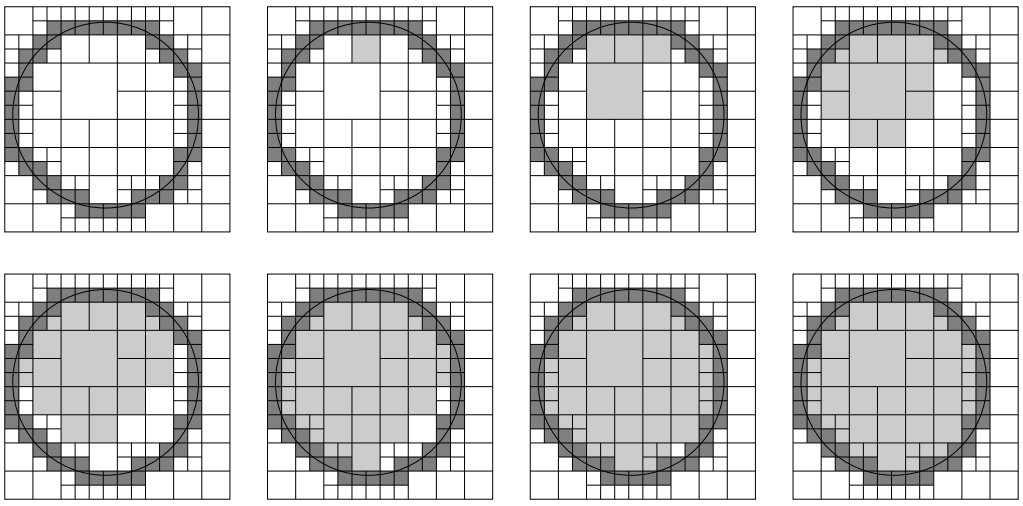

Abbildung 4.14: Informationsausbreitung - Überflutungsalgorithmus

4.4.3.3 Graphentheoretische Veränderungen

Bei dieser neuen spacetreebasierten Datenstruktur handelt es sich nach wie vor um einen Graphen. Er ist durch die Vermischung mit den Eigenschaften von Oberflächennetzen allerdings nicht mehr zyklenfrei. Streng genommen handelt es sich also nicht mehr um eine Baumdatenstruktur. Dennoch wird weiterhin von Bäumen oder Spacetrees gesprochen, da weiterhin deren Vorteile der räumlichen Organisation und Hierarchie verwendet werden können.

4.4.4 Berechnungsgitter

Nach der Octree-Generierung kann man ein Berechnungsgitter ohne weiteren Informationsverlust erzeugen. Da der vorhandene Simulationskernel die Geometrie-Informationen in Form eines 1D-Vektors benötigt, muss ausgehend von der vorhandenen Octree-Struktur ein Voxelmodell (Abbildung 4.15, gestricheltes Gitter) erzeugt werden. In dem neuen Modell soll es nur noch ganzzahlige Koordinaten geben, so dass in der folgenden Operation alle Koordinaten so transformiert und skalieriert werden, dass die minimalen Koordinaten mit dem Ursprung übereinstimmen. In einer Schleife über alle Blätter des Baumes werden die Geometrie-Informationen (innen, aussen, Berandung=innen) auf das Voxelmodell übertragen. Für Blätter, die nicht die maximale Auflösung aufweisen, werden alle Zwischenzellen (siehe Abbildung 4.15, z. B. grauer Bereich) mit den gleichen Attributen belegt, wie diese im überlagerten Blatt des Baumes vorliegen. Die nachträgliche Uniformisierung des Gitters ist notwendig, da der vorhandene Simulationskern nur mit äquidistanten Gittern rechnen kann. Als letztes wird diese Voxelmatrix als binärer 1D-Vektor in eine Datei geschrieben, die als Eingabedatei

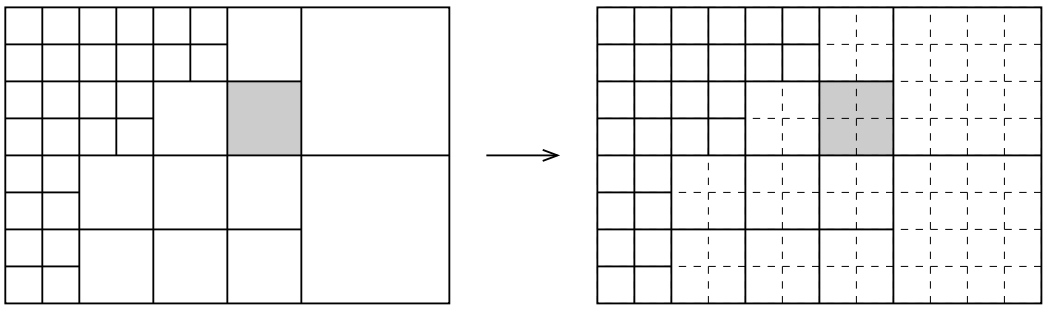

Abbildung 4.15: Füllen der Voxelmatrix

der Geometrie für den Lattice-Boltzmann Simulator *VirtualFluids* dient.

4.5 Einsatz: Simulation von In- und Umluftströmungen

Im Folgenden wird das entwickelte und implementierte Rahmenwerk, welches auf dem in den vorherigen Kapitel beschriebenen Intergrationskonzept basiert, anhand zweier Strömungen, einer Innenraumströmung eines Büroraumes und einer Außenumströmung eines Kühlturms, validiert.

4.5.1 Ähnlichkeitsanforderungen

Für den Transfer der Simulationsergebnisse auf den realen Zustand oder für die Modellierung des CFD-Modells sind für Einphasenströmungen ohne Wärmetransport im Wesentlichen zwei Ähnlichkeitsanforderungen einzuhalten:

Geometrisches Ähnlichkeitsmodell: Um eine geometrisch ähnliche Umströmung von Hindernissen zu gewährleisten, muss das geometrische Modell maßstabsgetreu entsprechend den realen, geometrischen Verhältnissen modelliert werden.

Dynamisches Ähnlichkeitsmodell: Die dynamische Ähnlichkeit wird dann erreicht, wenn das Verhältnis zwischen den Trägheits- und viskosen Reibungskräften der Strömung in der Natur und in der Simulation übereinstimmen. Dieses Verhältnis wird mit einer dimensionslosen Kennzahl, der sogenannten *Reynolds-Zahl* ausgedrückt:

$$Re = \frac{UL}{\nu} \tag{4.1}$$

L stellt eine charakteristische Vergleichslänge dar, die prinzipiell frei gewählt werden darf, aber im realen und geometrischen Modell einander entsprech-

en. U ist typischerweise die Anströmgeschwindigkeit und ν die kinematische Viskosität.

Froude-Modell: Im Gegensatz zum Ähnlichkeitsmodell berücksichtigt dieses Modell Trägheits- und Gravitationskräfte:

$$Fr = \frac{U^2}{gl} \tag{4.2}$$

Es wird gefordert, dass die Froude-Zahl für das Modell und den realen Versuch gleich sind. Dieses Modell findet insbesondere bei aeroelastischen Bauwerken, bei denen windinduzierte Schwingungen von der Gravitation abhängen, Anwendung [40].

4.5.2 Setup einer Strömungssimulation

Zu den Vorbereitungen einer Strömungsberechnung gehört neben dem Importieren der Geometriedaten in Form des Voxelmodells die Festlegung von Strömungsparametern und die Definition der gewünschten Auswertungen.

Die wichtigsten Parameter der Strömungssimulation werden nun kurz angesprochen:

Gittermodell: Mit dem Gittermodell wird der Typ des Rechengitters, welches die numerische Diskretisierung festlegt, gewählt. Für dreidimensionale Simulationen kommen das D3Q15- oder das D3Q19-Modell in Frage [80].

Kinematische Viskosität: Die kinematische Viskosität ν entspricht der Zähigkeit und sollte aus Konsistenz- und Stabilitätsgründen im Intervall $[10^{-4}\frac{\Delta x^2}{\Delta t}, \frac{1}{6}\frac{\Delta x^2}{\Delta t}]$ liegen[11] (vgl. Kapitel 5.10).

Abbruchkriterium: Das Abbruchkriterium ist meist ein vorgegebenes Änderungskriterium (z. B. Betrag der Geschwindigkeit), das allerdings nur bei stationären Strömungen erreicht werden kann.

Turbulenzmodell: In VirtualFluids ist ein *Large-Eddy* Turbulenzmodell (LES) implementiert [61]. Zwei Parameter werden hierfür eingestellt:

- Die *Smagorinsky-Konstante* wird standardmäßig zu 0.16 gesetzt.
- Nachdem sich das turbulente Strömungsfeld vollständig entwickelt hat, ist oft ein gemitteltes Verhalten der Strömung von Interesse. Der Zeitpunkt, ab dem die Mittelung beginnen soll, muss hierfür angegeben werden.

[11] Genaue Grenzwerte existieren nicht.

Zeitschritte: Als weiteres Abbruchkriterium dient die Simulationsdauer in Form der maximalen Anzahl der Zeitschritte. Weiterhin können Zeitintervalle zur Steuerung der Postprocessing-Arbeiten (z. B. AVS/Express Ausgabedatei) angegeben werden.

4.5.3 Innenraumströmung: Einzelbüro

Im ersten Beispiel wird die Ablaufkette anhand einer Innenraumströmung eines Einzelbüros validiert. Eine mögliche Fragestellung könnte darin bestehen, den Arbeitsplatz im Raum zu optimieren, sodass eine dort beschäftigte Person möglichst geringer Zugluft ausgesetzt ist.
Geometrie:

- Innenraum: Länge = 4,60 m; Breite = 3,45 m; Höhe = 2,30 m
- Fenster: Höhe = 1,40 m; Breite = 1,20 m
- Tür: Höhe = 1,80 m; Breite = 0,90 m

Randbedingungen:

- Allen Wänden und Hindernissen (Büromöbel) wird die Haftbedingung zugeordnet.
- Einlass: Das Fenster wurde mit einer konstanten Einströmgeschwindigkeit belegt.
- Auslass: An der Tür wurde ein konstanter Referenzdruck vorgegeben.

Das Strömungsfeld ist durch die Geometrie und die Randbedingungen vollständig definiert. Die Reynoldszahl wird bezüglich der Raumhöhe $h = 2,30m$ und der Einlassgeschwindigkeit $u_0 = 0,113\frac{m}{s}$ mit der kinematischen Viskosität für Luft $\nu = 15 \cdot 10^{-6}\frac{m^2}{s}$ folgendermaßen berechnet:

$$Re = \frac{u_0 \cdot h}{\nu} \approx 17333$$

Zunächst wird analog der in Kapitel 4.3.5 beschriebenen Vorgehensweise ein Facettenmodell von einem mit physikalischen Randbedingungen belegten CAD-Modell abgeleitet. In der Abbildung 4.16 sind die Bodenplatte, die Tür, das Fenster und die Büromöbel dargestellt. Die Seitenwände und die Decken wurden aus Gründen der Übersichtlichkeit weggelassen. Das Facettenmodell ist die Ausgangsgeometrie der Octree-Generierung, in der ein Oktalbaum völlig automatisch erzeugt wird und dessen Blätter die Randbedingungen erben. In Abbildung 4.17 ist das Facettenmodell mit dem entstandenen Octree überlagert visualisiert. Der Octree besteht aus 24 Wurzeloktanten, 16481 Blättern und 21839 Eckknoten.

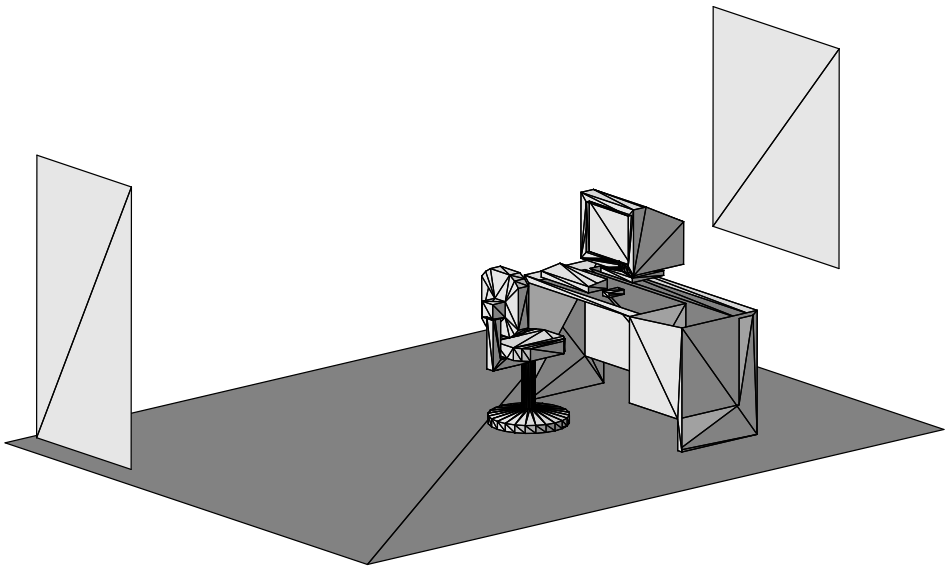

Abbildung 4.16: Facettenmodell des Einzelbüros

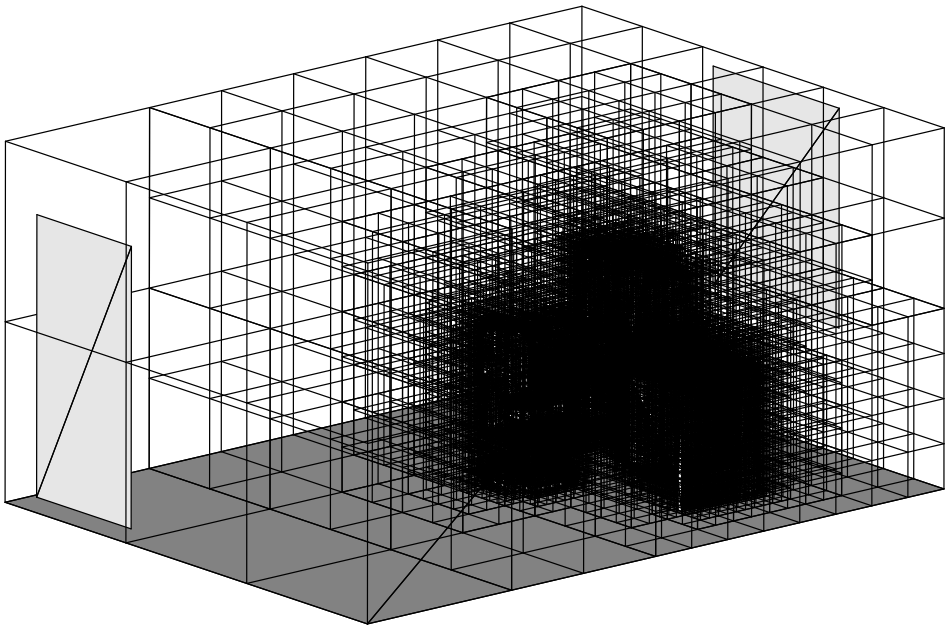

Abbildung 4.17: Oktalbaum des Einzelbüros

Anschließend wird vom Oktalbaum ein uniformes Berechnungsgitter in Form einer Voxelmatrix abgeleitet und eine entsprechende Eingabedatei für den Strömungssimulator erzeugt. Das kartesische Gitter für die Simulation besteht aus $129 \times 97 \times 65 = 813345$ Berechnungspunkten (vgl. Abb. 4.18). Dies entspricht einer Auflösung von 28,26 Gitterpunkten pro Meter oder von 3,59 cm pro Gitterpunktabstand. Auf einem PC (Notebook, Intel Pentium II, 400 MHz) benötigte

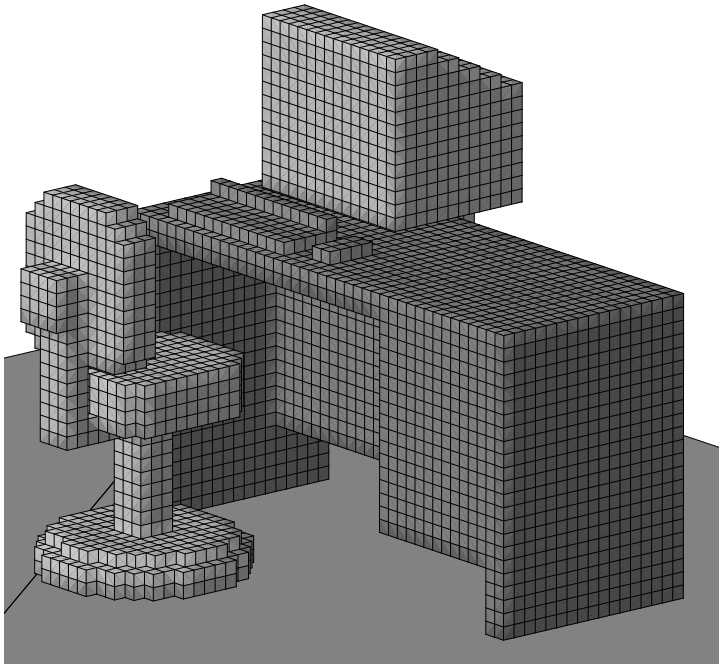

Abbildung 4.18: Büromöbel in Normzellen-Darstellung

der Vorgang beginnend mit der Octree-Generierung bis hin zum Schreiben der Datei mit der Voxelmatrix und einer UCD-Datei zur Visualisierung des Oktalbaumes mit AVS/Express 51 Sekunden.

Um eine Reynoldszahl von 17333 in der Lattice-Boltzmann Simulation, bei einer Einlassgeschwindigkeit von $0,008\frac{\Delta x}{\Delta t}$ und 65 Gitterpunkten für die Höhe des Raumes zu erreichen, muss eine Viskosität von $0,00003\frac{\Delta x^2}{\Delta t}$ gewählt werden. Bei dieser geringen Viskosität muss das Momentenmodell der Lattice-Boltzmann Methode in Kombination mit einer Large-Eddy Turbulenzmodellierung (Smagorinsky-Konstante = 0,16) aktiviert werden. Die Simulation wurde auf einem gewöhnlichen PC[12] berechnet. Um ein quasi-stationäres Strömungsbild zu erhalten, wurden circa 100000 Zeitschritte in 60 Stunden gerechnet. Die gewonnenen Datensätze wurden mit den in [38, 55] beschriebenen Techniken reduziert und anschließend mit AVS/Express visualisiert:

[12]CPU: Pentium IV, 1700 MHz

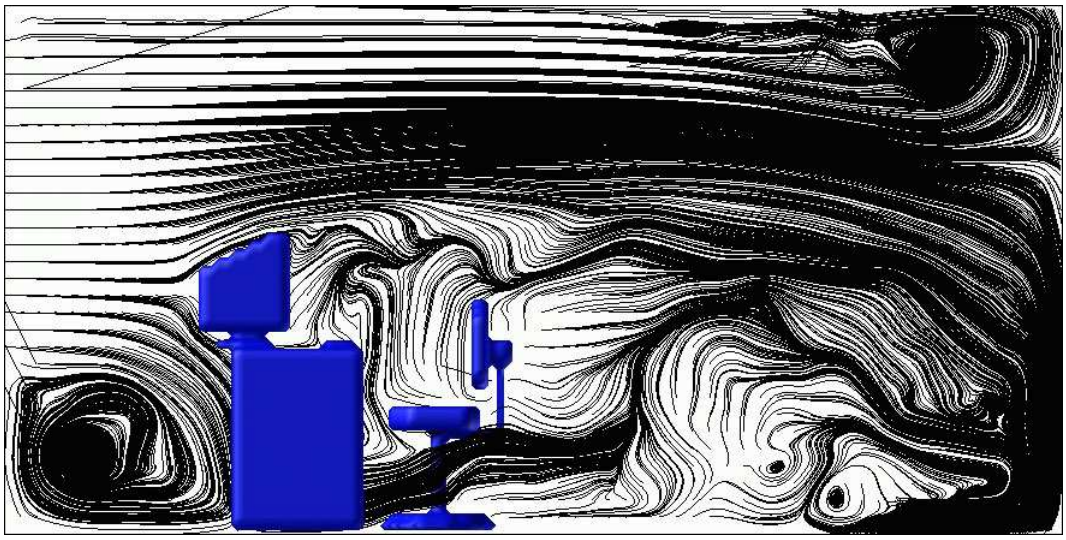

Abbildung 4.19: Stromlinien (orthogonale Projektion) - Momentaufnahme bei t=170000

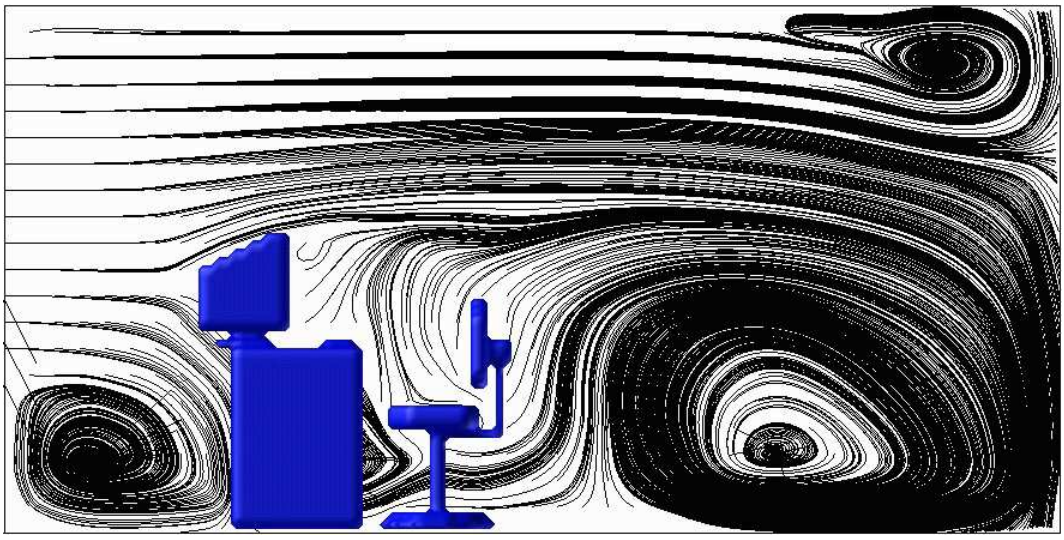

Abbildung 4.20: Stromlinien (orthogonale Projektion) - gemitteltes Geschwindigkeitsfeld

4.5.4 Kühlturm-Umströmung

Im zweiten Beispiel wird das Integrationskonzept am Beispiel einer Kühlturm-Umströmung zur Windlast-Bestimmung demonstriert. In der folgenden Abbildung sind die vier Modelle, das CAD-Modell, das Facettenmodell, der Octree

und das Voxelmodell dargestellt:

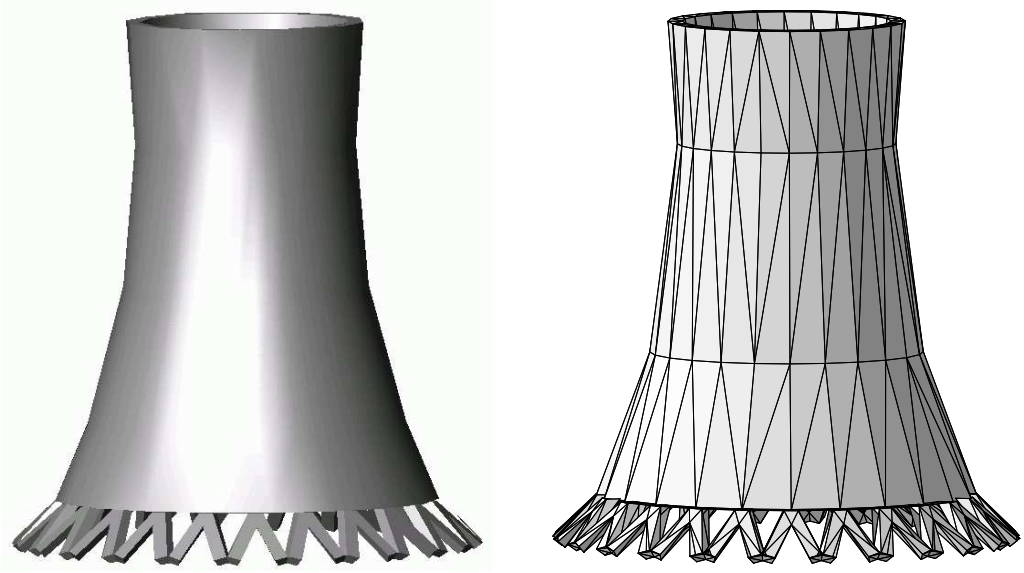

Abbildung 4.21: CAD- und Facettenmodell des Kühlturms

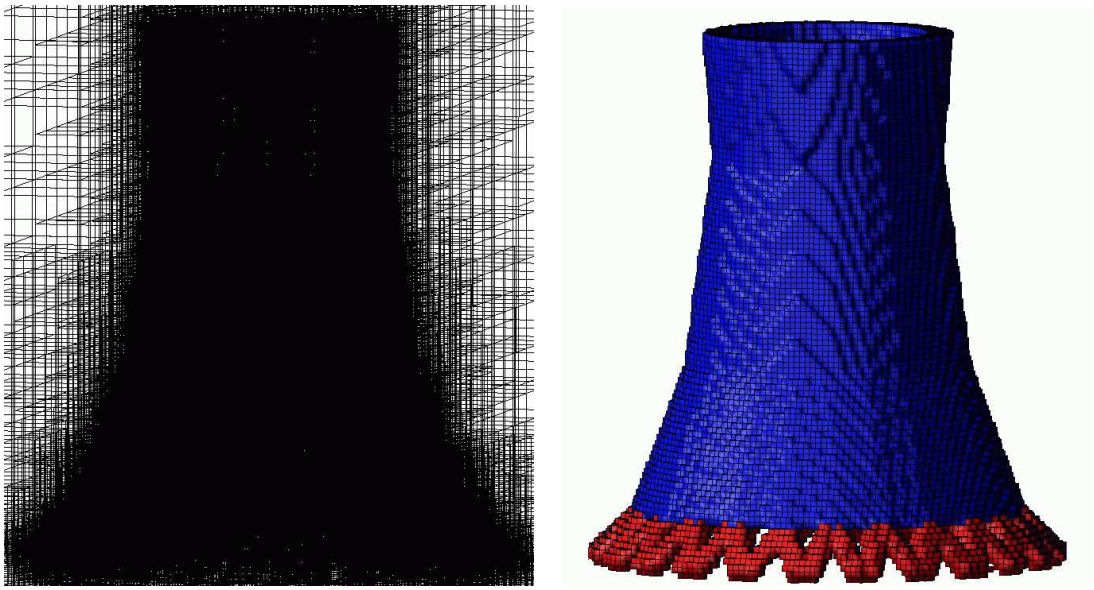

Abbildung 4.22: Octree- und Voxelmodell des Kühlturms

Bei einer Anströmgeschwindigkeit von $u_0 = 20\frac{m}{s}$, einer Bauwerkshöhe von $h = 165m$ und der kinematischen Viskosität von Luft $\nu = 15 \cdot 10^{-6}\frac{m^2}{s}$ ergibt

sich eine Reynoldszahl von $Re = 2.2 \cdot 10^8$. In [105] wird die benötigte Anzahl der Zeitschritte mit

$$\left(\frac{T}{\Delta t}\right) \approx Re^{\frac{1}{2}} \quad \Rightarrow \quad T \approx Re^{\frac{1}{2}} \cdot \Delta t \tag{4.3}$$

und die benötigte Anzahl der Gitterpunkte mit

$$\frac{L}{\Delta x} \approx Re^{\frac{3}{4}} \quad \Rightarrow \quad L^3 \approx Re^{\frac{9}{4}} \cdot \Delta x^3 \tag{4.4}$$

abgeschätzt. Insgesamt werden circa

$$T \cdot L^3 \approx Re^{\frac{11}{4}} = (2.2 \cdot 10^8)^{\frac{11}{4}} = 8.7 \cdot 10^{22} \tag{4.5}$$

Knotenupdates für die Berechnung einer makroskopischen Bewegung (z. B. eine Wirbelablösung) notwendig. Auf dem Höchstleistungsrechner SR-8000 des LRZ[13] in München kann eine Knotenupdaterate von $20 \cdot 10^6 \frac{Knotenupdates}{sec}$ angenommen werden. Die Berechnungsdauer liegt somit in einer Größenordnung von etwa $4.4 \cdot 10^{15}$ Sekunden ($\approx 1.38 \cdot 10^8$ Jahre). Die Abschätzung bezieht sich auf eine *direkte numerische Simulation* (DNS). Durch den Einsatz geeigneter Turbulenzmodelle können diese Berechnungszeiten reduziert werden, wobei durch die dafür notwendige, weitere Modellbildung (Turbulenzmodellierung) wiederum Genauigkeitsverluste in Kauf genommen werden müssen.

4.6 Grenzen des Verfahrens

Das Integrationskonzept hat beiden Anwendungsfällen standgehalten und Eingangsdaten im Rahmen des erwünschten semi-automatischen Preprocessings erzeugt.

Die Innenraumsimulation lief insgesamt circa 114 Stunden[14] ($\approx$ 5 Tage) für 200000 Zeitschritte auf einem PC[15]. Damit ist aufgrund der Simulationsdauer die Grenze der Einsatzfähigkeit für Ingenieure bereits erreicht. Gleichzeitig wurden sämtliche Simulationsparameter ausgeschöpft, um eine möglichst große Reynoldszahl zu erreichen. Da sich die Reynolds-Zahl in einem gerade noch akzeptablen Bereich für diese Problemklasse befindet, ist hier die Möglichkeit der Interpretation der Ergebnisse zumindest denkbar. Eine Simulation auf einem Hochleistungsrechner oder auf einem Rechencluster würde sicherlich zu deutlich verbesserten und genaueren Ergenissen führen, wenn damit eine Verfeinerung der Diskretisierung einhergehen würde.

[13] www.lrz-muenchen.de

[14] In dieser Simulation waren circa 60 Stunden Berechnungszeit notwendig, um ausgehend von den gewählten Anfangsbedingungen ein quasi-stationäres Strömungsfeld zu erhalten. Typischerweise entstehen zu Beginn einer Lattice-Boltzmann Simulation aufgrund unzureichender Anfangsbedingungen Druckwellen und Strömungsartifakte, die der Realität nicht entsprechen.

[15] CPU: Pentium IV, 1700 MHz

Aufgrund der enorm großen Abmessungen des Simulationsgebietes im zweiten Beispiel werden alle vorhandenen Parameter und Rechenkapazitäten wegen der Begrenzung der Viskosität oder der enorm großen Anzahl der Gitterpunkte bei weitem überfordert. Selbst Hochleistungsrechner können hier das Problem nur abmindern, aber nicht lösen. Der Grund für die große Anzahl der Gitterpunkte liegt in der äquidistanten Natur des Rechengitters. So muss selbst in Bereichen, von denen man ausgehen kann, dass deren Einfluss auf die Lösung nur gering ist, der gleiche Rechenaufwand betrieben werden, wie im Bereich nahe des Kühlturms. Abhilfe könnte durch den Einsatz von nicht-uniformen Gittern geleistet werden. Auf diese Weise könnte sodann Rechenleistung problemgerecht im Berechnungsgebiet gezielt verteilt werden. Im nächsten Schritt würden sogenannte selbstorganisierende Gitter im Rahmen einer adaptiven Simulation die Verteilung der Freiheitsgrade für den Benutzer übernehmen, sodass die Effizienz der Berechnung optimiert wird.

Die theoretischen und algorithmischen Grundlagen für Lattice-Boltzmann Simulationen auf nicht-uniformen Gittern bilden einen weiteren Schwerpunkt dieser Arbeit, der im Folgenden beschrieben wird.

Teil II

Lattice-Boltzmann Strömungssimulationen auf nicht-uniformen Gittern

Kapitel 5

Grundlagen des Lattice-Boltzmann Verfahrens

In den letzten 15 Jahren wurde ausgehend von der *Lattice-Gas Methode* [58] die *Lattice-Boltzmann Methode* [46] als neues Verfahren zur näherungsweisen Lösung der *Navier-Stokes Gleichungen* entwickelt. Vor allem bei Forschern und Anwendern aus den Bereichen der numerischen Physik und den Ingenieurdisziplinen hat diese Methode großes Interesse gefunden.

Dieses Kapitel soll den Leser in die Lage versetzen, die Herleitung der Lattice-Boltzmann Methode prinzipiell zu verstehen. Die Erklärungen sind bewusst anschaulich formuliert und motiviert, sodass ohne vertiefte Kenntnisse der Thermodynamik und der statistischen Mechanik ein Zugang zur gängigen Standardliteratur (z. B. [99, 106]) verschafft werden soll.

5.1 Top-down versus bottom-up

Für die Diskretisierung der partiellen Differential-Gleichungen, die das Strömungsverhalten von Fluiden beschreiben, die Navier-Stokes Gleichungen, werden in konventionellen Verfahren meist Finite-Differenzen, Finite-Elemente oder Finite-Volumen Verfahren verwendet. Das makroskopische Verhalten von Fluiden wird hierbei durch Aufteilen des Gesamtproblems in viele kleine Teilprobleme (lokale Differenzengleichung) vereinfacht bestimmt. Dieses Vorgehen nennt man *Top-down Näherung* (vgl. Abb. 5.1).

In der Lattice-Boltzmann Methode wird der entgegengesetzte Weg gewählt. Die mikroskopische Welt spiegelt sich in einem diskreten Modell, in dem Masse-, Impuls- und Energieerhaltung per Konstruktion gewährleistet werden, wider. Mithilfe der *Multiskalen-Analyse*, auch *Chapman-Enskog Entwicklung* genannt, werden die makroskopischen Erhaltungsgleichungen, die *Navier-Stokes Gleichungen*, abgeleitet [106] (*bottom-up* Ansatz).

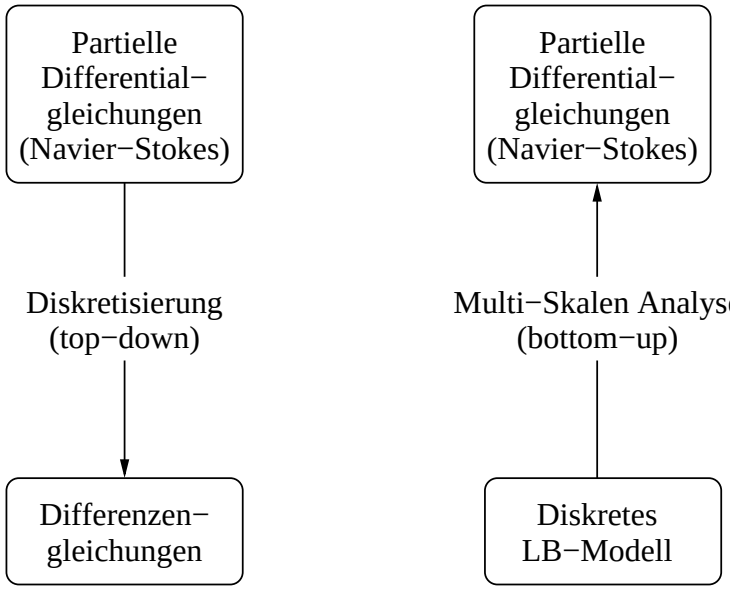

Abbildung 5.1: Top-down versus bottom-up [106]

5.2 Verteilungsfunktion und makroskopische Zustandsgrößen

Ein Kubikzentimeter Luft enthält unter Standardbedingungen ungefähr $2,69{\cdot}10^{19}$ Moleküle. Aufgrund der hohen Anzahl von Freiheitsgraden, die eine Simulation auf Partikelebene erfordern würde, ist leicht einzusehen, dass derzeit und auch in absehbarer Zukunft eine Simulation auf mikroskopischer Ebene, welche durch die sogenannten *Hamilton'schen Gleichungen* beschrieben wird, nicht möglich ist.

Ludwig Boltzmann führte die sogenannte Wahrscheinlichkeitsdichtefunktion[1] $f(t, \vec{x}, \vec{\xi})$ ein. Sie gibt für jeden Aufenthaltsort $\vec{x}$ zur Zeit t die Wahrscheinlichkeit dafür an, ein Partikel mit der Geschwindigkeit $\vec{\xi}$ vorzufinden. In Abbildung 5.2 werden die Zusammenhänge zwischen dem Fluid in der Makroskala und der Mikroskala aufgezeigt. Es werden nicht mehr einzelne Partikel betrachtet (Abb. 5.2 rechts oben), sondern ihr statistisch gemitteltes Verhalten an jedem Ort $\vec{x}$ zur Zeit t (Abb. 5.2 rechts unten). Mit der Verteilungsfunktion wird der Phasenraum[2] nunmehr angenähert beschrieben.

[1] Im weiteren Verlauf der Arbeit wird vereinfacht nur noch von Verteilungsfunktion oder Verteilungen gesprochen.

[2] Der Phasenraum wird durch den Ort $\vec{x}$ und der Geschwindigkeit $\vec{\xi}$ aller N Teilchen eines Kontrollvolumens aufgespannt.

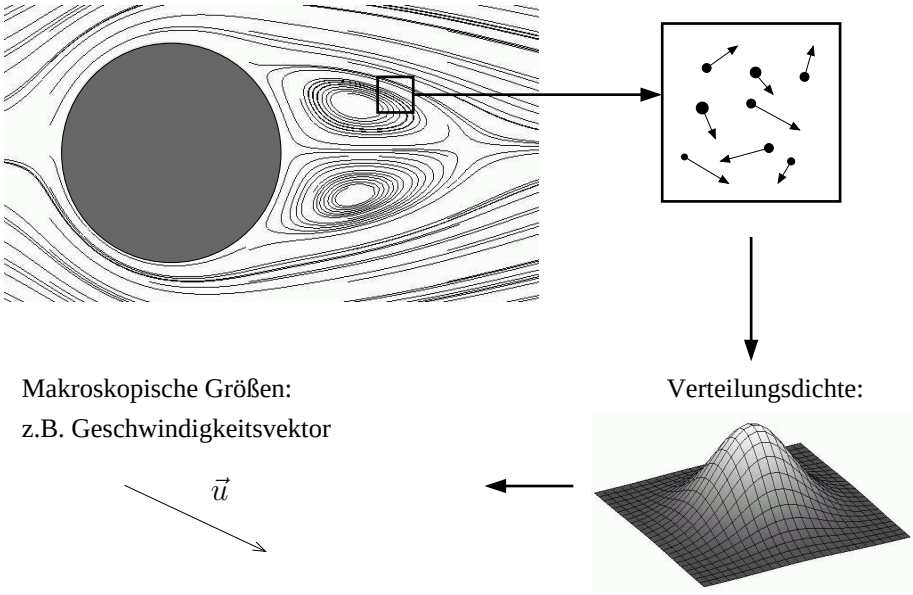

Abbildung 5.2: Zusammenhang zwischen der Makro- und Mikroskala

5.2.1 Makroskopische Größen

Unter der Annahme, dass alle Partikel gleichartig sind (gleiche Masse), ergeben sich die makroskopischen Größen als Momente [88, 102] der Verteilungsfunktion bezüglich der mikroskopischen Geschwindigkeit $\vec{\xi}$:

Dichte: Das Moment nullter Ordnung entspricht der Dichte.

$$\rho(t,\vec{x}) = \int_{\vec{\xi}=-\infty}^{\infty} (\vec{\xi})^0 f(t,\vec{\xi},\vec{x})\, d\vec{\xi} = \int_{\vec{\xi}=-\infty}^{\infty} f(t,\vec{\xi},\vec{x})\, d\vec{\xi} \tag{5.1}$$

Die Verteilungsfunktion gibt an, welcher Anteil f aller Partikel in einem Kontrollvolumen eine bestimmte Geschwindigkeit aufweist. Dieser Partikelanteil kann und muss auch als entsprechender Anteil an der Gesamtmasse aller Partikel aufgefasst werden. Bezogen auf ein Kontrollvolumen (vgl. Abb. 5.2) ist das genau die Dichte[3].

Impuls: Die Summe aller Teilimpulse, welche durch Multiplikation der Auftretenswahrscheinlichkeiten für bestimmte Geschwindigkeiten mit diesen Geschwindigkeiten berechnet wird, ergibt den Gesamtimpuls. Dies entspricht

[3] f ist eine Teilchendichte. Die Masse m eines Kontrollvolumens ergibt sich aus dem Produkt aus Einheitsmasse m_E und der Summe aller Teilchendichten $m = m_E \cdot \sum f$.

dem Moment erster Ordnung.

$$\rho(t,\vec{x})\vec{u}(t,\vec{x}) = \int_{\vec{\xi}=-\infty}^{\infty} (\vec{\xi})^1 f(t,\vec{\xi},\vec{x})\, d\vec{\xi} = \int_{\vec{\xi}=-\infty}^{\infty} \vec{\xi}\cdot f(t,\vec{\xi},\vec{x})\, d\vec{\xi} \tag{5.2}$$

Impulsstromdichtetensor: Der Impulsstromtensor wird durch das Moment zweiter Ordnung bestimmt und gibt den Transport des Impulses, der in Richtung α zeigt, in Richtung β an.

$$\Pi_{\alpha\beta}(t,\vec{x}) = \int_{\vec{\xi}=-\infty}^{\infty} \vec{\xi}_\alpha\vec{\xi}_\beta f(t,\vec{\xi},\vec{x})\, d\vec{\xi} = \rho u_\alpha u_\beta + D_{\alpha\beta} \quad \text{mit} \quad \vec{u} = \begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} \tag{5.3}$$

Im Impulsstromtensor ist der Drucktensor $D_{\alpha\beta}$ enthalten, welcher sich aus einem Kugelanteil und einem Deviatoranteil zusammensetzt.

$$D_{\alpha\beta} = \sigma_{Kugel} + \sigma_{Deviator} = P_{\alpha\beta} - S_{\alpha\beta} \tag{5.4}$$

$P_{\alpha\beta}$ ist der hydrostatische Druck.

$$P_{\alpha\beta} = p\cdot\delta_{\alpha\beta} \quad mit \quad p = c_s^2\cdot\rho \tag{5.5}$$

c_s ist die Schallgeschwindigkeit. Auf die Berechnung des Spannungstensors $S_{\alpha\beta}$ wird in Kapitel 5.8.1 näher eingegangen.

5.2.2 Gleichgewichtsverteilungen

Ein Fluid, das sich in einem spannungsfreien Zustand befindet, lässt sich mithilfe der sogenannten Gleichgewichtsverteilungen $f^{(M)}$ beschreiben. Dieser Zustand liegt nur dann vor, wenn im gesamten Strömungsgebiet sowohl die Dichte als auch die makroskopischen Geschwindigkeiten gleich sind. Für diesen Fall berechnet man die sogenannte *Maxwellverteilung* mit:

$$f^{(M)}(\rho,\vec{u}) = \frac{\rho}{2\pi c_s^2}\cdot e^{-\frac{(\vec{\xi}-\vec{u})^2}{2c_s^2}} \tag{5.6}$$

$\vec{\xi}$ ist die absolute, mikroskopische Geschwindigkeit der Teilchen und $\vec{u}$ die mittlere, makroskopische Geschwindigkeit. In einem geschlossenen System, das sich in Ruhe befindet, verschwindet $\vec{u}$, sodass sich Gleichung (5.6) zum absoluten Gleichgewicht

$$f^{(eq)}(\rho,\vec{u}) = \frac{\rho}{2\pi c_s^2}\cdot e^{-\frac{\vec{\xi}^2}{2c_s^2}} \tag{5.7}$$

vereinfacht. Im Folgenden werden die Gleichgewichtsverteilungen mit $f^{(0)}$ bezeichnet. Zu betonen ist nochmals, dass die Gleichgewichtsverteilungen *nur* von

der Dichte $\rho(t, \vec{x})$ und der Geschwindigkeit $\vec{u}(t, \vec{x})$ bzw. Impuls $\vec{j}(t, \vec{x})$, also nur implizit von der Zeit t und dem Ort $\vec{x}$ (siehe Kapitel 5.8.1), abhängen.

Die Nichtgleichgewichtsverteilungen $f^{(neq)}$ sind für die Berechnung der höheren Momente, wie z. B. des Spannungstensors, notwendig.

$$f = f^{(eq)} + f^{(neq)} = f^{(0)} + f^{(1)} + f^{(2)} + \cdots \approx f^{(0)} + f^{(1)} \tag{5.8}$$

In den diskreten Formulierungen wird nur noch $f^{(1)}$ verwendet werden.

5.3 Boltzmann-Gleichung

Das Verhalten von Fluiden kann mithilfe der Boltzmann-Gleichung auch jenseits des Kontinuumslimits[4] beschrieben werden.

$$\frac{\partial f}{\partial t} + \vec{\xi} \cdot \frac{\partial f}{\partial \vec{x}} + \vec{F} \cdot \frac{\partial f}{\partial \vec{\xi}} = Q(f, f) \quad mit \quad \vec{F} = \frac{\vec{K}}{m} \tag{5.9}$$

Dies ist eine Integro-Differentialgleichung der Verteilungsfunktion f, wobei $\vec{\xi}$ den mikroskopischen Geschwindigkeitsraum, $\vec{K}$ die Volumenkräfte und m die Masse der Partikel im Kontrollvolumen darstellt. Während die linke Seite der Gleichung (Differentialanteil) den advektiven Teilchentransport angibt, wird mit der rechten Seite das Kollisionsintegral, also die Interaktion der Teilchen, modelliert. Der Geschwindigkeitsraum wird von $\vec{\xi}$ aufgespannt. Die Ableitung der Boltzmann-Gleichung obliegt folgenden Annahmen [106]:

1. Es kommen nur zwei-Partikel Kollisionen in Betracht. Hierdurch werden auf der Boltzmann-Gleichung basierte Applikationen scheinbar auf verdünnte Gase[5] beschränkt. Tatsächlich reicht die Forderung nach kleinen Knudsenzahlen für eine Simulation von vielen Fluiden aus.

2. Vor der Kollision hängen die Geschwindigkeiten zweier Partikel nicht voneinander ab. Diese Annahme ist auch unter dem Begriff *Theorie vom molekularen Chaos* bekannt.

3. Extern wirkende Kräfte (Volumenkräfte) haben keinen oder keinen wesentlichen Einfluss auf den Kollisionsvorgang. Dies trifft dann zu, wenn diese deutlich kleiner sind als jene, die bei der Kollision auftreten.

[4] Ein Fluid befindet sich dann innerhalb des Kontinuumslimits, wenn für die Knudsenzahl gilt: $\epsilon \leq 0.01$. Die Knudsenzahl ist der Quotient aus Teilchenabstand d und makroskopischer Längenskala L: $\epsilon = \frac{d}{L}$

[5] Man spricht dann von einem verdünnten Gas, wenn das Volumen der Partikel klein gegen das Gesamtvolumen ist.

5.4 Kollisionsoperator

Im Lattice-Boltzmann Verfahren ist der Kollisionsoperator von zentraler Bedeutung. Durch ihn werden die Teilchenkollisionen unter Beachtung der Annahmen aus Kapitel 5.3 modelliert. Das Kollisionsintegral

$$Q(f,f) = \int_{\vec{\xi}} \int_{\Omega} \left\{ \sigma(\Omega) |\vec{\xi} - \vec{\xi}_1| \left[f(\vec{\xi'}) f(\vec{\xi'}_1) - f(\vec{\xi}) f(\vec{\xi}_1) \right] \right\} d\Omega \, d\vec{\xi}_1 \tag{5.10}$$

mit dem differentiellen Wirkungsquerschnitt der Zwei-Teilchen-Kollision $\sigma(\Omega)$ transformiert die ankommenden Geschwindigkeiten $\{\vec{\xi}, \vec{\xi}_1\}$ in die neuen Geschwindigkeiten $\{\vec{\xi'}, \vec{\xi'}_1\}$ [106].

5.4.1 Kollisionsinvarianten

Während des Kollisionsvorgangs dürfen sich die Masse, der Impuls und die Energie weder lokal noch global verändern. Es kann gezeigt werden, dass es entsprechend fünf Kollisionsinvarianten ψ_k gibt, die folgende Gleichung erfüllen:

$$\int_{\vec{\xi}} (Q(f,f) \cdot \psi_k) \, d\vec{\xi} = 0 \tag{5.11}$$

Die Invarianten korrelieren mit den Erhaltungsgrößen selbst:

$\psi_0 = 1$**:** Gleichung (5.11) mit der ersten Invariante entspricht der Massenerhaltung. Man kann den Kollisionsoperator auch als Veränderung der Verteilungsfunktion mit einer tendenziellen Annäherung zur Gleichgewichtsfunktion deuten.

$$Q(f,f) = \Delta f \tag{5.12}$$

In diesem Fall wird aus dem Integral (5.11) nichts anderes als eine Dichteänderung (Gleichung (5.1)), die natürlich verschwinden muss.

$$\Delta\rho = \int_{\vec{\xi}} Q((f,f) \cdot \psi_0) \, d\vec{\xi} = \int_{\vec{\xi}} (\Delta f \cdot 1) \, d\vec{\xi} = 0 \tag{5.13}$$

$\psi_\alpha = \xi_\alpha$**, für** $\alpha = 1,2,3$**:** Die weiteren drei Invarianten, eingesetzt in Gleichung (5.11), spiegeln die Impulserhaltung in den drei Raumrichtungen wider. Der Integrand steht in diesem Fall mit einer Impulsänderung, welche nicht stattfinden darf, in Beziehung.

$$\Delta\vec{j}_\alpha = \int_{\vec{\xi}} (Q(f,f) \cdot \psi_\alpha) \, d\vec{\xi} = \int_{\vec{\xi}} (\Delta f \cdot \vec{\xi}_\alpha) \, d\vec{\xi} = 0 \tag{5.14}$$

$\psi_4 = \vec{\xi}^2$**:** Die fünfte Invariante in Kombination mit Gleichung (5.11) entspricht der Energieerhaltung.

$$\Delta e = \int_{\vec{\xi}} (Q(f,f) \cdot \psi_4)\, d\vec{\xi} = \int_{\vec{\xi}} (\Delta f \cdot \vec{\xi}^2)\, d\vec{\xi} = 0 \tag{5.15}$$

Die Invarianten werden bei der Herleitung der Erhaltungsgleichungen eine entscheidende Rolle spielen.

5.4.2 BGK-Approximation des Kollisionsoperators

Der Kollisionsoperator (Gleichung (5.10)) ist extrem komplex und schwierig numerisch effizient umzusetzen. Bhatnagar, Gross und Krook stellten im Jahre 1954 ein einfaches Modell, den *BGK*-Kollisionsoperator, zur Beschreibung der Zwei-Teilchen Interaktion, vor [5]:

$$Q = -\frac{1}{\tau}(f - f^{(0)}) = -\frac{1}{\tau} f^{(neq)} \approx -\frac{1}{\tau} f^{(1)} \tag{5.16}$$

Er erfüllt alle erwünschten und auch zwingend erforderlichen Ansprüche:

1. Er ist numerisch effizient umzusetzen.
2. Er erfüllt die Erhaltungsgleichungen (siehe Kapitel 5.4.1).
3. Er tendiert zum globalen Gleichgewicht $f^{(0)}$.

Der Divisor τ ist die sogenannte Relaxationzeit, welche als Dauer des Kollisionsvorgangs interpretiert werden kann und in der Größenordnung der Knudsenzahl (vgl. Kapitel 5.5) liegen muss.

Die weiteren Erklärungen basieren auf dem BGK-Ansatz (Gleichung (5.16)). Prinzipiell kann der Kollisionsoperator jederzeit ausgetauscht werden, sofern er den eben genannten Ansprüchen genügt.

5.5 Chapman-Enskog Analyse

Die Herleitung der Navier-Stokes Gleichungen erfolgt unter dem Namen *Chapman-Enskog Analyse*[6]. Im Folgenden soll die Idee der Chapman-Enskog Analyse, die oft auch als *Multiskalen Analyse* bezeichnet wird, skizziert und der Weg von der Boltzmann Gleichung zu den Navier-Stokes Gleichungen angedeutet werden. Eine exakte und vor allem detaillierte Ableitung kann in [102, 106] nachgelesen werden.

[6]Diese Methode wurde zwischen 1910 und 1920 von den Herren Chapman und Enskog entwickelt [12].

Viele interessante Aspekte des Strömungsverhaltens von Fluiden liegen in der Variation der Verteilungsfunktion f in Raum und Zeit um die Gleichgewichtsverteilung $f^{(0)}$. Der *reale* Zustand des Kontinuums wird hierbei als Zusammensetzung vieler kleiner lokaler thermodynamischer Gleichgewichte, deren Parameter wie etwa Masse, Impuls und Energie sich nur wenig in Raum und Zeit verändern, angesehen. Um einen Eindruck über eine mögliche Interpretation dieser Variationen und dem Begriff *Multiskalen* zu erlangen, sei folgende Aufsplittung der Zeitskala genutzt:
Zunächst wird der Parameter ϵ^{-1} eingeführt, der die Größenordnung einer räumlichen Ausdehnung (z. B. $\epsilon^{-1}\Delta x = 100\Delta x$, mit $\Delta x =$ Gitterpunktabstand), in der Masse und Impuls spürbar variieren, angeben soll. ϵ ist die Knudsenzahl und ist dimensionslos. Es können nun drei Phänomene in Abhängigkeit ihrer Zeitskalen unterschieden werden [106]:

1. *Relaxation zum lokalen Gleichgewicht:*
   Um ein lokales, thermodynamisches Gleichgewicht zu erreichen, sind nur sehr wenige Kollisionen notwendig. Die entsprechende charakteristische Zeit liegt in der Größenordnung $\epsilon^0\Delta t$ ($\Delta t =$ Zeit in Gittereinheiten).

2. *Schallwellen und Advektion:*
   Die Entwicklung dieser Phänomene dauert länger als die Relaxation zum lokalen Gleichgewicht und liegt in der Größenordnung $\epsilon^{-1}\Delta t$.

3. *Diffusion:*
   Dieses Phänomen dauert noch länger und liegt in der Größenordnung $\epsilon^{-2}\Delta t$.

Bezogen auf eine räumliche Variation $\epsilon^{-1}\Delta x$, die eine Ausdehnung von 100 Gitterzellen aufweist, lässt sich der folgende zeitliche Zusammenhang erkennen. Die Relaxation zum lokalen Gleichgewicht verläuft in der Zeitskala $\epsilon^0\Delta t \approx \Delta t$ und dauert gewöhnlich nur wenige Zeitschritte. Die Dauer für die Ausbreitung von Informationen durch Advektion und Schall kann mit $\epsilon^{-1}\Delta t \approx 100\Delta t$ abgeschätzt werden. Diffusive Effekte breiten sich nur deutlich langsamer aus und können mit $\epsilon^{-2}\Delta t \approx 10000\Delta t$ angegeben werden [106].

In entsprechender Weise lässt sich die räumliche Ausdehnung selbst in Multiskalen zerlegen. Als Basis hierfür und als Entwicklungsparameter von f im Rahmen der Chapman-Enskog Analyse dient wieder die Knudsenzahl ϵ:

$$f = f^{(0)} + \epsilon f^{(1)} + \epsilon^2 f^{(2)} + \ldots \tag{5.17}$$

Der Parameter ϵ nimmt in dieser Entwicklung einen formalen Charakter an und wird zu Eins gesetzt. Er ist sehr hilfreich, um die relativen Größenordnungen der Terme untereinander erkennen zu können[7]. Dies wird beim Durchführen der Multiskalen Analyse von entscheidender Bedeutung sein.

[7] Im Kontinuumslimit liegen kleine Knudsenzahlen vor. Mit dem Term z. B. $\epsilon f^{(1)}$ wird durch ϵ angezeigt, dass die erste Variation der Verteilung $f^{(0)}$ sehr klein ist.

5.5.1 Erhaltungsgleichungen

Wie in Kapitel 5.4.1 erläutert, korrelieren die Erhaltungsgrößen mit den Kollisionsinvarianten. Es ist leicht einzusehen, dass wenn die Gleichung (5.11) gilt, dass dann auch die folgende Gleichung erfüllt sein muss, da dies die Integrale der rechten und der linken Seite der Boltzmann-Gleichung (Gleichung (5.9)) unter Weglassung der äußeren Kräfte sind:

$$\int_{\vec{\xi}} \left(\frac{\partial f}{\partial t} + \vec{\xi} \cdot \frac{\partial f}{\partial \vec{x}} \right) \cdot \psi_k \, d\vec{\xi} = 0 \tag{5.18}$$

Für $\psi_0 = 1$ ergibt sich die Erhaltungsgleichung für die Masse:

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u_\alpha)}{\partial x_\alpha} = 0 \tag{5.19}$$

Die Erhaltungsgleichungen für den Impuls werden in analoger Weise für die Kollisionsinvarianten $\psi_\alpha = \xi_\alpha$ abgeleitet:

$$\rho \frac{\partial u_\alpha}{\partial t} + u_\beta \frac{\partial (\rho u_\alpha)}{\partial x_\beta} = -\frac{\partial (D_{\alpha\beta})}{\partial x_\alpha} \tag{5.20}$$

$D_{\alpha\beta}$ ist der Drucktensor und ist nur in nullter (ϵ^0) Näherung bekannt:

$$D^{(0)}_{\alpha\beta} = c_s^2 \rho \delta_{\alpha\beta} \tag{5.21}$$

5.5.2 Spannungstensor

Bezugnehmend auf die Reihenentwicklung von f um $f^{(0)}$ mit dem Expansionsparameter ϵ wird der Drucktensor wie folgt entwickelt:

$$D_{\alpha\beta} = D^{(0)}_{\alpha\beta} + \epsilon D^{(1)}_{\alpha\beta} + \epsilon^2 D^{(2)}_{\alpha\beta} + \ldots \tag{5.22}$$

Der zweite Term der Entwicklung des Drucktensors führt zum Spannungstensor:

$$S_{\alpha\beta} = -\epsilon D^{(1)}_{\alpha\beta} + \mathcal{O}(\epsilon^2) = \epsilon \int \varsigma_\alpha \varsigma_\beta f^{(1)} \, d\vec{\xi} + \mathcal{O}(\epsilon^2) \quad \text{mit} \quad \varsigma_\alpha = \xi_\alpha - u_\alpha \tag{5.23}$$

Es wird deutlich, dass für die Berechnung des Spannungstensors nur die erste Variation der Verteilungen $f^{(1)}$ notwendig ist. Hierfür wird zunächst die Reihenentwicklung (5.17) in die Boltzmanngleichung mit BGK-Approximation des Kollisionsoperators eingesetzt

$$\begin{aligned} \frac{\partial}{\partial t}(f^{(0)} + \epsilon f^{(1)} + \epsilon^2 f^{(2)} + \ldots) + \vec{\xi} \cdot \frac{\partial}{\partial \vec{x}}(f^{(0)} + \epsilon f^{(1)} + \epsilon^2 f^{(2)} + \ldots) = \\ - \frac{1}{\epsilon}(f^{(0)} + \epsilon f^{(1)} + \epsilon^2 f^{(2)} + \ldots - f^{(0)}) \end{aligned} \tag{5.24}$$

und anschließend Gleichungen bezüglich der Potenzen von ϵ gebildet:

$$\mathcal{O}(\epsilon^{-1}): \qquad f^{(0)} - f^{(0)} = 0 \tag{5.25}$$

$$\mathcal{O}(\epsilon^{0}): \quad \frac{\partial f^{(0)}}{\partial t} + \vec{\xi} \cdot \frac{\partial f^{(0)}}{\partial \vec{x}} = -f^{(1)} \tag{5.26}$$

$$\mathcal{O}(\epsilon^{1}): \qquad \ldots \tag{5.27}$$

Die Gleichung (5.26) beinhaltet nur noch bekannte Größen bzw. solche Größen, die mithilfe der Kontinuitäts- und Impulsgleichungen abgeschätzt werden können. Dieses Vorgehen ist bereits ausführlich in der Dissertation von Tölke [102] beschrieben und soll hier nicht nochmal nachvollzogen werden. Es resultiert der Spannungstensor in folgender Form:

$$S_{\alpha\beta} = \tau c_s^2 \rho \left(\frac{\partial u_\alpha}{\partial x_\beta} + \frac{\partial u_\beta}{\partial x_\alpha} \right) + \mathcal{O}(\epsilon^2) \tag{5.28}$$

5.5.3 Inkompressible Navier-Stokes Gleichungen

In [106] wurde gezeigt, dass die Boltzmanngleichung in der Lage ist, die kompressiblen Navier-Stokes Gleichungen zu approximieren. Da dies jedoch die Reihenentwicklung von f um die (absoluten) Gleichgewichtsverteilungen $f^{(0)}$ voraussetzt und somit nur kleine Variationen von f in Betracht kommen, gelten die Ableitungen nur für kleine Mach- und Knudsenzahlen. In diesem Regime handelt es sich näherungsweise um inkompressible Strömungen. Die Dichte kann hier als konstant angenommen werden. Der Vollständigkeit halber werden die inkompressiblen Navier-Stokes Gleichungen aufgeführt:

$$\frac{\partial u_\alpha}{\partial x_\alpha} = 0 \tag{5.29}$$

$$\frac{\partial u}{\partial t} + u_\beta \frac{\partial u_\alpha}{\partial x_\beta} + \frac{\partial p}{\partial x_\alpha} = \nu \frac{\partial^2 u_\alpha}{\partial x_\beta \partial x_\beta} \tag{5.30}$$

5.6 Lattice-Boltzmann Gleichung

Die Herleitung der *Lattice-Boltzmann Gleichung* unterliegt mehreren Approximations- und Diskretisierungsschritten. Zunächst wird, wie in Kapitel 5.4.2 beschrieben, der Kollisionsoperator durch den BGK-Ansatz approximiert.

$$\frac{\partial f}{\partial t} + \vec{\xi} \cdot \frac{\partial f}{\partial \vec{x}} = Q(f, f) = -\frac{1}{\tau}(f - f^{(0)}) \tag{5.31}$$

Mit der Verteilungsfunktion $f(t, \vec{x}, \vec{\xi})$ wird der Geschwindigkeitsraum (oder auch: Phasenraum) kontinuierlich beschrieben. Durch die Einführung eines diskreten

Satzes von Geschwindigkeiten $\vec{\xi_i}$, erhält man einen diskreten Geschwindigkeitsraum, der durch die diskrete Verteilungsfunktion $f_i(t, \vec{x})$ aufgespannt wird. Die entsprechende *diskrete Boltzmann Gleichung* lautet:

$$\frac{\partial f_i}{\partial t} + \vec{\xi_i} \cdot \frac{\partial f_i}{\partial \vec{x}} = -\frac{1}{\tau}(f_i - f_i^{(0)}) \tag{5.32}$$

Die Gleichung (5.32) lässt sich als Gleichungssystem von i linear unabhängigen Gleichungen interpretieren. Eine Diskretisierung der räumlichen und zeitlichen Ableitungen mithilfe eines Finite-Differenzen Ansatzes bietet sich daher an [102]. Die Geschwindigkeiten $\vec{\xi_i}$ werden so gewählt, dass ein raumfüllendes Berechnungsgitter erzeugt wird:

$$\vec{\xi_i} = c \cdot \vec{e_i}, \quad \text{wobei} \quad \vec{e_i} = \text{Einheitsbasis} \tag{5.33}$$

Der Finite-Differenzen Ansatz führt zu folgendem Gleichungssystem[8]:

$$\frac{f_i(t + \Delta t, \vec{x}) - f_i(t, \vec{x})}{\Delta t} + c \cdot \frac{f_i(t + \Delta t, \vec{x} + \vec{e_i}\Delta x) - f_i(t + \Delta t, \vec{x})}{\Delta x_\alpha} = -\frac{1}{\tau}(f_i(t, \vec{x}) - f_i^{(0)}(t, \vec{x})) \tag{5.34}$$

Nach Multiplikation der Gleichung (5.34) mit Δt und Verwendung der Gleichung (5.33) erhält man die sogenannte *Lattice-Boltzmann Gleichung*. Der Gitterabstand Δx muss zu $\Delta x = c \cdot \Delta t$ gewählt werden:

$$f_i(t + \Delta t, \vec{x} + \vec{e_i}\Delta x) - f_i(t, \vec{x}) = -\frac{\Delta t}{\tau}(f_i(t, \vec{x}) - f_i^{(0)}(t, \vec{x})) \tag{5.35}$$

Für äquidistante Gitter wird $\Delta t = 1$ und $\Delta x = 1$ gesetzt und Gleichung (5.35) vereinfacht sich zu:

$$f_i(t + 1, \vec{x} + \vec{e_i}) - f_i(t, \vec{x}) = -\frac{1}{\tau}(f_i(t, \vec{x}) - f_i^{(0)}(t, \vec{x})) \quad \text{mit} \quad \tau = \frac{\nu}{3} + \frac{1}{2} \tag{5.36}$$

Nach [12] ist ν die kinematische Viskosität. Diese Gleichung ist Grundlage des sogenannten *Lattice-Boltzmann Verfahrens* und wird kurz *LBGK* genannt.

Die linke Seite der Gleichung (5.35) approximiert den advektiven Teilchentransport, während die rechte Seite die Kollision der Teilchen auf der Mikroskala mit dem BGK-Ansatz annähert.

5.7 Iterationszyklus

Durch Umformen der Gleichung 5.35 wird der Iterationszyklus (vgl. Abb. 5.3) sofort ersichtlich:

$$f_i(t + 1, \vec{x} + \vec{e_i}) = f_i(t, \vec{x}) - \frac{1}{\tau}(f_i(t, \vec{x}) - f_i^{(0)}(t, \vec{x})) \tag{5.37}$$

[8] Da $\vec{e_i}$ dimensionslos ist, ergibt der Term $\vec{e_i}\Delta x$ einen gerichteten Weg. c hat die Einheit einer Geschwindigkeit.

Auf der linken Seite befinden sich nur Größen des nächsten Zeitschritts, während sich auf der rechten Seite nur Größen des aktuellen Zeitschritts befinden. Demnach handelt es sich um ein *vollständig explizites* Zeitschrittverfahren. Die Verteilungen auf der rechten Seite befinden sich lokal an einem Ort. Somit kann ein Zeitschritt im Berechnungsalgorithmus logisch in die zwei Zwischenschritte *Kollision*

$$f_i^*(t,\vec{x}) = f_i(t,\vec{x}) - \frac{1}{\tau}(f_i(t,\vec{x}) - f_i^{(0)}(t,\vec{x})) \tag{5.38}$$

und *Propagation*

$$f_i(t+1,\vec{x}+\vec{e}_i) = f_i^*(t,\vec{x}) \tag{5.39}$$

unterteilt werden. Die Verteilungen f_i^* des nächsten Zeitschritts werden im Kollisionsschritt (Gleichung (5.38)) lokal berechnet und im darauf folgenden Propagationsschritt (Gleichung (5.39)) auf die entsprechend umliegenden Gitterpunkte kopiert. Die Kollision wird auch als *Relaxation* und die Propagation als *Advektion* bezeichnet.

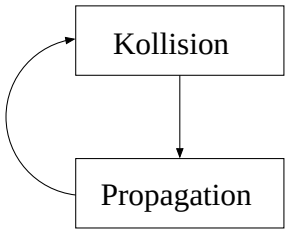

Abbildung 5.3: Basis-Zyklus einer LB-Simulation

5.8 D2Q9-Modell

Die Nomenklatur DkQb geht auf Qian [80] zurück. Das D steht dabei für die *Dimension* k, das Q für den Namensgeber, deutet aber die Anzahl b der diskreten Geschwindigkeiten, inklusive der Null-Geschwindigkeit (ruhende Partikel), im mikroskopischen Geschwindigkeitsraum an. Abbildung 5.4 zeigt den Einflussbereich der Verteilungen eines Gitterknotens eines D2Q9-Gitters für einen Zeitschritt. Die neun Geschwindigkeitsvektoren $\vec{c}_i$ sind wie folgt definiert:

$$\begin{aligned} \vec{c}_{\alpha i} &= (0,1,0,-1,0,1,-1,-1,1)^T = \vec{e}_{\alpha i}, \\ \vec{c}_{\beta i} &= (0,0,1,0,-1,1,1,-1,-1)^T = \vec{e}_{\beta i}, \\ \vec{c}_i &= (c_{\alpha i}, c_{\beta i})^T \quad mit \quad i = 0..8 \end{aligned} \tag{5.40}$$

Die Vektoren $\vec{c}_i$ spannen den diskreten Geschwindigkeitsraum für das D2Q9-Modell auf.

Bezugnehmend auf Kapitel 5.7 kann nun der Simulations-Zyklus veranschaulicht werden (vgl. Abb. 5.5 und 5.6). Im Kollisionsschritt werden die Verteilun-

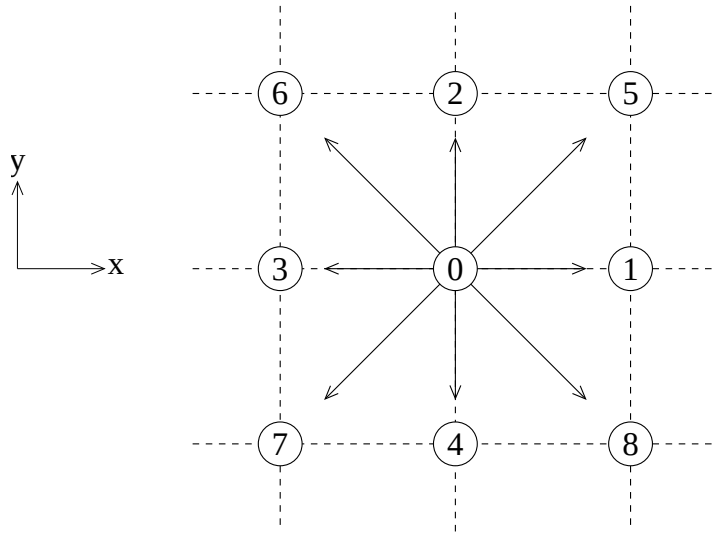

Abbildung 5.4: D2Q9 Gitter

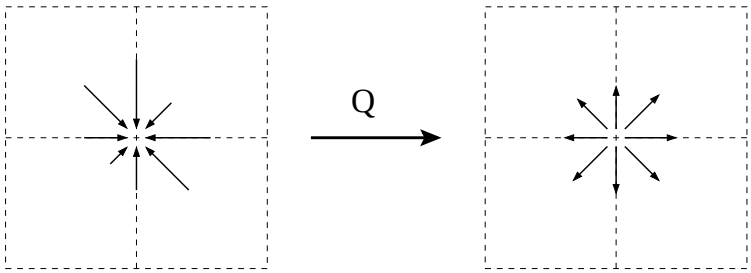

Abbildung 5.5: Kollision

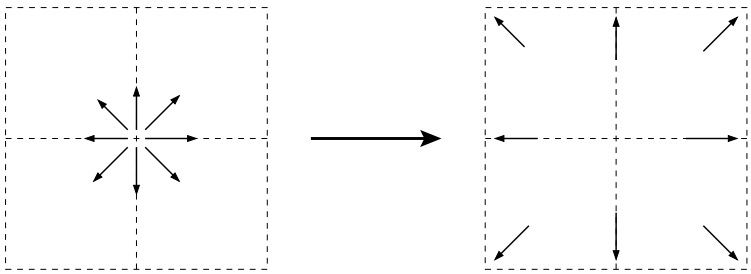

Abbildung 5.6: Propagation

gen entsprechend der Viskosität des Fluids so umgeordnet, dass die Verteilungen einen gleichförmigeren Zustand einnehmen. Im Propagationsschritt werden die Verteilungen entsprechend der Geschwindigkeitsvektoren auf dem Gitter bewegt.

5.8.1 Makroskopische Zustandsgrößen

Für die diskreten Lattice-Boltzmann Gleichungen werden die makroskopischen Größen entsprechend der Formulierungen aus Kapitel 5.2 bestimmt:

Dichte:

$$\rho(t,\vec{x}) = \sum_{i=0}^{8} f_i(t,\vec{x}) = \sum_{i=0}^{8} f_i^{(0)}(t,\vec{x}) \tag{5.41}$$

Geschwindigkeit:

$$\vec{u}(t,\vec{x}) = \frac{\sum_{i=0}^{8} \vec{c}_i f_i(t,\vec{x})}{\rho(t,\vec{x})} = \frac{\sum_{i=0}^{8} \vec{c}_i f_i^{(0)}(t,\vec{x})}{\rho(t,\vec{x})} \tag{5.42}$$

Spannungstensor:

$$S_{\alpha\beta}(t,\vec{x}) = (1-\frac{\Delta t}{2\tau}) \sum_{i=0}^{8} c_{\alpha i} c_{\beta i} f_i^{(1)}(t,\vec{x}) \quad \text{mit} \quad f^{(1)}(t,\vec{x}) \simeq f(t,\vec{x}) - f^{(0)}(t,\vec{x}) \tag{5.43}$$

Für die weiteren Diskussionen sind folgende Größen zu beachten:

Schallgeschwindigkeit: Es ist zwischen der physikalischen Schallgeschwindigkeit c_s und der numerischen Schallgeschwindigkeit c zu unterscheiden:

$$c_s = \frac{c}{\sqrt{3}} \quad \text{mit} \quad c = \frac{\Delta x}{\Delta t} = 1 \tag{5.44}$$

Druck:

$$p(t,\vec{x}) = c_s^2 \cdot \rho(t,\vec{x}) = \frac{1}{3}\frac{\Delta x^2}{\Delta t^2} \cdot \rho(t,\vec{x}) \tag{5.45}$$

5.8.2 Berechnung der Gleichgewichtsverteilungen

Für die Berechnung der Gleichgewichtsverteilungen $f^{(0)}$ werden die Dichte und die Komponenten des Geschwindigkeitsvektors benötigt. Es existieren zwei Berechnungsvarianten, wobei die inkompressiblere Version einige Kompressibilitätseffekte reduziert [42]. Die Formel zur Bestimmung der Gleichgewichtsverteilungen

des für stationäre Strömungen inkompressibleren Lattice-Boltzmann Modells lautet:

$$f_i^{(0)} = w_i \left\{ \rho + \rho_0 \left[3\frac{\vec{\xi_i} \cdot \vec{u}}{c^2} + \frac{9}{2}\frac{(\vec{\xi_i} \cdot \vec{u})^2}{c^4} - \frac{3}{2}\frac{\vec{u}^2}{c^2} \right] \right\} \tag{5.46}$$

In entsprechender Weise werden die Gleichgewichtsverteilungen des kompressibleren Modells mit

$$f_i^{(0)} = w_i \rho \cdot \left\{ 1 + 3\frac{\vec{\xi_i} \cdot \vec{u}}{c^2} + \frac{9}{2}\frac{(\vec{\xi_i} \cdot \vec{u})^2}{c^4} - \frac{3}{2}\frac{\vec{u}^2}{c^2} \right\} \tag{5.47}$$

berechnet. Die Wichtungskoeffizienten w_i sind dabei:

$$w_i = \begin{cases} \frac{4}{9} & i = 0 \\ \frac{1}{9} & i = 1, 2, 3, 4 \\ \frac{1}{36} & i = 5, 6, 7, 8 \end{cases} \tag{5.48}$$

5.8.3 Algorithmus der BGK-Lattice-Boltzmann Gleichung

Mit den bisher dargestellten Grundlagen lässt sich der Algorithmus eines Zeitschrittes (Simulationszyklus) wie folgt skizzieren:

Algorithmus A-1 LBGK-Algorithmus für einen Zeitschritt

1: **Berechne Kollision:**
2: **for all** Gitterknoten **do**
3: Berechne $f^{(0)}$ (Gleichung (5.47))
4: Berechne f^* des nächsten Zeitschrittes (Gleichung (5.38))
5: **end for**
6:
7: **Berechne Propagation:**
8: **for all** Gitterknoten **do**
9: Kopiere f_i^* (Gleichung (5.39))
10: **end for**

5.9 Momentenmethode

Die LBGK-Gleichung (Gleichung (5.35)) wurde unter dem Gesichtspunkt entwickelt, ein dynamisches System, basierend auf einem möglichst einfachen, symmetrischen Gitter und der Verteilungsfunktion f_i zu konstruieren. Der Relaxationsparameter $s = \frac{\Delta t}{\tau}$ ist für alle Gleichungen i konstant (Gleichung (5.35)). Man spricht daher auch vom *Single-Time-Relaxation-Approximation* Lattice-Boltzmann Modell, kurz *STRA-LBM* (LBGK). Das Verfahren wird instabil, wenn die

Viskosität zu klein gewählt wird ($\mathcal{O}(10^{-2})$), um beispielsweise eine gewünschte Viskosität oder Reynoldszahl einzustellen (vgl. Kapitel 4.6).

In den Arbeiten von d'Humiéres, Lallemand und Luo ist für die Relaxation (=Kollisionsvorgang) ein neues Modell, das *Generalized Lattice Boltzmann Equation* Modell (kurz: *GLBE*), das eine deutlich höhere Stabilität als der BGK-Ansatz aufweist, entwickelt und analysiert worden [63]. In weiteren Arbeiten wurde zudem die Effizienz und die Stabilität für 3D-Modelle, auch in Kombination mit Turbulenzmodellen nachgewiesen [21, 61, 64].

5.9.1 Geschwindigkeitsraum und Momentenraum

Wie in Kapitel 5.2 beschrieben, können die Momente der Verteilungsfunktion physikalisch interpretiert werden. Die Idee des neuen Modells besteht darin, aus den neun für das D2Q9-Modell vorhandenen Verteilungen f_i einen neuen Raum aus neun linear unabhängigen Momenten m_i zu konstruieren und die Relaxation im Momentenraum durchzuführen. Aufgrund der Orientierung der Verteilungen im Raumgitter muss die Propagation nach wie vor im Geschwindigkeitsraum stattfinden. Die Verteilungen $\vec{f}$ werden durch die Transformationsvorschrift

$$\vec{m} = M\vec{f} = m_i = [m_0, m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8]^T \tag{5.49}$$

mit

$$M \equiv \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -4 & -1 & -1 & -1 & -1 & 2 & 2 & 2 & 2 \\ 4 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & -1 & 0 & 1 & -1 & -1 & 1 \\ 0 & -2 & 0 & 2 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1 \\ 0 & 0 & -2 & 0 & 2 & 1 & 1 & -1 & -1 \\ 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \end{bmatrix} \tag{5.50}$$

in die Momente $m_i = \vec{m}$ überführt. Die entsprechende Vorschrift für die Rücktransformation der Momente m_i in den Geschwindigkeitsraum lautet:

$$\vec{f} = M^{-1}\vec{m} = f_i = [f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8]^T \tag{5.51}$$

mit

$$M^{-1} \equiv \begin{bmatrix} \frac{1}{9} & \frac{-1}{9} & \frac{1}{9} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{9} & \frac{-1}{36} & \frac{-1}{18} & \frac{1}{6} & \frac{-1}{6} & 0 & 0 & \frac{1}{4} & 0 \\ \frac{1}{9} & \frac{-1}{36} & \frac{-1}{18} & 0 & 0 & \frac{1}{6} & \frac{-1}{6} & \frac{-1}{4} & 0 \\ \frac{1}{9} & \frac{-1}{36} & \frac{-1}{18} & \frac{-1}{6} & \frac{1}{6} & 0 & 0 & \frac{1}{4} & 0 \\ \frac{1}{9} & \frac{-1}{36} & \frac{-1}{18} & 0 & 0 & \frac{-1}{6} & \frac{1}{6} & \frac{-1}{4} & 0 \\ \frac{1}{9} & \frac{1}{18} & \frac{1}{36} & \frac{1}{6} & \frac{1}{12} & \frac{1}{6} & \frac{1}{12} & 0 & \frac{1}{4} \\ \frac{1}{9} & \frac{1}{18} & \frac{1}{36} & \frac{-1}{6} & \frac{-1}{12} & \frac{1}{6} & \frac{1}{12} & 0 & \frac{-1}{4} \\ \frac{1}{9} & \frac{1}{18} & \frac{1}{36} & \frac{-1}{6} & \frac{-1}{12} & \frac{-1}{6} & \frac{-1}{12} & 0 & \frac{1}{4} \\ \frac{1}{9} & \frac{1}{18} & \frac{1}{36} & \frac{1}{6} & \frac{1}{12} & \frac{-1}{6} & \frac{-1}{12} & 0 & \frac{-1}{4} \end{bmatrix} \tag{5.52}$$

Beispiel:
Das Moment m_0 ergibt sich, wenn die erste Zeile der Matrix M mit dem Geschwindigkeitsverteilungsvektor $\vec{f}$ (Gleichung (5.51)) skalar multipliziert wird. Dies entspricht der Dichte ρ. Die skalare Multiplikation des Verteilungsvektors mit den Zeilen 4 und 6 entspricht jeweils der Berechnung der Impulse j_x und j_y.

In ähnlicher Weise können alle neun Momente physikalisch interpretiert werden:

$$\vec{m} = m_i = [\rho, e, \epsilon, j_x, q_x, j_y, q_y, p_{xx}, p_{xy}]^T \tag{5.53}$$

e, ϵ, q_x und q_y stehen in Beziehung zur kinetischen Energie und deren Fluss, p_{xx} und p_{xy} sind proportional zu den diagonalen und nicht-diagonalen Elementen des Spannungstensors [6].

Die Transformationen sind lineare Abbildungen zwischen zwei äquivalenten Räumen. Die Abbildungen sind eindeutig und sogar invertierbar, sodass die physikalische Aussagekraft durch die Transformation nicht leidet.

5.9.2 Relaxation im Momentenraum

Die Kollision mit dem BGK-Ansatz (vgl. Gleichung (5.35)) wird nun durch die Relaxation im Momentenraum ersetzt.

$$\begin{aligned} & f_i^*(t, \vec{x}) = f_i(t, \vec{x}) - \frac{1}{\tau}[f_i(t, \vec{x}) - f_i^{(0)}(t, \vec{x})] \\ \longrightarrow \quad & m_i^*(t, \vec{x}) = m_i(t, \vec{x}) - s_i[m_i(t, \vec{x}) - m_i^{(0)}(t, \vec{x})] \end{aligned} \tag{5.54}$$

Die für stationäre Strömungen inkompressiblere Ausführung der Berechnung der Gleichgewichtsmomente m_0 ist gegeben durch:

$$e^{(0)} = -2\rho + 3(j_x^2 + j_y^2) \tag{5.55}$$
$$\epsilon^{(0)} = \rho - 3(j_x^2 + j_y^2) \tag{5.56}$$
$$q_x^{(0)} = -j_x \tag{5.57}$$
$$q_y^{(0)} = -j_y \tag{5.58}$$
$$p_{xx}^{(0)} = j_x^2 - j_y^2 \tag{5.59}$$
$$p_{xy}^{(0)} = j_x j_y \tag{5.60}$$

Die Momente Dichte (Masse) und Impuls werden in der Relaxation nicht berücksichtigt, da sie Erhaltungsgrößen sind und sich während diesen Vorgangs nicht verändern dürfen. Aus Gründen der Modellierung [63] wird die Energie nicht als konservative Größe betrachtet. Die Schallgeschwindigkeit wird weiterhin mit Gleichung (5.44) bestimmt.

Der entscheidende Vorteil des Momentenmodells besteht darin, dass jedes Moment mit verschiedenen, physikalisch motivierten Relaxationparametern s_i relaxiert werden kann. Die entsprechende Relaxationsmatrix S ist diagonal und wird wie folgt definiert:

$$S = diag(0, s_2, s_3, 0, s_5, 0, s_7, s_8, s_9) \tag{5.61}$$

Die Relaxationparameter für die erhaltenden Momente ρ, j_x und j_y sind erwartungsgemäß 0. Aus Symmetriegründen müssen s_8 und s_9 (Spannungen) sowie s_5 und s_7 (Energiefluss) jeweils gleich sein. Somit existieren für das D2Q9-Modell vier unabhängige Parameter (*Multiple-Relaxation-Time*, kurz: *MRT*) zur Optimierung der Modelleigenschaften, die durch eine systematische Analyse der hydrodynamischen Eigenschaften bestimmt werden können [63].

In [63] werden beispielsweise folgende Relaxationsparameter vorgeschlagen:

$$S = diag\left(0, 1.63, 1.14, 0, 1.92, 0, 1.92, \frac{1}{\tau}, \frac{1}{\tau}\right) \tag{5.62}$$

Prinzipiell können die Relaxationsparameter s_2, s_3, s_5 und s_8 frei gewählt werden, müssen jedoch, um hydrodynamisch sinnvolle Ergebnisse zu erhalten, im Intervall

$$s_i \in [1, 2[\tag{5.63}$$

liegen. s_8 korreliert mit dem Spannungstensor und entspricht genau dem Relaxationsparameter des BGK-Ansatzes. Werden alle Relaxationsparameter $s_i \neq 0$ gleich $s_i = \frac{1}{\tau}$ gesetzt, reduziert sich das Momentenmodell zum LBGK-Modell. Die kinematische Viskosität ν kann im Vergleich zum BGK-Ansatz im GLBE-Modell bis auf Werte von 10^{-4} reduziert werden, wodurch der Bereich der erfassbaren Strömungsphänomene deutlich erweitert wird.

5.9.3 Algorithmus der MRT-Lattice-Boltzmann Gleichung

Die auf dem MRT-Modell basierende Lattice-Boltzmann Gleichung nimmt nun folgende Form an (siehe Gleichung (5.35) und (5.54)):

$$f_i(t+1, \vec{x} + \vec{e}_i) = M^{-1}\Big[m_i(t, \vec{x}) - s_i[m_i(t, \vec{x}) - m_i^{(0)}(t, \vec{x})]\Big] \tag{5.64}$$

Der entsprechende Algorithmus unterscheidet sich vom Algorithmus A-1 nur in der Kollision:

Algorithmus A-2 GLBE-Algorithmus für einen Zeitschritt

Berechne Relaxation:
for all Gitterknoten **do**
  Transformiere f in den Momentenraum, Gleichung 5.49
  Berechne $m^{(0)}$ (Gleichung 5.55 - 5.60)
  Berechne m^* des nächsten Zeitschrittes (Gleichung 5.54)
  Transformiere m zurück in den Geschwindigkeitsraum (Gleichung 5.51)
end for

Berechne Propagation:
for all Gitterknoten **do**
  Kopiere f_i^*(Gleichung 5.39)
end for

5.10 Zusammenfassung

Die Lattice-Boltzmann Modelle haben drei fundamentale Bestandteile:

1. Der diskrete Phasenraum wird durch ein Gitter und die entsprechenden Geschwindigkeitsvektoren $\vec{c}_i$ definiert. Die Verteilungsfunktion f_i wird entsprechend den Geschwindigkeitsvektoren auf jedem Knoten des Gitters definiert.

2. Die Maxwell'schen Gleichgewichts-Verteilungsfunktionen werden mittels der Größen Dichte und Impuls bestimmt.

3. Die zeitliche Entwicklung (Kinetik) wird durch Finite-Differenzen für den advektiven Teilchentransport und mittels des BGK-Ansatzes für die mikroskopische Teilcheninteraktion realisiert. Der BGK-Ansatz kann durch andere Modelle, wie z. B. das Momentenmodell, ohne weiteres ersetzt werden.

Die Simulationsparameter unterliegen aufgrund der Annahmen in der Herleitung des Lattice-Boltzmann Verfahrens einigen Einschränkungen:

Dichte: Die Dichte kann prinzipiell frei gewählt werden, da für die Simulation und das makroskopische Verhalten nur die Dichte- oder Druckunterschiede von Bedeutung sind. Für Auswerte-Routinen hat sich die Dichte $\rho_0 = 1.0$ als sinnvoll erwiesen.

Geschwindigkeit: Aufgrund der Annahmen in der Chapman-Enskog-Analyse (Entwicklung von f um das Ruhegleichgewicht, $\vec{\xi} \approx 0$) dürfen die makroskopischen Geschwindigkeiten den Wert $0.1\frac{\Delta x}{\Delta t}$ nicht oder nur geringfügig überschreiten.

Kinematische Viskosität: Aus theoretischen Gründen muss die kinematische Viskosität ν stets kleiner als $\frac{1}{6}$ sein. Eine genaue *untere Schranke* existiert nicht. Die numerische Stabilität ist hier das entscheidende Kriterium. Je nach Anforderungen der Strömung kann die Viskosität im BGK-Ansatz bis zu 10^{-2} angenommen werden. Durch Verwendung der Momentenmethode gewinnt man bis zu zwei Größenordnungen und kann die kinematische Viskosität strömungsbedingt bis auf Werte von 10^{-4} reduzieren. Die entsprechenden Relaxationszeiten $\tau_{LBGK} \approx 1,887$ und $\tau_{GLBE} \approx 1,998$ sollten nicht überschritten werden.

Eine ausführliche Diskussion der Gültigkeitsbereiche der physikalischen Größen und Parameter ist in Kapitel 6.6 zu finden.

Kapitel 6

Multiskalen Lattice-Boltzmann Verfahren

In diesem Kapitel werden die Grundlagen des Lattice-Boltzmann Verfahrens bezüglich der Erweiterungen für eine *Multiskalen* LB-Strömungssimulation vertieft. Unter einer Multiskalen LB-Strömungssimulation wird in diesem Kontext eine Simulation auf einem nicht-uniformen, kartesischen Gitter verstanden. Die Gitterpunktabstände können dabei um bis zu drei Größenordnungen variieren[1].

6.1 Einführung und Motivation

Wie in Kapitel 4.6 beschrieben, ist der beschränkende Faktor für die Simulation von ingenieurrelevanten Strömungsproblemen die fehlende Rechenleistung, die durch die enorme Anzahl von Freiheitsgraden für reale Probleme notwendig wird. Durch die Verwendung von uniform aufgelösten Strömungsgebieten werden in Bereichen, in denen keine nennenswerten Variationen der physikalischen Kenngrößen vorhanden sind und diese damit im physikalischen und numerischen Sinne nicht relevant sind, unnötig Freiheitsgrade verschwendet. Die Entwicklung und der Einsatz von Gitterverfeinerungsstrategien ist daher zwingend notwendig.

Während in bereits älteren und etablierten Simulationsverfahren, wie z. B. Finite-Elemente oder Finite-Volumen Methoden, die Verwendung von nicht-uniformen Netzen weit verbreitet ist und weitestgehend als erforscht gilt [10, 25, 93], befindet man sich bei den Lattice-Boltzmann Verfahren diesbezüglich noch in den Anfängen. Erste Ansätze wurden bereits entwickelt, verwenden aber bisher ausschließlich blockstrukturierte Gitter [29].

Prinzipiell kann man folgende Strategien unterscheiden:

Blockstrukturierte Gitter: Diese Gitter (siehe Kapitel 3.2.3) werden vorwiegend im Rahmen einer *a priori* Gitterverfeinerung eingesetzt. Hierbei wird

[1] In diesem Kontext wird von nun an von Multiskalen, also *Mehrskalen*, gesprochen.

jedoch Expertenwissen zur Definition der lokalen Netzdichte vorausgesetzt. Weder Geometrie, noch lokale physikalische Eigenschaften nehmen Einfluss auf die Verfeinerung.

Geometrisch angenäherte Gitter: Hierbei handelt es sich ebenfalls um eine *a priori* Gitterverfeinerung. Allerdings wird nur die Berandung der geometrischen Objekte automatisch feiner aufgelöst, indem diesen eine höhere Auflösung zugewiesen wird, als dem Fluid selbst. Der Gittergenerator hat dann eigenständig für einen akzeptablen Gitterübergang im Bereich der Auflösungsunterschiede zu sorgen. Die Lösung der Strömung beeinflusst die Form des Gitters ebenfalls nicht.

Adaptive, selbstorganisierende Gitter: Durch die Wahl geeigneter Algorithmen zur Qualitätsschätzung und entsprechender Datenstrukturen ist das Gitter in der Lage, lokale Verfeinerungen ohne jede Benutzerinteraktion durchzuführen. Eine optimierte Verteilung der Freiheitsgrade in Abhängigkeit von den lokalen Strömungsverhältnissen und den entsprechenden Fehlermaßen ist somit erzielbar (vgl. Kapitel 8).

Im Wesentlichen werden immer zwei Ziele verfolgt:

1. Die Anzahl der Freiheitsgrade soll für eine vorgegebene Genauigkeit reduziert oder bestenfalls minimiert werden.

2. Für eine gegebene, maximale Anzahl von Freiheitsgraden soll die Genauigkeit erhöht oder idealerweise maximiert werden.

In dieser Arbeit dienen Quadralbäume, das zweidimensionale Analogon der Oktalbäume, als Datenbasis der nicht-uniformen Gitter. Hieraus ergibt sich die Möglichkeit, die Form der Gitterübergänge (Interfaces) nahezu beliebig zu gestalten und sogar zur Laufzeit zu modifizieren (vgl. Kapitel 7).

Die Abbildung 6.1 zeigt einen Ausschnitt aus einem nicht-uniformen Gitter, bestehend aus zwei verschieden fein aufgelösten Gitterebenen. Dem gröbsten Level l_0 wird der orthogonale Gitterpunktabstand $\Delta x_g = 1$ zugewiesen. Aus Gründen der Datenhaltung (vgl. Kapitel 7.3.1) und der Reduktion der Anzahl von möglichen Gitterkonfigurationen (vgl. Kapitel 7.3.2) wird ein Unterschied der Baumtiefen benachbarter Gitterebenen von maximal *Eins* zugelassen. Damit unterscheiden sich die Ausdehnungen der Gitterzellen benachbarter Gitter maximal um den Faktor *Zwei*. Entsprechend müssen die Gittervektoren $\vec{v}_i$, welche den neundimensionalen Geschwindigkeitsraum aufspannen, für jedes Gitterlevel l angegeben werden:

$$\vec{v}_{i,l} = \Delta x_l \cdot \begin{bmatrix} e_{\alpha i} \\ e_{\beta i} \end{bmatrix} \quad \text{mit} \quad \Delta x_l = \left(\frac{1}{2}\right)^{(l-l_0)} \tag{6.1}$$

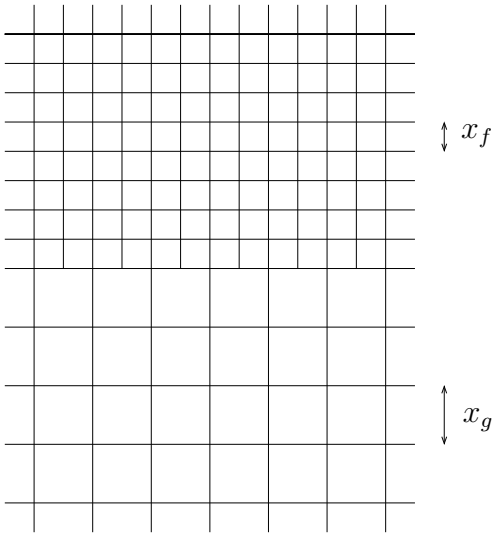

Abbildung 6.1: Minimalbeispiel für ein nicht-uniformes Gitter

Dies stellt ein für diese Methode spezifisches Problem dar, da die verschieden aufgelösten Gitter nun verschiedene Geschwindigkeitsräume, aufgespannt durch die in Formel 6.1 definierten Gittervektoren, besitzen. Daher liegen auf der Mikro - skala verschiedene Diskretisierungen vor, wodurch ein einfaches Übertragen durch Kopieren der Systemvariablen f_i zwischen den Gitterebenen nicht möglich ist. Die hierfür nötigen mathematischen Ableitungen wurden erstmals von Filippova und Hänel [29, 30] durchgeführt. Ausführlichere Darstellungen können in [31, 110] eingesehen werden. Die Ansätze dieser Arbeiten basieren im Gegensatz zur vorliegenden Arbeit auf blockstrukturierten Ansätzen.

Im Folgenden werden alle Erweiterungen beschrieben, die notwendig sind, um der Gesamtsimulation trotz der differierenden mikroskopischen Diskretisierungen ein homogenes makroskopisches Verhalten aufzuprägen. Alle weiteren Erklärungen werden, soweit nicht anders angegeben, auf ein Gitter mit zwei Auflösungsebenen $\{g, f\}$[2] bezogen.

6.2 Physikalische Eigenschaften

Für die Simulation von Fluiden in isothermen Systemen sind die Viskosität und die Schallgeschwindigkeit die wichtigsten physikalischen Eigenschaften. Diese müssen unabhängig von der lokalen Gitterdichte definiert sein.

[2]Die Index g entspricht der groben und f der feinen Auflösung.

6.2.1 Schallgeschwindigkeit

Die Schallgeschwindigkeit (siehe Gleichung 5.44) muss im gesamten System konstant sein. Als *Propagationsgeschwindigkeit* wird die numerisch maximal mögliche Ausbreitungsgeschwindigkeit von Information im System bezeichnet. In einem Zeitschritt Δt kann die Information (hier: die Verteilungsdichte f) um genau $\Delta x = c \cdot \Delta t$ propagiert werden.

$$c = \frac{\Delta x_g}{\Delta t_g} = \frac{\Delta x_f}{\Delta t_f} = 1 \tag{6.2}$$

Für ein Gitter, wie es in Abbildung 6.1 dargestellt ist, bedeutet dies, dass der feine Gitteranteil zwei Zeitschritte ($\Delta x_f = \Delta t_f = 0.5$), während der grobe Gitteranteil nur einen Zeitschritt ($\Delta x_g = \Delta t_g = 1$) arbeiten muss, um die gleiche Ausbreitungsgeschwindigkeit zu erzielen. Das feine Gitter führt doppelt soviel *Takte* durch, wie das grobe Gitter.

6.2.2 Viskosität und Relaxationszeit

Die kinematische Viskosität ν steht in direktem Zusammenhang mit der Relaxationszeit τ. In der Literatur (z. B. [46, 69]) wird die Relaxationszeit oft vereinfacht mit

$$\tau = 3 \cdot \left(\nu + \frac{1}{6}\right) \quad \Longleftrightarrow \quad \nu = \frac{\tau}{3} - \frac{1}{6} \tag{6.3}$$

angegeben, mit der Annahme, dass $c = 1, \Delta x = 1$ und $\Delta t = 1$. Für den allgemeinen Fall treffen diese Annahmen jedoch nicht mehr zu und τ_l muss in Abhängigkeit der Auflösung l und des damit verbundenen Zeitschritts Δt_l mit

$$\tau_l = 3 \cdot \left(\frac{\nu}{c^2} + \frac{\Delta t_l}{6}\right) = 3\nu + \frac{\Delta t_l}{2} \quad , \quad c = 1 \tag{6.4}$$

berechnet werden. Während die Viskosität konstant ist, muss die Relaxationszeit τ_l für jede Gitterebene, also für jeden Geschwindigkeitsraum bestimmt werden. Somit lautet die Lattice-Boltzmann Gleichung für verschiedene Auflösungen l:

$$f_{i,l}(t + \Delta t_l, \vec{x} + \vec{e}_i \Delta t_l) - f_{i,l}(t, \vec{x}) = -\frac{\Delta t_l}{\tau_l}(f_{i,l}(t, \vec{x}) - f_{i,l}^{(0)}(t, \vec{x})) \tag{6.5}$$

Der Relaxationsparameter s_l der Auflösung l wird mit

$$s_l = \frac{\Delta t_l}{\tau_l} \tag{6.6}$$

definiert.

6.3 Gitterübergangs-Bedingungen

Die Simulation auf Gittern unterschiedlicher Auflösung kann als eine Vielzahl von Simulationen auf äquidistanten Teilgittern interpretiert werden, wobei versucht wird, die jeweiligen Randbedingungen durch die benachbarten Gitter bestmöglich zu modellieren. Jene Bereiche, in denen sich die Teilgitter überlappen, werden im weiteren Verlauf als *Interface* bezeichnet (schraffierte Flächen in Abb. 6.2).

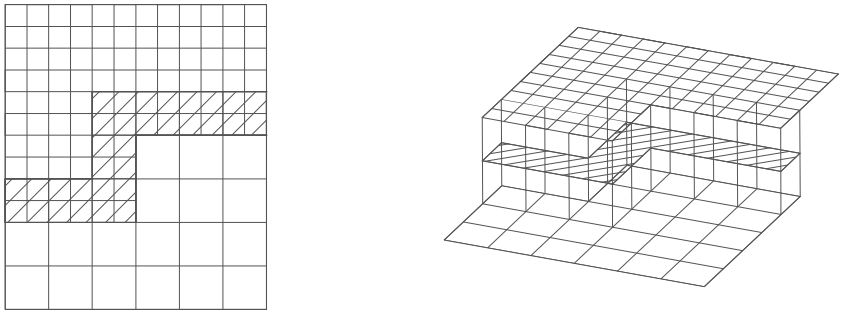

Abbildung 6.2: Zwei-Gittersystem mit Interface

Anhand der Abbildung 6.3 werden die Begriffe Interface, *feines* Interface und *grobes* Interface definiert:

Interface: Als Interface wird jener Bereich bezeichnet, der sowohl vom feinen, als auch vom groben Gitter überdeckt wird. Die Überlappung hat die Ausdehnung einer groben Gitterzelle. Auf die Notwendigkeit des Interfaces wird in Kapitel 6.4 im Detail eingegangen.

Feines Interface: Als feines Interface wird für zweidimensionale Gitter der eindimensionale Rand des feinen Gitters bezeichnet. Dieser befindet sich bereits um eine grobe Gitterzelle im Inneren des groben Gitters.

Grobes Interface: Das grobe Interface ist die eindimensionale Begrenzung des groben Gitters und befindet sich um zwei feine Gitterzellen versetzt innerhalb des feinen Gitters.

Die Herausforderung einer Multiskalen Lattice-Boltzmann Simulation besteht darin, die Kontinuität von Dichte, Impuls und **Spannungen** bei den Gitterübergänge zu gewährleisten. Die weiteren Erläuterungen sollen mithilfe eines Zwei-Gitter-Systems (vgl. Abb. 6.2) veranschaulicht werden.

6.3.1 Kontinuität von Dichte und Impuls

Die Gleichgewichtsverteilungen $f^{(0)}$ sind im Gegensatz zu den Nichtgleichgewichtsverteilungen $f^{(1)}$ *nicht* vom Geschwindigkeitsraum, sondern nur von den

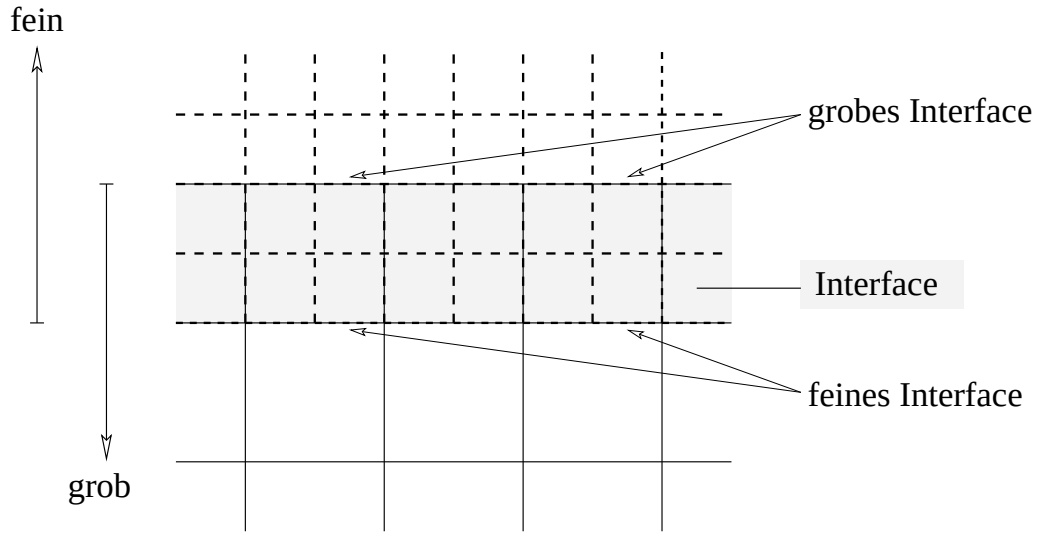

Abbildung 6.3: Detailzeichnung: Interface

makroskopischen Größen Dichte und Impuls abhängig (Gleichung (5.47)). Da diese Beziehung auch bidirektional gilt (Gleichungen (5.1)-(5.2)) ist die Kontinuität der primären Strömungsvariablen, der Dichte und des Impulses, bezüglich des Interfaces trivialerweise für alle Teilgitter der Auflösungen $l = \{g, f\}$ gegeben mit:

$$f_{i,g}^{(0)}(t, \vec{x}) = f_{i,f}^{(0)}(t, \vec{x}) = f_i^{(0)}(t, \vec{x}) \tag{6.7}$$

Im Interface können die Gleichgewichtsverteilungen an einem Ort $\vec{x}$ zur Zeit t beliebig zwischen allen Gitterebenen ausgetauscht werden.

6.3.2 Kontinuität der Spannungen

Um weiterhin für die Ableitungen der Strömungsvariablen, die Spannungen, einen kontinuierlichen und konsistenten Verlauf am Interface zu gewährleisten, müssen weitere Beziehungen zwischen den verschiedenen Gitterebenen abgeleitet werden.

Zunächst wird $f_{i,l}(t + \Delta t_l, \vec{x} + \vec{e_i}\Delta t_l)$ aus Gleichung (5.35) durch eine Taylorreihen-Entwicklung um $\Delta t = 0$ approximiert und Δx_α durch $e_{i,\alpha}\Delta t_l$ substituiert:

$$f_{i,l}(t+\Delta t_l, \vec{x}+\vec{e_i}\Delta t_l) = f_{i,l}(t, \vec{x}) + \frac{\partial f_{i,l}(t, \vec{x})}{\partial t}\Delta t_l + e_{i,\alpha}\frac{\partial f_{i,l}(t, \vec{x})}{\partial x_\alpha}\Delta t_l + \mathcal{O}(\Delta t_l^2) \tag{6.8}$$

Einsetzen von Gleichung 6.8 in Gleichung 5.35 und anschließende Division durch Δt_l liefert die sogenannte *äquivalente Differentialgleichung*. Zur vereinfachten Darstellung wird im weiteren Verlauf auf die Darstellung der Abhängigkeit von Raum und Zeit $(t, \vec{x})$ verzichtet und die Propagationsgeschwindigkeit c mit 1

angenommen:

$$\frac{\partial f_{i,l}}{\partial t} + e_{i,\alpha}\nabla f_{i,l} + \mathcal{O}(\Delta t_l) = \frac{Df_{i,l}}{Dt} + \mathcal{O}(\Delta t_l) = -\frac{1}{3\frac{\nu}{c^2} + \frac{1}{2}\Delta t_l}(f_{i,l} - f_{i,l}^{(0)}) \quad (6.9)$$
$$= -\frac{1}{3\nu + \frac{1}{2}\Delta t_l}(f_{i,l}^{(1)})$$

Die Verteilungsfunktionen $f_{i,l}$ werden für kleine Knudsenzahlen ϵ in einer Reihe entwickelt:

$$f_{i,l} = \epsilon^0 f_{i,l}^{(0)} + \epsilon^1 f_{i,l}^{(1)} + \mathcal{O}(\epsilon^2) = f_{i,l}^{(0)} + \epsilon f_{i,l}^{(1)} + \mathcal{O}(\epsilon^2) \quad (6.10)$$

Die so erhaltene Abschätzung von $f_{i,l}$ wird in Gleichung 6.9 eingesetzt und anschließend ein Gleichungssystem bezüglich der Potenzen von ϵ gebildet (Multiskalen Analyse). Als erste Näherung erhält man die vereinfachte partielle Differentialgleichung bezüglich ϵ^0:

$$\frac{Df_{i,l}^{(0)}}{Dt} + \mathcal{O}(\Delta t_l) = -\frac{1}{3\nu + \frac{1}{2}\Delta t_l}(f_{i,l}^{(1)}) \quad (6.11)$$

Da $f_{i,l}^{(0)}$ für alle Gitter der Auflösung l identisch ist, bedingt die Gleichheit der totalen Ableitungen $\frac{Df_{i,l}^{(0)}}{Dt}$ für die Nichtgleichgewichtsanteile $f_{i,l}^{(1)}$ folgende Beziehung:

$$scale_{f\to g} := \frac{f_{i,g}^{(1)}}{f_{i,f}^{(1)}} = \frac{3\nu + \frac{1}{2}\Delta t_g}{3\nu + \frac{1}{2}\Delta t_f} \quad (6.12)$$

$$scale_{g\to f} := \frac{1}{scale_{f\to g}} \quad (6.13)$$

Anhand des diskreten Spannungstensors $S_{\alpha\beta}$ kann nun die Gleichheit der Spannungen bei Verwendung der Beziehung (6.12) bestätigt werden. In diskreter Form lautet der Spannungstensor der groben Auflösung:

$$S_{\alpha\beta,g} = (1 - \frac{\Delta t_g}{2\tau_g})\sum_{i=0}^{8} e_{\alpha i}e_{\beta i}f_{i,g}^{(1)} = (1 - \frac{\Delta t_g}{6\nu + \Delta t_g})\sum_{i=0}^{8} e_{\alpha i}e_{\beta i}f_{i,g}^{(1)} \quad (6.14)$$

Die Nichtgleichgewichtsanteile $f_{i,g}^{(1)}$ des *groben* Spannungstensors werden durch Skalierung (Gleichung (6.12)) der *feinen* Nicht-Gleichgewichtsverteilungen $f_{i,f}^{(1)}$ ersetzt:

$$S_{\alpha\beta,g} = (1 - \frac{\Delta t_g}{6\nu + \Delta t_g})\sum_{i=0}^{8} e_{\alpha i}e_{\beta i}\left(\frac{3\nu + \frac{1}{2}\Delta t_g}{3\nu + \frac{1}{2}\Delta t_f}f_{i,f}^{(1)}\right) = \quad (6.15)$$

$$= (1 - \frac{\Delta t_f}{6\nu + \Delta t_f})\sum_{i=0}^{8} e_{\alpha i}e_{\beta i}f_{i,f}^{(1)} = S_{\alpha\beta,f} \quad \text{(q.e.d.)} \quad (6.16)$$

Somit lassen sich die Verteilungen am Interface zweier Gitter unterschiedlicher Auflösung mithilfe der Skalierungsfaktoren $scale_{f\to c}$ und $scale_{c\to f}$ aufeinander abbilden:

$$f_{i,f} = f_{i,g}^{(0)} + scale_{g\to f} \cdot f_{i,g}^{(1)} \tag{6.17}$$

$$f_{i,g} = f_{i,f}^{(0)} + scale_{f\to g} \cdot f_{i,f}^{(1)} \tag{6.18}$$

6.3.3 Räumliche Kontinuität

Beim Übergang von einem feinen auf ein grobes Gitter treten am feinen Interface hängende Knoten auf. Wie in Abbildung 6.4 angedeutet, sind jeweils die drei von der groben auf die feine Seite zeigende Verteilungen unbekannt. Um die räum-

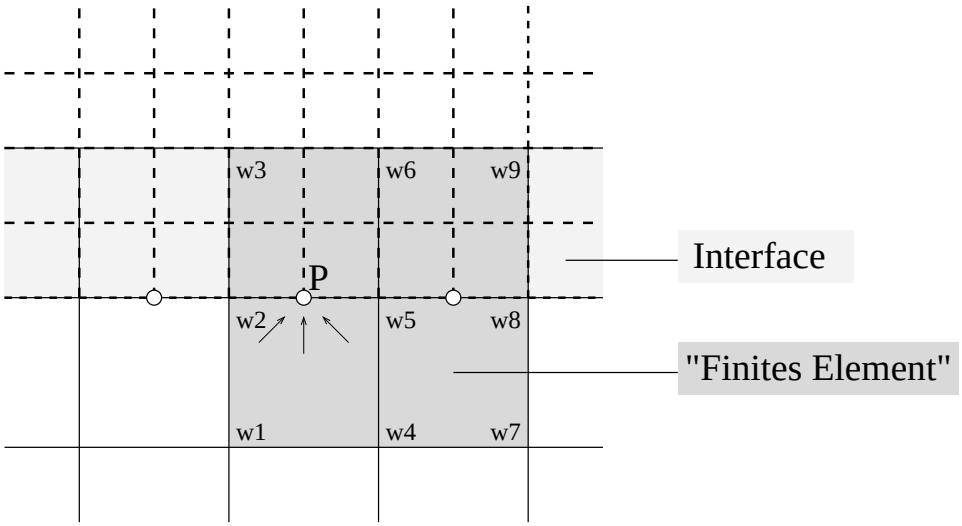

Abbildung 6.4: Interface, Finites Element und hängender Knoten P

liche Kontinuität zu gewährleisten, hat sich der Ansatz einer zweidimensionalen Interpolation von mindestens zweiter Ordnung zur Rekonstruktion der fehlenden Verteilungen als vielversprechend erwiesen. Hierfür wird ein neunknotiges *"Finites Element"* so über das Interface gelegt, dass es sowohl Gitterknoten feiner als auch grober Auflösung überdeckt. Als Ansatzfunktion wird ein quadratisches Polynom der Form

$$w(x,y) = a_0 + a_1 x + a_2 y + a_3 xy + a_4 x^2 + a_5 y^2 + a_6 x^2 y + a_7 y^2 x + a_8 x^2 y^2 \tag{6.19}$$

gewählt. Die Größen w_k entsprechen den zu interpolierenden Größen (z. B. f_2). Für die zu interpolierende Verteilung des Punkts P, der exakt auf halber Strecke zwischen P2 und P5 liegt, resultiert folgende Interpolationformel:

$$w(P) = \frac{3}{8}w2 + \frac{3}{4}w5 - \frac{1}{8}w8 \tag{6.20}$$

Wie zu erkennen ist, hat sich die biquadratische zweidimensionale Interpolation auf eine quadratische eindimensionale Interpolation reduziert. Es kann ge-

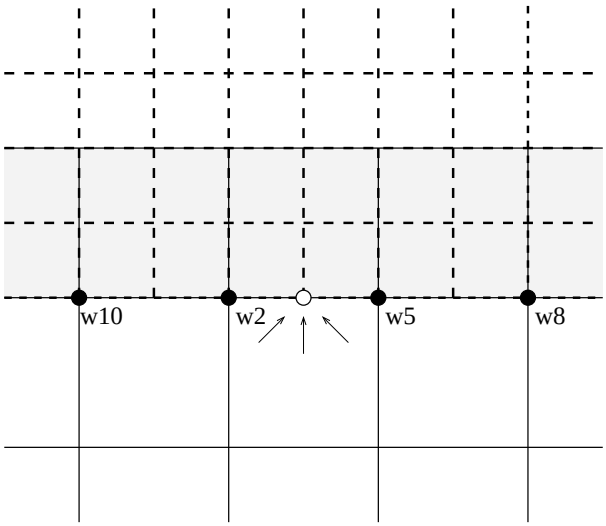

Abbildung 6.5: Kubische Interpolation

zeigt werden, dass die quadratische Interpolation der Lösung der Simulation eine Asymmetrie aufgeprägt. Deshalb wird eine kubische Interpolation (vgl. Abb. 6.5) entlang des Interface vorgeschlagen und verwendet:

$$w(P) = -\frac{1}{16}w10 + \frac{9}{16}w2 + \frac{9}{16}w5 - \frac{1}{16}w8 \tag{6.21}$$

Für einige Gitterkonstellationen, in denen die Knoten $w8$ und $w10$ nicht gleichzeitig dem groben und dem feinen Gitter angehören, werden dennoch die Verteilungen beider Gitterebenen vorgehalten, um eine kubische Interpolation zu ermöglichen (vgl. Kapitel 7.3.2 und Abb. 7.8).
Andere Interpolationsschemata, wie z. B. kubische Splines, haben keine erkennbaren Verbesserungen gezeigt.

6.3.4 Zeitliche Kontinuität

Da Gitter mit höherer Auflösung öfter arbeiten (=takten) als das am gröbsten aufgelöste Gitter, wird eine Interpolation in der Zeit für den Übergang von grob nach fein erforderlich. Diese findet in dieser Implementierung (für zwei benachbarte Gitter gilt: $\Delta x_g = 2 \cdot \Delta x_f$ und $\Delta t_g = 2\Delta t_f$) genau zwischen zwei Takten des groben Gitters statt, also dann, wenn das feine Gitter taktet und das grobe nicht. Hierfür kommen im Wesentlichen drei Interpolations-Schemata zur Bestimmung der gesuchten Verteilungen im *Zwischenschritt* T in Frage (vgl. Abb. 6.6):

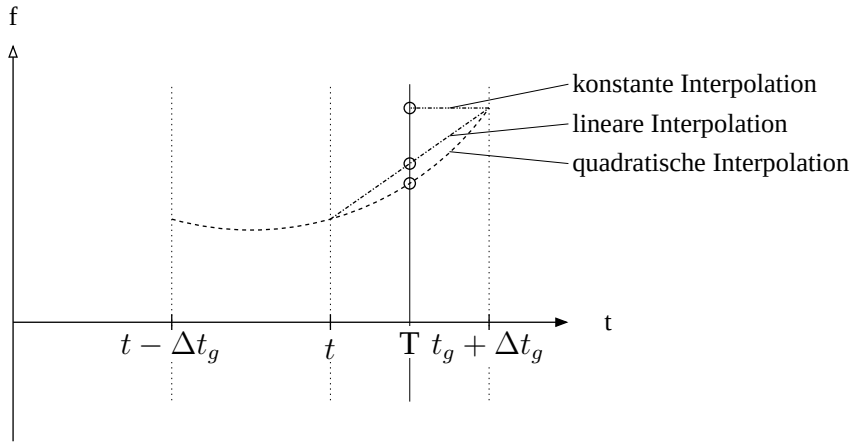

Abbildung 6.6: Zeitinterpolationsschemata

Konstante Interpolation: Da das gröbere Gitter sich zeitlich entweder auf gleichem oder im fortgeschrittenem Niveau (der Zukunft) befindet, werden für die *feinen* Verteilungen $f(T)$ zum Zeitpunkt T die von grob nach fein skalierten Verteilungen des Zeitschritts $t + \Delta t_g$ kopiert. Für stationäre Strömungen reicht diese Form der Approximation aus.

$$f(T) = f(t + \Delta t_g) \quad \text{mit} \quad \Delta t_g = 1 \tag{6.22}$$

Lineare Interpolation: Für instationäre Strömungen sollte jedoch mindestens eine lineare Interpolation verwendet werden.

$$f(T) = \frac{1}{2}\big(f(t) + f(t + \Delta t_g)\big) \tag{6.23}$$

Quadratische Interpolation: Der Einsatz von Polynom-Interpolationen zweiter Ordnung ist bei grob aufgelösten Gitter vorteilhaft. Hierzu wird ein quadratisches Polynom der Form

$$f(t) = a_0 + a_1 t + a_2 t^2 \tag{6.24}$$

verwendet. Die resultierende Interpolationsformel lautet:

$$f(T) = -\frac{1}{8} f(t - \Delta t_g) + \frac{3}{4} f(t) + \frac{3}{8} f(t + \Delta t_g) \tag{6.25}$$

Da die quadratische Interpolationsstrategie die Ergebnisse nur unwesentlich verbessert und die konstante Interpolation nur für stationäre Strömungssimulationen genügende Qualität aufweist, wird die lineare Interpolation empfohlen.

6.4 Algorithmus der Multiskalen LB-Simulation

In den vorhergehenden Punkten wurden die notwendigen Teilaufgaben isoliert betrachtet. Von entscheidender Bedeutung ist allerdings, diese Teilaufgaben in der algorithmisch und zeitlich korrekten Reihenfolge auszuführen. Wie am Anfang diesen Kapitels bemerkt, können die Gitterübergänge als Randbedingungen, mit bestmöglicher Modellierung, betrachtet werden.
Das Grundprinzip bei der Implementierung von Randbedingungen ist generell für jede Randbedingung anwendbar und lautet:

> Vor dem Kollisionsvorgang müssen alle Fluid-Knoten einen gültigen und vollständigen Satz von Verteilungen f_i besitzen. Nicht vorhandene Verteilungen müssen hierfür bestmöglich approximiert werden.

Wegen der zentralen Bedeutung der Multiskalen Lattice-Boltzmann Simulation und deren Algorithmus in dieser Arbeit wird der zeitliche Ablauf der Teilvorgänge Kollision, Skalierungen, räumliche und zeitliche Interpolationen sowie der Propagation sowohl als Pseudocode (Algorithmus A-3) als auch als grafische Abfolge anhand eines eindimensionalen Beispiels (vgl. Abb. 6.7) für einen gesamten Simulationszeitschritt in den Abbildungen 6.8 bis 6.13 dargestellt. In diesem Modell existieren nur die Ruheverteilung f_0, die nach rechts zeigende Verteilung f_1 und die nach links zeigende Verteilung f_2. Zur Vereinfachung der Erklärungen werden einige Begriffe wie folgt definiert:

Voller Takt: Das zeitliche und räumliche Bezugssystem ist das gröbste Gitter, d. h. $\Delta x = \Delta t = 1$ per Definition. Ein *voller Takt* oder auch *grober Takt* ist dann abgeschlossen, wenn alle Verteilungen aller Gitterebenen wieder gültig sind und das zeitliche Niveau des gröbsten Gitters erreicht haben.

Feiner Takt: Ein vollständig durchgeführter Zeitschritt eines Gitters höherer Auflösung wird als *feiner Takt* bezeichnet. Bei n-facher Auflösung werden n feine Takte benötigt, um das gleiche zeitliche Niveau eines groben Takts zu erreichen.

Im folgenden Algorithmus wird der zu leistende Mehraufwand im Vergleich zum Algorithmus A-1 infolge der Übergangsbedingungen erkennbar. Die Gitterknoten der Auflösung l werden im Algorithmus mit GK_l bezeichnet:

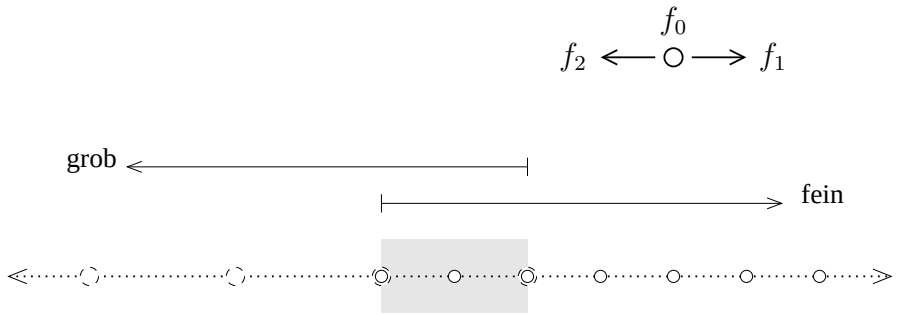

Abbildung 6.7: 1D-Gitterkonfiguration

Algorithmus A-3 LBGK-Multiskalen-Algorithmus für einen *groben* Zeitschritt Δt_g eines Zwei-Gitter-Systems

1: **for all** $GK_{l=\{f,g\}}$ **do**
2: Kollision (Gleichung 5.38, Abb. 6.8)
3: **end for**
4: **for all** $GK_{l=\{f,g\}}$ **do**
5: Propagation (Gleichung 5.39, Abb. 6.9)
6: **end for**
7: **for all** $GK_{l=g} \in$ feines Interface **do**
8: Skaliere grob nach fein (Gleichung 6.17, Abb. 6.10)
9: Zeitinterpolation des feinen Zwischenschritts (Gleichung 6.23, Abb. 6.10)
10: Rauminterpolation der hängenden Knoten (Gleichung 6.21)
11: **end for**
12: **for all** $GK_{l=f}$ **do**
13: Kollision (Gleichung 5.38, Abb. 6.11)
14: **end for**
15: **for all** $GK_{l=f}$ **do**
16: Propagation (Gleichung 5.39, Abb. 6.11)
17: **end for**
18: **for all** $GK_{l=g} \in$ grobes Interface **do**
19: Skaliere fein nach grob (Gleichung 6.18, Abb. 6.12)
20: **end for**
21: **for all** $GK_{l=g} \in$ feines Interface **do**
22: Skaliere grob nach fein (Gleichung 6.17, Abb. 6.12)
23: Rauminterpolation der hängenden Knoten (Gleichung 6.21)
24: **end for**

Dieser auf die gröbste Auflösung bezogene Zyklus stellt wiederum den Baustein der expliziten Gesamtzeitschleife dar.

Die Erklärungen und Illustrationen beziehen sich auf ein Gitter mit zwei verschiedenen Gitterauflösungen. Hierfür ist in den Abbildungen 6.8 bis 6.13 ein

eindimensionales Zeitraumgitter mit unterschiedlicher Zeit- und Raumdiskretisierung der Gitterauflösungen dargestellt. Für (theoretisch) beliebig viele verschieden aufgelöste Gitterkonfigurationen ist der Algorithmus anhand der dargestellten Vorgehensweise zu erweitern.

Zum Zeitpunkt t befinden sich alle *Fluid*-Verteilungen des Gitters auf gleichem zeitlichen Niveau und sind gültig. Zunächst wird auf allen Fluid-Gitterknoten die Kollision durchgeführt. Im Bereich des Interfaces wird an den sich überlagernden Gitterknoten der Teilgitter die Kollision der groben *und* der feinen Verteilungen berechnet (vgl. Abb. 6.8).

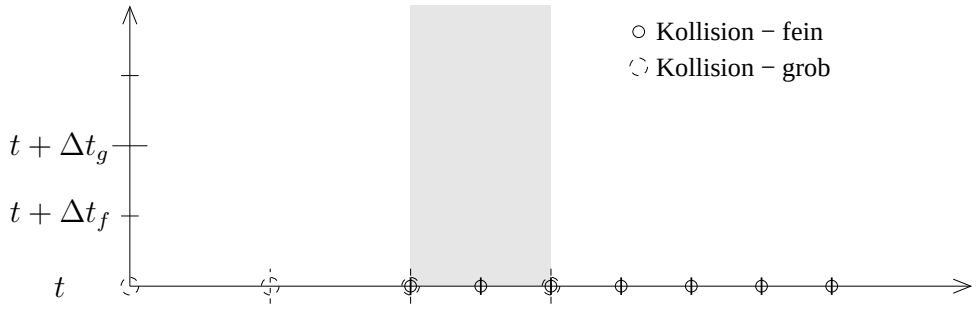

Abbildung 6.8: Simultane Kollision der feinen und groben Verteilungssets

Mit der darauffolgenden Propagation aller Verteilungen werden die Verteilungen entsprechend der Geschwindigkeitsvektoren auf dem Gitter bewegt. Die groben Verteilungen sind zeitlich und räumlich doppelt so weit fortgeschritten wie die feinen Verteilungen (vgl. Abb. 6.9).

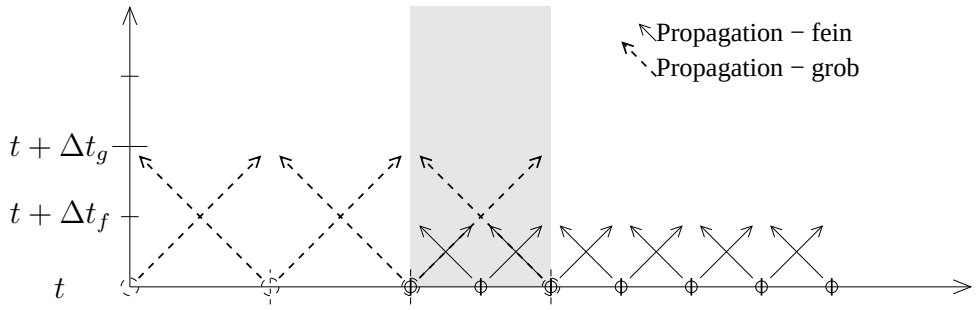

Abbildung 6.9: Simultane Propagation der feinen und groben Verteilungssets

Wie in Abbildung 6.10 zu sehen ist, konnte der feine Interface-Knoten keine Verteilung von links durch die Propagation zum Zeitpunkt $t + \Delta t_f$ empfangen, weil jenseits des feinen Interfaces an der entsprechenden Stelle keine feinen Verteilungen vorliegen. Um diese zu rekonstruieren, werden die groben Verteilungen

des feinen Interface im Zeitpunkt $t + \Delta t_g$ in den feinen Geschwindigkeitsraum skaliert. Nun können die fehlenden Verteilungen aus den vorhandenen feinen Verteilungen der Zeitpunkte t und $t + \Delta t_g$ durch lineare Interpolation gewonnen werden. In diesem Beispiel ist eine räumliche Interpolation nicht notwendig, da in der 1D-Version keine hängenden Knoten auftauchen. Für zweidimensionale oder dreidimensionale Gebiete wäre nach der zeitlichen Interpolation, die räumliche Interpolation notwendig.

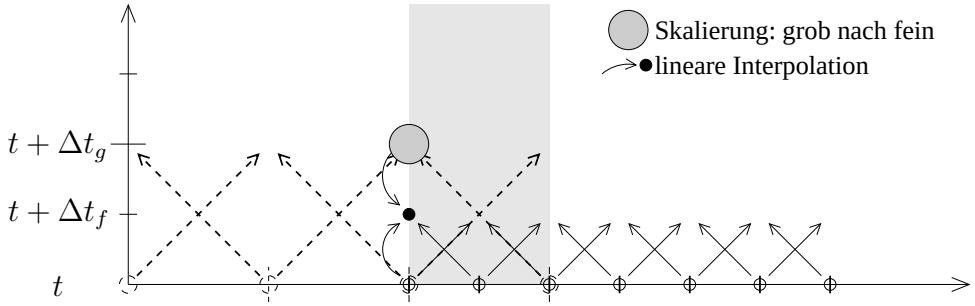

Abbildung 6.10: Skalierung und Interpolation der feinen Verteilungen für den Zwischenschritt

Somit sind alle Verteilungen der Fluid-Knoten des feinen Gitters zum Zeitpunkt $t + \Delta t_f$ gültig. Um das zeitliche Niveau des groben Takts zu erreichen, muss das feine Gitter erneut takten, also Kollision und Propagation ausführen (vgl. Abb. 6.11).

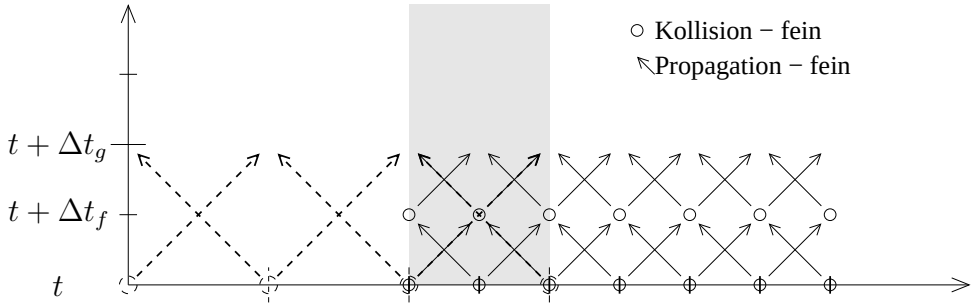

Abbildung 6.11: Kollision und Propagation des feinen Gitteranteils

Jetzt sind zwar alle Verteilungen auf richtigem zeitlichen und räumlichen Niveau, es tritt jedoch wieder das Problem auf, dass das feine Gitter am feinen Interface und das grobe Gitter am groben Interface durch die Propagation nicht vollständig bedient wurden. Am feinen Interface wird ähnlich, wie in Abbildung 6.10 dargestellt, vorgegangen, mit dem Unterschied, dass die Verteilungen nur

noch skaliert und nicht mehr zeitlich interpoliert werden müssen. Am groben Interface werden die Verteilungen von fein nach grob skaliert. Wegen des bereits vorhandenen gleichen zeitlichen Niveaus ist hier ebenfalls keine zeitliche Interpolation erforderlich.

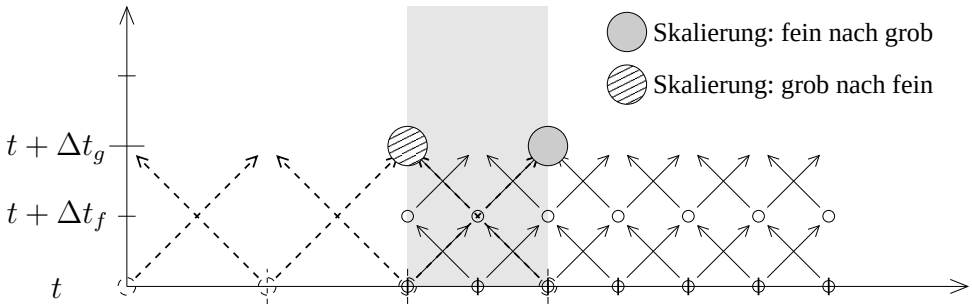

Abbildung 6.12: Skalierung der Verteilungen von fein nach grob

Der *volle Takt* ist nun vollendet, womit ein neuer Zyklus gestartet werden kann.

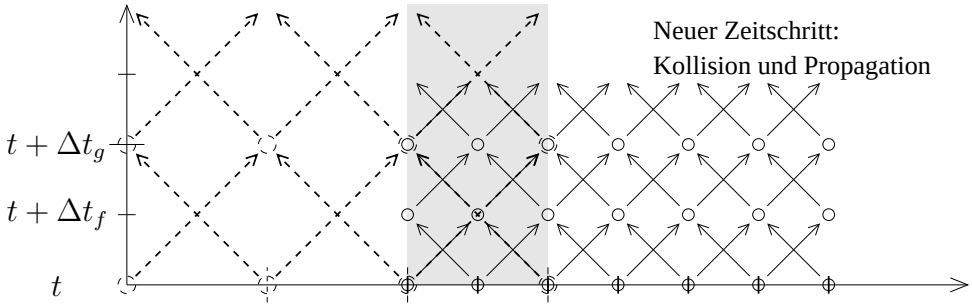

Abbildung 6.13: Beginn eines neuen *groben Takts*

6.5 Momentenmethode für die multiskalen LB-Methode

Die Multiskalen Momentenmethode (MMM) unterscheidet sich vom entsprechenden BGK-Ansatz im Wesentlichen nur in der Kollision. Wie in Kapitel 5.9 beschrieben, ist eine Transformation der Verteilungen in den Momentenraum und zurück in den Geschwindigkeitsraum notwendig. Während in der BGK-Relaxation alle Relaxationsparameter $s_{bgk,l} = \frac{\Delta t_l}{\tau_l}$ für eine bestimmte Gitterauflösung gleich sind und nur von Δt_l und ν abhängen, können im D2Q9-Gitter mit der Momentenmethode vier verschiedene Relaxationsparameter $s_{i,l}$ gewählt werden, wobei

die mit den Spannungen korrelierenden Relaxationsparameter $s_{8,l}$ und $s_{9,l}$ identisch mit $s_{bgk,l}$ und durch die Wahl der kinematischen Viskosität für alle Gitter vorgegeben sind. Das Verhältnis von $s_{8,l}$ zu $s_{8,l+1}$ ist mithilfe von Gleichung 6.6 bestimmbar. Für ein makroskopisch homogenes Verhalten des Fluids müssen alle Verhältnisse $\frac{s_{i,l}}{s_{i,l+1}}$ der Auflösungen l und $l+1$ gleich sein:

$$\frac{s_{2,l}}{s_{2,l+1}} = \frac{s_{3,l}}{s_{3,l+1}} = \frac{s_{5,l}}{s_{5,l+1}} = \frac{s_{8,l}}{s_{8,l+1}} \tag{6.26}$$

Wegen der eineindeutigen linearen Abbildungsvorschrift zwischen dem Geschwindigkeitsraum und dem Momentenraum, sind die Vorgänge Skalierung, räumliche und zeitliche Interpolation im Geschwindigkeitsraum und im Momentenraum äquivalent. Folgende Umformungen sollen dies veranschaulichen:

$$f_{i,g} = f_{i,f}^{(0)} + scale_{f\to g} \cdot f_{i,f}^{(1)} \tag{6.27}$$

Auf beiden Seiten wird nun die Transformationsvorschrift M angewendet und der Skalierungsfaktor ausgeklammert:

$$Mf_{i,g} = Mf_{i,f}^{(0)} + scale_{f\to g} \cdot (Mf_{i,f}^{(1)}) = m_{i,f}^{(0)} + scale_{f\to g} \cdot m_{i,f}^{(1)} \tag{6.28}$$

Die Skalierung ebenso wie alle anderen Interpolationen können nun im Momentenraum durchgeführt werden und anschließend mit der inversen Abbildungsvorschrift M^{-1} zurück in den Geschwindigkeitsraum transformiert werden:

$$M^{-1}Mf_{i,g} = M^{-1}m_{i,g} = f_{i,g} \quad \text{(q.e.d.)} \tag{6.29}$$

Der Basisalgorithmus ist im Wesentlichen identisch mit dem der LBGK-Multiskalen Simulation. Lediglich die Kollision findet im Momentenraum statt.

6.6 Diskussion: Grenzen des Verfahrens

In diesem Abschnitt wird das Augenmerk besonders auf die Stabilität und den Gültigkeitsbereich von Multiskalen Lattice-Boltzmann Simulationen gelegt. Die hier diskutierten Eigenschaften werden vor allem in dem Kontext von Simulationen auf nicht-uniformen Gittern betrachtet, gelten aber prinzipiell auch für Simulationen auf äquidistanten Gittern.

6.6.1 Diskretisierungsfehler

Bei Multiskalen Lattice-Boltzmann Simulation sind den eingeführten Fehlern,

- Diskretisierungsfehler: $\mathcal{O}(\Delta t^2)$
- Knudsen-Fehler: $\mathcal{O}(\epsilon^2)$

- Kompressibilitäts-Fehler: $\mathcal{O}(Ma^2)$

wie sie in [102] dargestellt sind, besondere Beachtung zu schenken, da diese durch die unterschiedlichen Auflösungen verschiedene Größenordnungen annehmen und somit anisotropes Verhalten implizieren können. Der Betrag des makroskopischen Geschwindigkeitsvektors sollte daher stets kleiner als $0.1\frac{\Delta x}{\Delta t}$ sein.

Die Interpolationsverfahren, insbesondere die räumliche Interpolation, führen zudem Fehler ein, die das Ergebnis spürbar beeinflussen können (vgl. Kapitel 8.7). Eine Erhöhung der Auflösung ist oftmals der einzige Ausweg, Unstetigkeiten der Strömungsvariablen im Bereich des Interfaces zu glätten. Eine ausführliche Diskussion hierüber wird in Kapitel 8 geführt.

6.6.2 Kinematische Viskosität versus maximale Gitterlevel

In diesem Abschnitt wird die zulässige Anzahl der Gitterebenen abgeschätzt. Es sei ausdrücklich darauf hingewiesen, dass diese Abschätzung nicht allgemeingültig ist, da die Grenzen der zu verwendenden Relaxationparameter strömungsabhängig sind.

Die Theorie bedingt für hydrodynamisch sinnvolle Simulationsergebnisse folgende Grenzen:

$$1.0 \leq s_l < 2.0 \tag{6.30}$$

Je näher die Relaxationparameter an der Grenze 2.0 sind, um so instabiler wird die Simulation, während numerische Stabiltät für $s_{i,l} = 1.0$ zu erwarten ist. Bei Anwendung des BGK-Kollisionsoperators sollte der Relaxationsparameter den Wert 1.9 nicht wesentlich übersteigen[3]. Auf einem äquidistanten Gitter entspricht dies einer kinematischen Viskosität von etwa $10^{-2}\frac{\Delta x^2}{\Delta t}$. Strömungen mit geringen Gradienten verkraften weitaus geringere Viskositäten.

Unter der Annahme, dass auf dem gröbsten Gitter der Auflösung l_0 der Zeitschritt Δt_{l_0} mit 1 angenommen wird, kann die Beziehung

$$\Delta t_l = \left(\frac{1}{2}\right)^{l-l_0} \tag{6.31}$$

zwischen der Anzahl der Gitterebenen und dem Zeitschritt angegeben werden. Diese Beziehung wird in die Formel 6.4 eingesetzt:

$$s_l = \frac{(\frac{1}{2})^{l-l_0}}{3\nu + \frac{(\frac{1}{2})^{l-l_0}}{2}} \tag{6.32}$$

[3]Diese Grenze dient den folgenden Erläuterungen und kann meist höher angenommen werden.

Weiterhin müssen für die Relaxationparameter Stabilitätsgrenzen s_{min} und s_{max} angenommen werden. Somit erhält man einen funktionalen Zusammenhang zwischen der Gitterebene l und der kinematischen Viskosität ν (l_0 wird im weiteren Verlauf mit 1 angenommen):

$$s_{min} \Rightarrow l_{max} = 1.0 - 1.44 \cdot ln\left(-6\frac{\nu \cdot s_{min}}{s_{min} - 2}\right) \tag{6.33}$$

$$s_{max} \Rightarrow l_{min} = 1.0 - 1.44 \cdot ln\left(-6\frac{\nu \cdot s_{max}}{s_{max} - 2}\right) \tag{6.34}$$

mit

$$l \geq 1 \tag{6.35}$$

Dies führt zur Formulierung einer Vorschrift zur Bestimmung der maximal zulässigen Anzahl von Gitterebenen l_{zul}:

$$l_{zul} = l_{max} - l_{min} + 1 = 1 - 1.44 \cdot ln\left(\frac{s_{min}(s_{max} - 2)}{s_{max}(s_{min} - 2)}\right) \tag{6.36}$$

Interessanterweise hängt die Anzahl der möglichen Gitterebenen scheinbar nicht von der kinematischen Viskosität ab. Da die Relaxationsparameter s_{min} nicht kleiner als 1.0 werden sollten, ist hier eine implizite Abhängigkeit über die Wahl des Zeitschritts Δt_l gegeben, die in Abbildung 6.14 veranschaulicht wird. Auf der Abzisse ist die kinematische Viskosität logarithmisch aufgetragen. Die Ordinate gibt die Gitterebene an. Jene Wertepaare, die im Bereich oberhalb der s_{min}-Kurve liegen, befinden sich außerhalb des hydrodynamisch gültigen Bereichs und jene Wertepaare, die unterhalb der s_{max}-Kurve liegen, befinden sich in einem Bereich, der wahrscheinlich zu numerischen Instabilitäten führt. Die Anzahl der verwendbaren Gitterebenen l_{zul} für eine bestimmte kinematische Viskosität ν kann nun grafisch bestimmt werden. Sie ergibt sich als Differenz der Ordinate l_2 der s_{min}-Kurve und der Ordinate l_1 der s_{max}-Kurve, wobei l_1 mindestens eins sein muss. Wie in Abbildung 6.14 zu sehen ist, ist das Maximum des Ordinaten-Intervalls für die BGK-Approximation ab $\nu \approx 0.01\frac{\Delta x^2}{\Delta t}$ gegeben (Ablesung a). Die Differenzen der Ordinate ergeben sich demnach zu

$$\Delta l \approx 4 \Rightarrow l_{zul} \approx 5 \tag{6.37}$$

Für $0.01\frac{\Delta x^2}{\Delta t} < \nu < \frac{1}{6}\frac{\Delta x^2}{\Delta t}$ reduziert sich die Anzahl der möglichen Gitterebenen (vgl. Kurve c in Abb. 6.14). Theoretisch sind die Multiskalen LBGK Simulation auf fünf verschiedene Gitterebenen unter Annahme der oben genannten Stabilitätsbedingungen beschränkt. Dies kann mit der Formel 6.36 bestätigt werden:

$$l_{zul} = 1 - 1.44 \cdot ln\left(\frac{1 \cdot (1.9 - 2)}{1.9 \cdot (1 - 2)}\right) = 5.24 \approx 5 \tag{6.38}$$

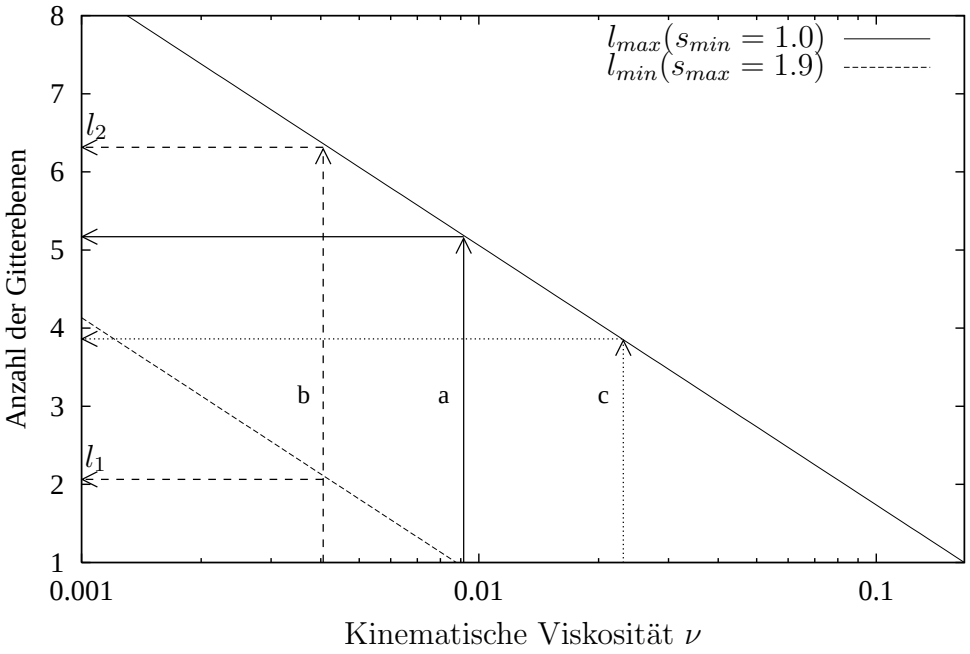

Abbildung 6.14: Anzahl der Gitterebenen in Abhängigkeit von der kinematischen Viskosität

Die diskutierten Zusammenhänge gelten ebenfalls für die auf der Momentenmethode basierten Multiskalen Lattice-Boltzmann Verfahren, solange die Viskosität maximal gewählt wird, da sich dann die Momentenmethode zum LBGK-Verfahren reduziert. Deutlich komplexere Verhältnisse ergeben sich, wenn die Relaxationszahlen s_8 und s_9 auf dem feinsten Gitter größer als Eins sind und somit die Viskosität nicht maximal angenommen wurde. In naiver Beschreibung kann die höhere Stabilität der Momentenmethode damit begründet werden, dass die Relaxationsparameter für die Momente spezifisch bestimmt werden dürfen. Die Relaxationsparameter für die Spannungen sind durch das viskose Verhalten der Strömung, also durch die kinematische Viskosität gegeben. Alle anderen Parameter dürfen im Wesentlichen frei gewählt werden. Man befindet sich im gültigen Bereich, solange Gleichung (6.30) erfüllt ist. Hier gilt jedoch, dass Relaxationsparameter, die sich der Stabilitätsgrenze von Zwei nähern, zu einer Destabilisierung der Relaxation der entsprechenden Momente führen. Vorteile kann man sich durch die Momentenmethode dann erhoffen, wenn die Relaxationsparameter s_2, s_3 und s_5 kleiner als s_8 gewählt werden können, denn dann können die korrespondierenden Momente, die keinen oder einen nur sehr geringen Einfluss auf die Navier-Stokes Lösung haben, stabilisierend auf die Simulation wirken.

Im Folgenden wird eine formale Beziehung zwischen der Viskosität ν, der Git-

terebene l und den frei wählbaren Relaxationsparameteren hergeleitet. Zunächst wird gefordert, dass die freien Relaxationsparameter s_{frei} minimal sind

$$s_{frei,l_{max}} = 1 \tag{6.39}$$

Die Verhältnisse aller Relaxationszahlen zwischen zwei Gitterebenen müssen konstant sein (Gleichung (6.26)). Einsetzen von Gleichung (6.32) in Gleichung (6.26) und Auflösen nach $s_{frei,l_{min}}$ des gröbsten Gitters führt zu:

$$s_{frei,l_{min}} = s_{frei,l} \cdot \frac{s_{8,l_{min}}}{s_{8,l}} = 1 \cdot \frac{3\nu + (\frac{1}{2})^l}{(3\nu + \frac{1}{2}) \cdot (\frac{1}{2})^{l-1}} \tag{6.40}$$

Zur weiteren Veranschaulichung werden in Abbildung 6.15 für verschiedene Viskositäten die maximalen, auf das gröbste Gitter bezogenen, freien Relaxationzahlen $s_{frei,l_{min}}$ aufgetragen. Die ν_1-Kurve erreicht bereits bei der fünften Gitterebene

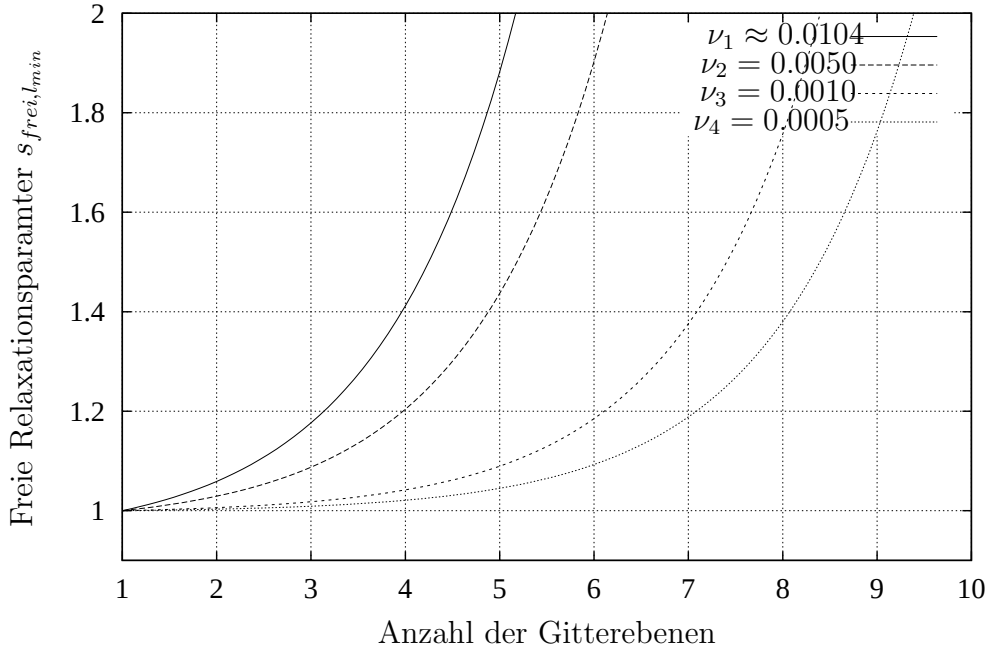

Abbildung 6.15: Freie Relaxationszahlen der jeweils gröbsten Auflösung in Abhängigkeit von der Viskosität und dem Gitterlevel

die als kritisch angenommene Grenze von 1.9. Die Viskosität wurde hierfür maximal bezüglich eines Gitters bestehend aus fünf Ebenen gewählt. Daher geht diese Simulation über in ein LBGK-Verfahren. Die anderen Kurven der freien Relaxationszahlen ν_2 bis ν_4 erreichen erst bei höheren Gitterebenen die kritischen

Bereiche der Relaxationszahlen. Die entsprechenden spannungsbezogenen Relaxationszahlen lauten:

Kurve 1	Kurve 2	Kurve3	Kurve 4
$\nu = \frac{1}{96}$	$\nu = 0.005$	$\nu = 0.001$	$\nu = 0.0005$
$s_{8,l=1} = 1.8823$	$s_{8,l=1} = 1.9417$	$s_{8,l=1} = 1.9881$	$s_{8,l=1} = 1.9988$
$s_{frei,l=5} = 1.8823$	$s_{frei,l=5} = 1.4369$	$s_{frei,l=5} = 1.0894$	$s_{frei,l=5} = 1.0449$

Tabelle 6.1: Vergleich der Relaxationparameter; l=1 entspricht der feinsten Auflösung; l=5 entspricht der am fünft-feinsten aufgelösten Gitterebene

In der vierten Zeile der Tabelle 6.1 sind die minimalen, freien Relaxationszahlen der fünften Gitterebene angegeben. Es wird deutlich, dass diese vor allem dann relativ kleiner werden, wenn die Viskosität geringer gewählt wird, als für diese Anzahl von Gitterebenen notwendig wären. Die Verwendung der Multiskalen Momentenmethode wird sich aus den genannten theoretischen Gründen nur bei Wahl von sehr geringen Viskositäten als vorteilhaft erweisen. Eine maximale zulässige Anzahl von Gitterebenen kann nicht angegeben werden. Es kann allerdings davon ausgegangen werden, dass unter den eben beschriebenen Bedingungen die Stabilität erhöht wird.

6.7 Zusammenfassung

Um mit nicht-uniformen Gittern rechnen zu können und trotzdem den expliziten Algorithmus, bestehend aus Kollision und Propagation, beizubehalten, muss dieser um die Schritte

- Skalierungen zwischen den Phasenräumen,
- räumliche und
- zeitliche Interpolationen

erweitert werden. Dies stellt natürlich einen gewissen *Mehraufwand* dar, welcher jedoch vertretbar ist, da das Interface und somit der Mehraufwand immer um eine Dimension geringer ist, als das Strömungsgebiet.

Während die Skalierungen mathematisch fundiert ableitbar sind, handelt es sich bei den anderen Interpolationsschemata um heuristische Vorgehensweisen. Hier liegt auch die Hauptquelle der numerischen Viskosität, die unter bestimmten Simulationsbedingungen eingeführt werden kann (vgl. Kapitel 8.7.1).

Bei geeigneter Wahl der Viskosität sollten mindestens fünf verschiedene Gitterebenen verwendbar sein. Dies reduziert lokal den Rechenaufwand auf dem gröbsten Gitter um bis zu 2 Größenordnungen für zweidimensionale Gebiete und um bis zu 3 Größenordnungen für dreidimensionale Gebiete, bezogen auf die feinste Auflösung.

Kapitel 7

Prototyp eines Multiskalen LB-Strömungslösers (2D)

Zu den Bestandteilen eines numerischen Berechnungskerns für Strömungssimulationen gehört die Diskretisierung der mathematischen Modelle, die das Verhalten des Fluids beschreiben, und die Modellierung und Diskretisierung der Randbedingungen. Da Teile des Preprocessings und des Postprocessings oft ganz entscheidend vom numerischen Lösungsmodell abhängen, ist es sinnvoll diese Teile möglichst nah am Berechnungskern zu implementieren.

In diesem Kapitel werden sowohl die Theorie als auch die entsprechenden Konzepte und Implementierungsdetails der Bausteine eines prototypischen Simulators für zweidimensionale Strömungen beschrieben. Die Ausführungen konzentrieren sich weniger auf den Simulationszyklus, da dieser im vorhergehenden Kapitel ausführlich dargestellt wurde, sondern auf die Programmteile, die die Datenstruktur für den Simulationszyklus (Preprocessing) aufbauen ebenso wie die Programmteile, die die erhaltenen Simulationsergebnisse auswerten. Die Schwerpunkte werden weiter auf die Datenbasis des Berechnungsgitters, die Baumdatenstrukturen, und auf die Sonderfälle, die sich durch die Nichtuniformität des Gitters ergeben, gelegt. Das Kapitel wird durch eine ausführliche Untersuchung der Implementierung an ausgewählten Strömungen abgeschlossen.

7.1 Konzept

7.1.1 Vorhandene LB-Codes

Seit dem Beginn der Erforschung der Lattice-Boltzmann Methoden wurden neben dem bisher einzigen bekannten kommerziellen LB-Code (PowerFLOW von Exa) hauptsächlich Forschungscodes entwickelt. Der Fokus lag hierbei auf der Untersuchung und Validierung verschiedener Strömungsphänome mit Lattice-Boltzmann Verfahren und weniger auf einer effizienten und gleichzeitig flexiblen Implemen-

tierung. Die dem Berechnungsgitter zugrunde liegende Datenstruktur wurde oftmals mit mehrdimensionalen Feldern, z. B. $Array[Dim_x][Dim_y][9]$ für das D2Q9-Modell, realisiert. Die Nachteile werden ersichtlich, wenn die Geometrie des zu simulierenden Fluids sich schlecht von rechteckigen (2D) oder quaderförmigen (3D) Feldern überlagern lässt oder geringe Porösität vorliegt (vgl. Abb. 7.1). In

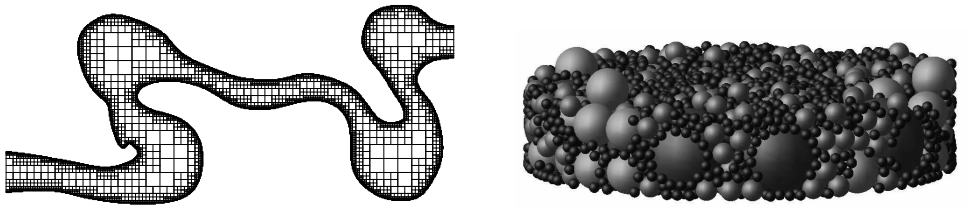

Abbildung 7.1: Ausschnitt eines Mäanders (links) und Kugelpackung (rechts)[35]

den Arbeiten von Krafczyk [59], Schulz [92] und Freudiger [34] wurden effizientere Datenstrukturen entwickelt und in Lattice-Boltzmann Codes integriert, sodass durchströmte geometrische Objekte beliebiger Form und Porösität mit maximaler Knotenupdaterate[1] simuliert werden können. Im Wesentlichen werden die Verteilungen der Gitterknoten auf 1D-Felder abgebildet, sodass die Datenlokalität verbessert werden kann. Mit weiteren Hilfsfeldern wird die Propagation gesteuert. In diesen Arbeiten werden jedoch nur äquidistante Gitter verwendet, wodurch Freiheitsgrade nicht Problem-angepasst verteilt werden können. Wie bereits in Kapitel 6.1 erwähnt, existieren bereits Codes für Simulationen auf nichtuniformen Gittern, die auf blockstrukturierten Ansätzen basieren. Üblicherweise werden zwei oder mehr Matrix-ähnliche Datenfelder überlagert und die räumliche Zugehörigkeit durch Abbildungsvorschriften der Feldindizes erzielt. Die oben genannten Nachteile, komplexe Hindernisobjekte und große Porösität, sind mit dieser Technik und diesen einfachen Datenstrukturen nicht zu lösen.

Die meisten Codes sind daher wertvoll für wissenschaftliche Zwecke, da sie Erkenntnisse über spezielle Phänome aufdecken. Gleichzeitig sind sie für praktische Zwecke oft untauglich, da sie den ingenieurrelevanten Anforderungen, wie z. B. eine automatische, lokale, geometrisch angepasste Verfeinerung eines Motorinnenraums, nicht nachkommen können.

Eine wesentliche Aufgabe besteht darin, ein Konzept aufzustellen, das zum einem Verfeinerungsstrategien unterstützt und zum anderem den heutigen Anforderungen, wie sie in Kapitel 2 dargelegt wurden, entspricht. In diesem Fall ist Wert darauf zu legen, für die Numerik der Multiskalen-Simulation eine geeignete Datenbasis zu konstruieren. Unter Verwendung von objektorientierten Ansätzen wird nicht nur die Datenbasis entworfen, sondern der Simulationsprozess selbst

[1] Die Knotenupdaterate wird durch die Anzahl der berechneten Zeitschritte für einen Gitterknoten pro Sekunde Rechenzeit definiert.

modelliert. Jüngste Arbeiten haben gezeigt (z. B. Dupuis [26]), dass eine Kombination aus numerischer Berechnung und Objektorientierung sehr effizient sein kann.

7.1.2 Basismodell des Prototypen

In Abbildung 7.2 werden die wichtigsten Bestandteile eines Klassenmodells des prototypischen Strömungssimulators dargestellt. Deren Inhalte und Klassenfunktionalität werden nun kurz angesprochen. Die Einbindung der Klassen in das

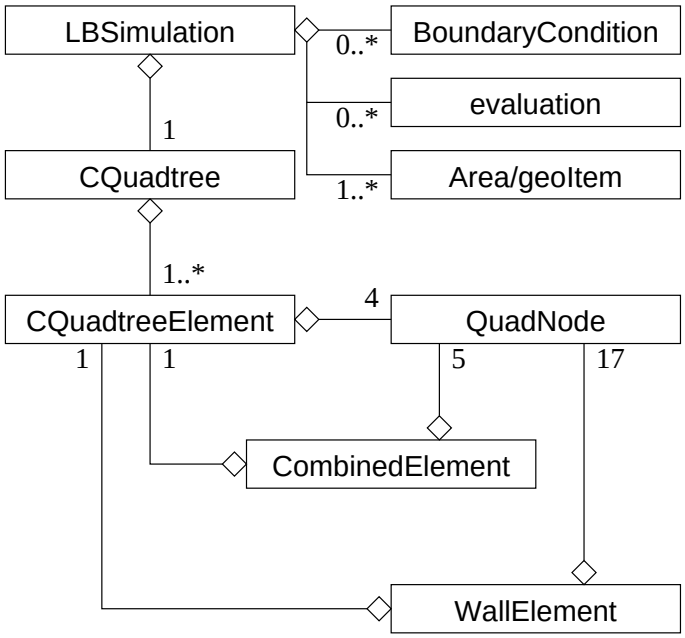

Abbildung 7.2: Quadtree-basierter LB-Simulator: Klassenkonzept

Gesamtkonzept soll im Verlauf diesen Kapitels schrittweise entwickelt werden.

LBSimulation: Für den Simulationsablauf ist die Klasse *LBSimulation* von zentraler Bedeutung. Sie steuert die Abfolge, beginnend mit der Initialisierung, in der große Teile des verfahrensspezifischen Preprocessings abgearbeitet werden, bis hin zur Simulationszeitschleife, in der neben der numerischen Berechnung der Strömung die Evaluierung vorgenommen wird. Gleichzeitig fungiert sie als Container für alle physikalischen Parameter, Listen für die Skalierungen und Interpolationen, etc.

CQuadtree: Die Klasse *CQuadtree* ist als Organisationsstruktur des Gitters zu verstehen. Neben den für den Quadtree erforderlichen Attributen, enthält sie die Methoden zur Erzeugung der Baumdatenstruktur (vgl. Kapitel 4.4.2).

CQuadtreeElement: Die Objekte der Klasse *CQuadtreeElement* dienen der Abbildung der Gitterzellen. Die horizontale Vernetzung der Elemente erlaubt ein effizientes Auffinden von räumlichen Beziehungen der Elemente zueinander (vgl. Kapitel 4.4.3). Die Eckknoten (QuadNode) stellen die Gitterknoten dar.

QuadNode: In den Gitterknoten werden die verschiedenen Verteilungssets gespeichert.

Area/geoItem: Die Ableitungen der abstrakten Klasse *geoItem* sind reine Geometriebeschreibungen der Hindernisse. Objekte der Klasse *Area* enthalten neben genau einem Objekt der Klasse geoItem zusätzliche, für die Gittergenerierung wichtige Informationen, wie z. B. die Baumtiefe.

CombinedElement: In der Klasse *CombinedElement* werden die vier Knotenobjekte, die für die kubische Rauminterpolation der Verteilungen an den hängenden Knoten benötigt werden, sowie ein Knotenobjekt (hängender Knoten), dessen Verteilungen gesucht werden, gehalten.

WallElement: Die Objekte der Klasse *WallElement* enthalten je einen Verweis auf eine Gitterzelle, die eine Schnittmenge mit einem als *Solid* markierten, geometrischen Objekt aufweist, und einem Gitterknoten, der Eckpunkt der Zelle ist und sich innerhalb des Solids befindet. Insgesamt müssen Zeiger auf 17 Knotenobjekte gehalten werden. Ein Zeiger ergibt sich, wie eben beschrieben, aus dem Solid-Knoten, die weiteren 16 Zeiger ergeben sich aus den potentiell acht Nachbarknoten im Abstand Δx und $2\Delta x$. Mit einem weiteren Verweis auf das geometrische Objekt können alle geometrischen Beziehungen bestimmt werden (vgl. Abb. 7.17).

evaluation: Diese Klasse ist abstrakt und soll hier stellvertretend für alle (abgeleiteten) Klassen stehen, die Auswertungen vornehmen.

BoundaryConditionSet: Die Instanz dieser Klasse definiert Ein- und Ausströmbedingungen.

7.2 Gittergenerierung

Die Gittergenerierung, deren wesentlicher Bestandteile die Erzeugung des Quadtrees ist, wurde schon in Kapitel 4.4.2 ausführlich diskutiert. Aus Gründen der Datenhaltung (vgl. Kapitel 7.3.1) und der Übersichtlichkeit (vgl. Kapitel 7.3.2) muss der Glättungsvorgang erweitert werden. Die Glättung, in der gefordert wird, dass sich benachbarte Gitterzellen um nicht mehr als einen Gitterlevel unterscheiden dürfen, wird dahingehend erweitert, daß sich die Interfaces nicht überlappen und nicht berühren dürfen. In der Implementierung wird daher erzwungen, dass

die feinen Interfaces einen Abstand von mindestens *drei* groben Gitterzellen aufweisen müssen.

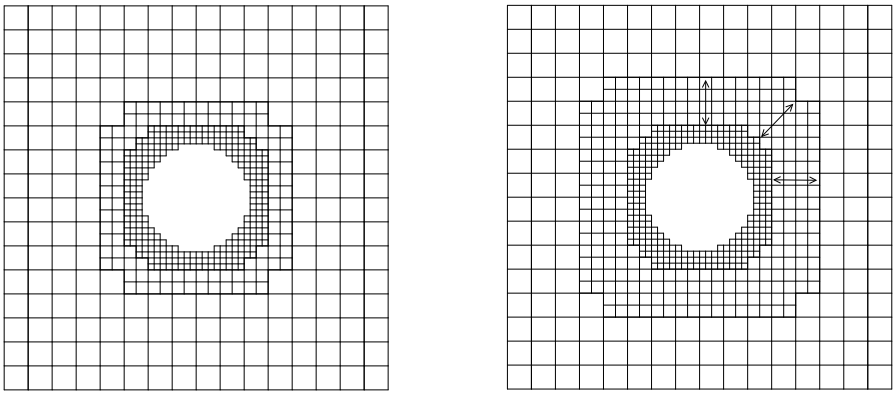

Abbildung 7.3: Gitter ohne (links) und mit (rechts) modifizierter Glättung

Diese neue Glättung wird im weiteren Verlauf als *modifizierte Glättung* bezeichnet. Im Anschluss werden offensichtlich ungünstige Konstellationen im Gitter im Rahmen einer *empirischen Gitterkorrektur* gesucht und durch eine erzwungene Verfeinerung beseitigt. Im Vordergrund steht hier die *Gesamtlänge* des Interfaces zu reduzieren sowie die Form des Interfaces zu glätten.

Regel 1: Wenn sich eine grobe Gitterzelle zwischen feiner aufgelösten Gitterzellen befindet, wird diese mit Regel 1 verfeinert (vgl. Abb. 7.4).

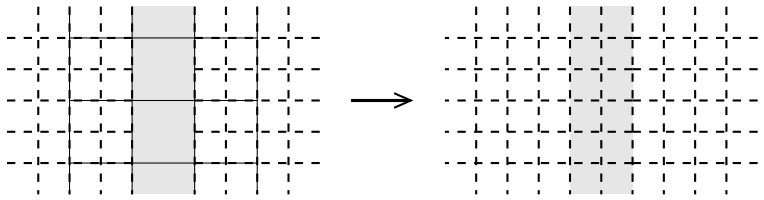

Abbildung 7.4: Gitterkorrektur nach Regel 1

Der Rechenaufwand der Simulationsphase, der sich durch eine Zwangsverfeinerung und der damit einhergehenden Zunahme der Gitterknoten zusätzlich ergibt, wird hier nur unwesentlich erhöht. Zudem kann davon ausgegangen werden, dass die Qualität der Berechnung lokal zunimmt, da nicht skaliert und interpoliert werden muss.

Regel 2: Wenn sich zwei feine Gitterbereiche berühren, wird mit Regel 2 versucht den Kontaktbereich flächenmäßig zu vergrößern, damit die Dichte der lokalen Skalierungen und Interpolationen abnimmt.

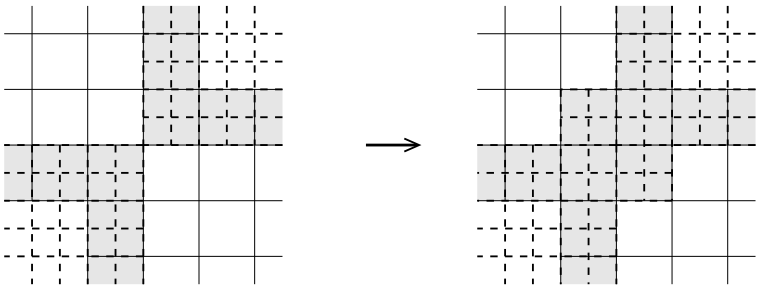

Abbildung 7.5: Gitterkorrektur nach Regel 2

7.3 Initialisierung der Simulation

Nachdem das Gitter für die Belange der Simulation erweitert wurde, müssen weitere Initialisierungen vorgenommen werden. Dieser Abschnitt beschränkt sich jedoch auf jene Initialisierungen, die für das Verständnis des Programmablaufs und des entsprechenden Codes von Bedeutung sind.

7.3.1 Datenhaltung der Primärvariablen

Der Speicher für die diskreten Verteilungssets wird in der Knotenklasse *QuadNode* organisiert. Damit die Verteilungen während der Propagation nicht überschrieben werden, stellt diese prototypische Implementierung einen zweiten Satz von Verteilungen pro Gitterauflösung zur Verfügung. Im Bereich des Interfaces wird sogar für vier Verteilungssets Speicher vorgehalten. Die Notwendigkeit hierfür ist gleichzeitig die Begründung für die Forderung der modifizierten Glättung und wird mit Abbildung 7.6 motiviert.

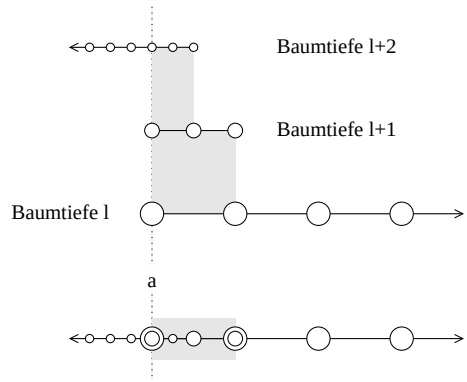

Abbildung 7.6: Datenhaltung: Geglättetes 1D-Gitter

Würde man auf einen Mindestabstand der Interfaces verzichten, müssten theoretisch beliebig viele Verteilungssets der angrenzenden Gitterebenen vorgehalten werden können (Linie a in Abbildung 7.6). Das entsprechende Gitter mit modifizierter Glättung ist in Abbildung 7.7 dargestellt.

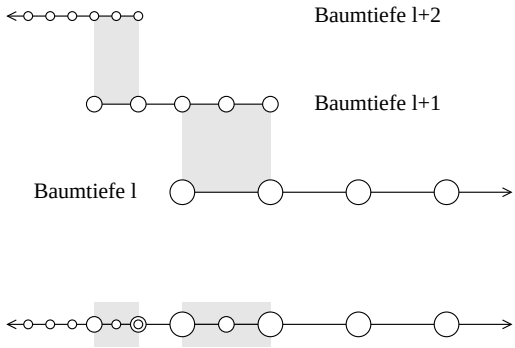

Abbildung 7.7: Datenhaltung - modifiziert geglättetes 1D-Gitter

Es ist zu beobachten, dass sich maximal zwei Knoten benachbarter Gitter mit unterschiedlicher Auflösung überlagern.

7.3.2 Klassifizierung der Gitterknoten

Mithilfe der Baumdatenstrukturen können, im Gegensatz zu den blockstrukturierten Ansätzen, prinzipiell beliebig geformte Gitterkonfigurationen erzeugt werden. Die Anzahl von verschiedenen Nachbarschaftsverhältnissen der Gitterzellen wurde durch die modifizierte Glättung deutlich reduziert. Die Aufgabe der Klassifizierung besteht in der Identifizierung und Markierung von Knoten, die in einer bestimmten Konstellation im Bereich des Interfaces zu anderen Gitterknoten stehen. Diese Knoten haben in der Regel Verteilungssets für das angrenzende grobe

und feine Gitter und werden im Simulationsablauf aufgrund ihrer Klassifizierung gesondert behandelt.

Zunächst bezieht sich die Klassifizierung auf *Fluid*-Knoten (vgl. Abb. 7.8). Im Wesentlichen werden die Interfaceknoten in zwei Typen unterschieden:

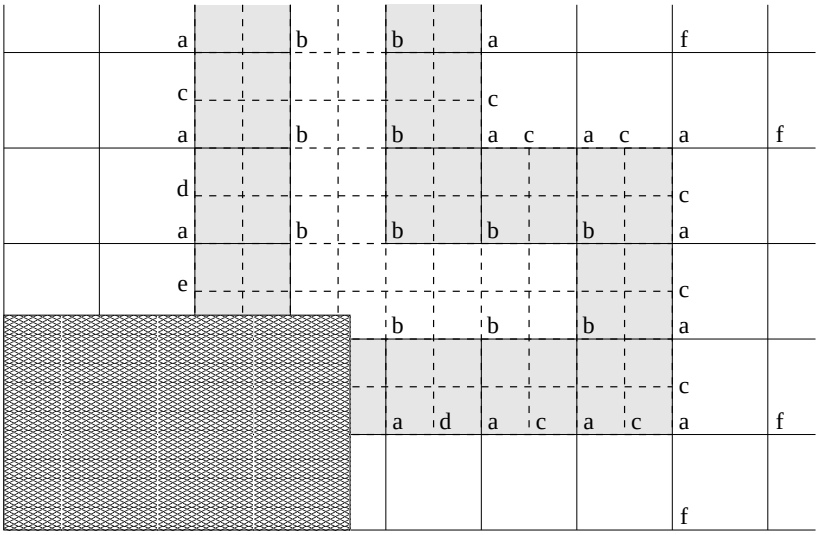

Abbildung 7.8: Beispiel: Klassifizierung für Fluid-Knoten

- Hängende Knoten (c, d, e)
- Nicht-hängende Knoten (a, b, f)

Die hängenden Knoten werden weiter in Knoten differenziert, die sich

- nah (d) oder
- sehr nah (e)

an Hindernis-Objekten befinden. Als *nah* (d) wird ein Fluid-Knoten bezeichnet, wenn sich ein oder zwei weitere Fluid-Knoten zwischen dem Fluid-Knoten und der Wand befinden. *Sehr nahe* (e) Fluid-Knoten befinden sich unmittelbar an der Wand. Die nicht-hängenden Knoten sind immer Knoten, die auch Teil des groben Gitters sind und befinden sich entweder auf dem feinen Interface (a) oder auf dem groben Interface (b). Weiter spielt für die Algorithmen die Orientierung, die in die Himmelsrichtungen Nord, Ost, Süd und West eingeteilt werden, eine wichtige Rolle.

Es ergeben sich die Aufgabenbereiche der verschiedenen Sorten Knoten (a) bis (f):

(a): Feines Interface, Skalierung von grob nach fein, Zeitinterpolation.

(b): Grobes Interface, Skalierung von fein nach grob.

(c): Kubische, eindimensionale Rauminterpolation.

(d): Quadratische, eindimensionale Rauminterpolation.

(e): Kombination: Eindimensionale, lineare Interpolation und Extrapolation (vgl. Kapitel 7.4.2.2).

(f): Skalierung von grob nach fein, um Verteilungen für die kubische Rauminterpolation zu berechnen.

In der Implementierung werden diese Knoten entsprechend der Aufgaben in Listen zur schnellen Abarbeitung vorgehalten.

7.3.3 Gitterabhängige Größen

Für alle Gitterebenen werden vor dem eigentlichen Simulationsdurchlauf alle Größen, die vom lokalen Zeitschritt oder von der Gitterweite abhängen, initialisiert.

Zeitschritte und Gitterpunktabstände: Das Bezugssystem ist das gröbste Gitter l_0. Hier werden Δt_{l_0} und Δx_{l_0} mit

$$\Delta t_{l_0} = \Delta x_{l_0} = 1$$

definiert. Für alle anderen Gitterebenen ($l > l_0$) werden Δt_l und Δx_l mit Formel 6.31 berechnet.

Relaxationsparameter: Die Relaxationsparameter werden in Abhängigkeit von der Gitterebene gemäß Gleichung (6.26) und Gleichung (6.32) unter Beachtung der Gültigkeitsgrenzen (vgl. Kapitel 6.6) bestimmt.

Skalierungsfaktoren: Die Skalierungsfaktoren werden mit den Gleichungen (6.17) und (6.18) berechnet.

Da alle Größen in 1D-Feldern gehalten werden, kann mit der entsprechenden Baumtiefe der Gitterebene der richtige Wert sofort selektiert werden.

7.3.4 Anfangsbedingungen

Die Verteilungen aller Fluid-Knoten müssen vor Beginn der eigentlichen Simulationsphase vorgegeben werden. Hat man keine Kenntnis über einen optimalen makroskopischen Anfangszustand des Fluids, wird für das gesamte Strömungsgebiet eine Dichte (z. B. $\rho = 1.0$) und die Nullgeschwindigkeit ($\vec{u} = 0$) angenommen.

Die Verteilungen werden mit der Gleichung (5.47) berechnet. Somit wird die Divergenzfreiheit

$$\nabla \cdot \vec{u} = 0 \tag{7.1}$$

und die Poisson-Gleichung für den Druck

$$\Delta p = -\rho \nabla \cdot (\vec{u} \cdot \nabla \vec{u}) \tag{7.2}$$

erfüllt.

Für wenige Strömungsphänomene (Taylor-Wirbel) sind die analytischen, makroskopischen Anfangsbedingungen bekannt. In diesem Fall verbessert man die Güte der Initialisierung, wenn neben der Berechnung der Gleichgewichtsverteilungen, die Nichtgleichgewichtsverteilungen berücksichtigt werden. Diese lassen sich durch zeitliches und räumliches Ableiten der Gleichgewichtsverteilungen[2] ermitteln [102]:

$$f_i = f_i^{(0)} - \tau \left(\frac{\partial f_i}{\partial t} + \xi_{i\alpha} \frac{\partial f_i^{(0)}}{\partial x_\alpha} \right) \tag{7.3}$$

Alternativ besteht die Möglichkeit die Anfangswerte für die Nichtgleichgewichtsanteile der Verteilungen qualitativ aufzuwerten, indem für einige Zeitschritte ($\mathcal{O}(100)$) der normale Kollision-Propagations-Zyklus durchgeführt wird und gleichzeitig die makroskopischen Anfangsbedingungen permanent über den Kollisionsoperator aufgeprägt werden. Die Nichtgleichgewichtsverteilungen konvergieren zu einem Wert, der den lokalen Spannungszustand (vgl. Gleichung (5.43)) widerspiegelt.

7.3.5 Wandknoten

Auf die Analyse, das Design und die Realisierung der Wandknoten wird im Kapitel über die Behandlung der Haftrandbedingung (vgl. Kapitel 7.4.2.2) detailiert eingegangen.

7.4 Randbedingungen

Während die Anfangsbedingungen an jedem Ort $\vec{x}$ zur Zeit $t = 0$ die Systemvariablen definieren, ist die Aufgabe der Randbedingungen, zu jeder Zeit t an ausgewählten Orten, welche typischerweise die Berandung des Fluids anzeigen, makroskopische Zustände und Systemvariablen vorzugeben [37]. Randbedingungen können prinzipiell in die zwei Kategorien

- *geometrische* und

[2] Die Gleichgewichtsverteilungen können mit den makroskopischen Größen gemäß Gleichung (5.46) oder Gleichung (5.47) berechnet werden.

- *physikalische*

Randbedingungen eingeordnet werden. Alle Randbedingungen gleichen sich in der Eigenschaft der örtlichen Begrenztheit und können infolgedessen in deren Form und Lage durch Aggregation von geometrischen Objekten beschrieben werden. In der Implementierung können sich beide Kategorien grundlegend unterscheiden.

7.4.1 Geometrische Randbedingungen

Die Veränderung der Geometriebeschreibung eines Strömungsgebietes kann eingesetzt werden, um periodische oder symmetrische Strömungsphänomene zu modellieren. Das Ziel des Vorgehens besteht darin, die Datenstruktur so zu verändern, dass an den betreffenden Rändern stets gültige Verteilungen vorgegeben werden.

7.4.1.1 Periodische Randbedingungen

Periodische Randbedingungen zeichnen sich dadurch aus, dass sich das Strömungsproblem in periodischen Abschnitten exakt wiederholt. Typischerweise sind die Begrenzungen dieser Abschnitte parallel zu den Achsenrichtungen des Koordinatensystems. Wie in Abbildung 7.9 zu sehen ist, werden dem System jene Ver-

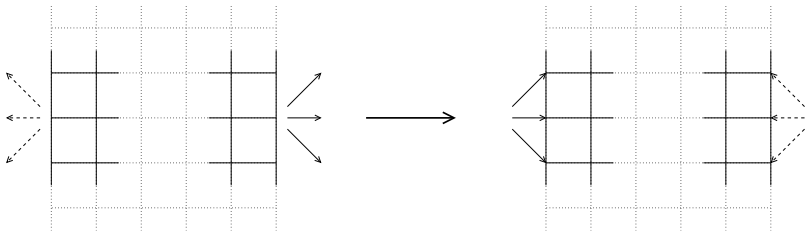

Abbildung 7.9: Horizontale, periodische Randbedingung vor (links) und nach (rechts) der Propagation

teilungen, die das System auf der einen Seite verlassen (linkes Gitter), auf der anderen Seite wieder zugeführt (rechtes Gitter). Die Realisierung erfolgt oft mithilfe von Kopiermatrizen, indem an den Grenzen des uniformen Gitters je eine zusätzliche Knotenspalte zur temporären Speicherung der Verteilungen der jeweils anderen Seite angeordnet wird. Dies erfordert weiteren Speicherbedarf. Bei der Verwendung von nicht-uniformen Gittern wird ein neues Konzept notwendig.

Durch die horizontale Verzeigerung der hybriden Netz-Baumdatenstruktur lässt sich eine Lösung implementieren, die keinen Mehraufwand impliziert und die das Problem der *Periodizität* logisch äquivalent umsetzt, indem die Nachbarschaftszeiger, die in periodischer Richtung keinen Nachbarn haben, auf das Element, das durch die Periodizität zum Nachbarelement werden soll, gesetzt

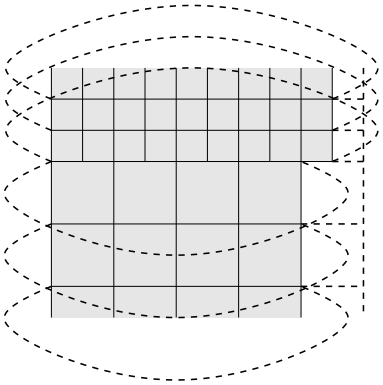

Abbildung 7.10: Horizontale, echt periodische Randbedingung

werden. In diesem Fall wird die Topologie der Geometrie entsprechend der Randbedingung angepasst. Aus der Sicht der Datenstruktur handelt es sich jetzt um ein echt periodisches System. Die Elemente, deren Nachbarschaftszeiger *modifiziert* wurden, sind in der Abbildung 7.10 gestrichelt dargestellt. Der Propagationsschritt muss keinerlei Sonderfälle beachten.

7.4.1.2 Symmetrische Randbedingungen

Wenn a priori feststeht, dass die Lösung eines stationären Strömungsproblems symmetrisch ist, genügt es, nur eine Hälfte des Gebietes zu berechnen. Diese Randbedingung ist identisch mit der *Rutschbedingung* und wird in Kapitel 7.4.2.3 erläutert.

7.4.2 Physikalische Randbedingungen

Bei den physikalischen Randbedingungen werden die makroskopischen Größen selbst (Dirichlet-Randbedingung) oder deren Normalableitungen zum Rand (Neumann-Randbedingung) vorgegeben. Da im Allgemeinen weniger makroskopische Größen als Momente bekannt sind, ist eine eindeutige Abbildung nicht möglich und die Verteilungen müssen approximiert werden. Im Gegensatz zu den topologischen Randbedingungen wird hier die Kontinuität der Ordnung des Verfahrens nicht zwingend erhalten. Zum einem spielt die Auflösung der der Randbedingung zugeordneten Geometrie und zum anderem der Grad der Näherung der vorgegebenen Verteilungen eine Rolle. Die Einbeziehung der Nichtgleichgewichtsverteilungen kann zwar das Konvergenzverhalten verbessern, kann aber unter Umständen die Stabilität reduzieren. Zudem sind für die Definition von Randbedingungen weitere Bedingungen zu beachten. Die Massenerhaltung muss gewährleistet sein, d. h. Masse, die dem System zugeführt wird, muss auch wieder abgeführt werden. Dies spielt vor allem dann eine Rolle, wenn an allen Rändern

Dirichlet-Geschwindigkeits-Randbedingungen verwendet werden. Es gilt:

$$\int u_\alpha n_\alpha \, ds = 0 \tag{7.4}$$

Weiterhin dürfen Geschwindigkeit und Druck nur dann gleichzeitig vorgegeben werden, wenn die Lösung des Problems bekannt ist.

7.4.2.1 Einlass- und Auslass-Randbedingungen

Zu den Einlass- und Auslass-Randbedingungen gehören die Einström-, Ausström- und Druckbedingungen. Alle Erklärungen beziehen sich auf die am linken Gebietsrand definierten Einströmbedingungen der Abbildung 7.11. Folgendes Vor-

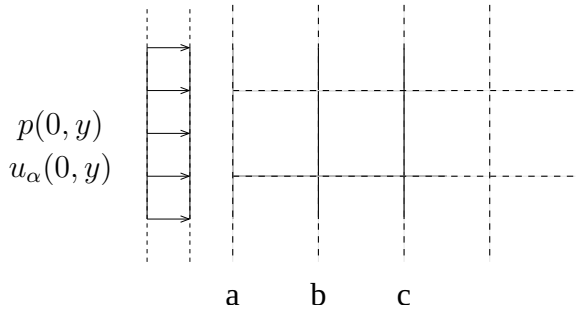

Abbildung 7.11: Einströmbedingungen

gehen wird vorgeschlagen:

1. Die bekannte Größe

   - Druck: $p(0, y) = p_0$ oder
   - Geschwindigkeitsprofil: $u_\alpha(0, y) = u_{0\alpha}(0, y)$

   wird in Spalte (a) vorgegeben.

2. Die zur Berechnung der Gleichgewichtsverteilungen notwendige korrespondierende Größe, Geschwindigkeit oder Druck, wird durch konstante (Spalte (b)) oder durch lineare (Spalte (b) und (c)) Extrapolation bestimmt.

3. Um auch am Rand ein Verfahren zweiter Ordnung zu erhalten, müssen die Spannungen durch Bilden der räumlichen (und zeitlichen) Ableitungen der Geschwindigkeiten in Spalte (a) erzeugt werden und die erste Näherung der Nichtgleichgewichtsanteile berechnet werden (Spannungstensor) (Gleichung (5.43)).

7.4.2.2 Haftbedingung

Die Haftrandbedingung wird an allen Körperoberflächen, die nicht mit einer anderen Randbedingung belegt sind, angetroffen. Um die Haftung des Fluids am Körper zu modellieren, werden die Geschwindigkeitskomponenten mit 0 angenommen:

$$u_\alpha = 0 \tag{7.5}$$

Bounce-Back-Schema: Da auch hier nicht alle makroskopischen Größen bekannt sind, ist eine eindeutige Bestimmung der Verteilungen nicht möglich. Eine bewährte alternative Modellierung basiert auf dem sogenannten *Bounce-Back*-Schema. Die Idee hierbei ist, dass Verteilungen, die gegen die Wand prallen (vgl. Abb. 7.12), reflektiert werden und somit der Impuls im zeitlichen Mittel Null ist. Die Methode zeichnet sich vor allem durch die Einfachheit der Implementie-

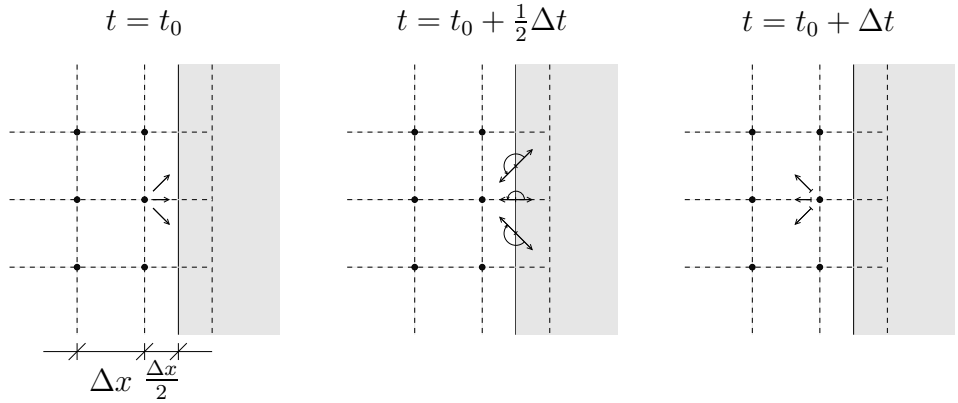

Abbildung 7.12: Klassisches Bounce-Back-Schema

rung aus. Für den Fall, dass die Position der Wand genau im Abstand $\frac{\Delta x}{2}$ vom ersten Fluid-Knoten liegt und somit die Verteilungen während der Propagation und der Reflexion exakt die Strecke Δx zurücklegen und wieder am Startpunkt ankommen, wird die Konvergenzordnung des Verfahrens durch die Randbedingungen nicht reduziert. Gekrümmte Ränder können mit dieser Methode jedoch nicht genau approximiert werden, womit für beliebig geformte Geometrien diese Randbedingung zu einer Konvergenz erster Ordnung degeneriert.

Boundary-Fitting Methode: Die Problematik der Modellierung und der Implementierung der Haftbedingung für beliebig geformete Geometrien wurde in einigen Arbeiten diskutiert und gründlich analysiert [28, 71]. In [7] wird ein auf dem Bounce-Back-Schema basiertes Verfahren, welches nur auf Interpolationsschemata zur verbesserten Approximation der wahren Position der Wand zurückgreift und deshalb numerisch robuste und stabile Eigenschaften besitzt, vorgestellt. Der

Grundgedanke besteht darin, die Startposition der Verteilungen, die an der Wand reflektiert werden, so zu bestimmen, dass sie nach der Propagation und Reflexion auf dem ersten wandnahen Fluid-Knoten ankommen. Entscheidend ist der Abstand q der Wand zum ersten Fluid-Knoten, der entweder *kleiner* oder *größer gleich* 0.5 ist.

$q < 0.5$: Dies ist der triviale Fall. Der Startpunkt[3] S der Verteilung f befindet sich zwischen den Punkten $F1$ und $F2$ (vgl. Abb. 7.13). Die Berechnung

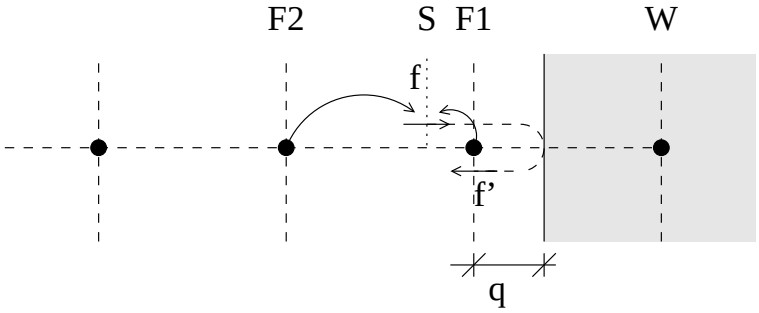

Abbildung 7.13: Boundary Fitting: $q < 0.5$

der reflektierten Verteilungen $f^{'}$ erfolgt *vor* der Propagation mittels linearer Interpolation[4], die in Abhängigkeit vom Abstand q formuliert wird:

$$f^{'}(t+1, F1) = 2 \cdot q \cdot f(t, F1) + (1 - 2 \cdot q) \cdot f(t, F2) \tag{7.6}$$

$q \geq 0.5$: Für diesen Fall existiert kein Startpunkt zwischen $F1$ und $F2$, sodass die Verteilung f nach der Reflexion wieder bei $F1$ ankommt (vgl. Abb. 7.14). Anstelle dessen wird zunächst die Propagation durchgeführt, womit die reflektierte Verteilung $f^{'}$ zwischen dem ersten Fluid-Knoten $F1$ und dem Wand-Knoten W am Ort Z zum Liegen kommt. Die Interpolation wird nun nach der Propagation mit der nach rechts zeigenden Verteilung f am Punkt $F2$ und der bereits reflektierten Verteilung f' am Punkt Z durchgeführt:

$$f^{'}(t+1, F1) = \frac{1}{2 \cdot q} \cdot f(t+1, Z) + \frac{2 \cdot q - 1}{2 \cdot q} \cdot f(t+1, F2) \tag{7.7}$$

Für eine effiziente Implementierung ist eine zeitliche Differenzierung der Durchführung der Interpolation in Abhängigkeit vom Abstand q ungünstig. Da die Werte der Verteilungen durch die Propagation nicht verändert werden, können die gleichen Verteilungen auch *vor* der Propagation verwendet

[3]Gedachter Punkt, an dem sich eine Verteilung vor der Propagation befindet.

[4]Denkbar ist auch die Verwendung höherer Polynominterpolationen. Eine erkennbare und vor allem effizientere Lösung ist allerdings nicht zu erwarten.

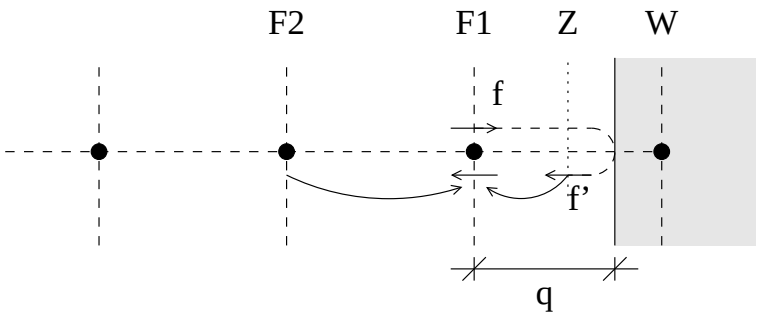

Abbildung 7.14: Boundary Fitting: $q \geq 0.5$

werden. Diese befinden sich dann am Punkt $F1$. Die zeitlich vorgezogene Interpolationsformel ist vollständig lokal und lautet:

$$f'(t+1, F1) = \frac{1}{2 \cdot q} \cdot f(t, F1) + \frac{2 \cdot q - 1}{2 \cdot q} \cdot f'(t, F1) \tag{7.8}$$

Mit der Boundary-Fitting Methode wird der wahre Verlauf der Strömung in Wandnähe und damit die Simulationsvariablen nun angenähert beschrieben. Wegen der fehlenden konservativen Eigenschaften des Interpolationsschemas, ist eine Massen- und Impulserhaltung nicht vollständig gegeben. Mit zunehmender Auflösung der Hindernisse nimmt die lokale Krümmung des Strömungsfeldes in Wandnähe ab [90], womit die eingeführten Fehler akzeptabel klein gehalten werden können. Diese Methode reduziert sich zum klassischen Bounce-Back-Schema, wenn alle Abstände q exakt 0.5 sind.

Alternativ werden in anderen Arbeiten [14, 84] volumenorientierte Algorithmen vorgestellt, mit denen ebenfalls die räumliche Genauigkeit zweiter Ordnung beibehalten wird.

Nicht-uniforme Ränder: Eine Herausforderung für das numerische Konzept und die Datenstruktur stellen nicht-uniform aufgelöste Ränder dar. Vor allem die hängenden Knoten, aber auch das überlappende Interface bedürfen einer genaueren Diskussion.

In der Abbildung 7.15 sind zwei mögliche Gitterkonstellation dargestellt, in denen die hängenden Knoten H_r bzw. H_l mittelbar bzw. unmittelbar in Wandnähe sind. Während rechts im Bild der hängende Knoten H_r von Fluidknoten umschlossen ist (mittelbar) und die fehlenden Verteilungen mittels quadratischer Interpolation bestimmt werden können, wird links im Bild zur Rekonstruktion der Verteilungen des Knoten H_l wegen des nicht vorhandenen Fluid-Knotens eine Extrapolation notwendig. Simulationsversuche haben gezeigt, dass diese Extrapolation lokale Spannungsspitzen erzeugt und somit numerische Viskosität spürbar einführt. Daher wird eine Untersuchung des Strömungsverhaltens in Wandnähe

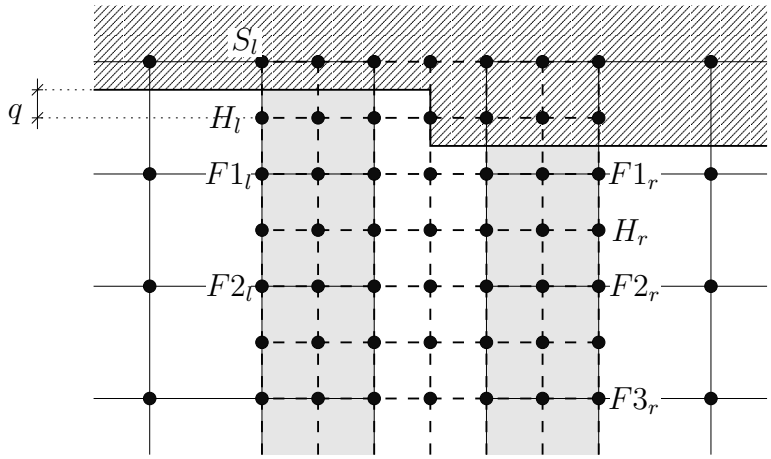

Abbildung 7.15: Hängende Knoten mittelbar und unmittelbar in Wandnähe

notwendig:
Zwischen dem Geschwindigkeitsfeld und den Spannungen besteht für inkompressible Strömungen der folgende Zusammenhang:

$$S_{\alpha\beta} = \mu\left(\frac{\partial u_\alpha}{\partial x_\beta} + \frac{\partial u_\beta}{\partial x_\alpha}\right) \tag{7.9}$$

Die Spannungen verhalten sich wie die räumlichen Ableitungen der Geschwindigkeiten. Die Modellbildung erfolgt mithilfe der Poiseuille-Strömung, deren Geschwindigkeits- und Spannungsprofil in Abbildung 7.16 dargestellt ist.

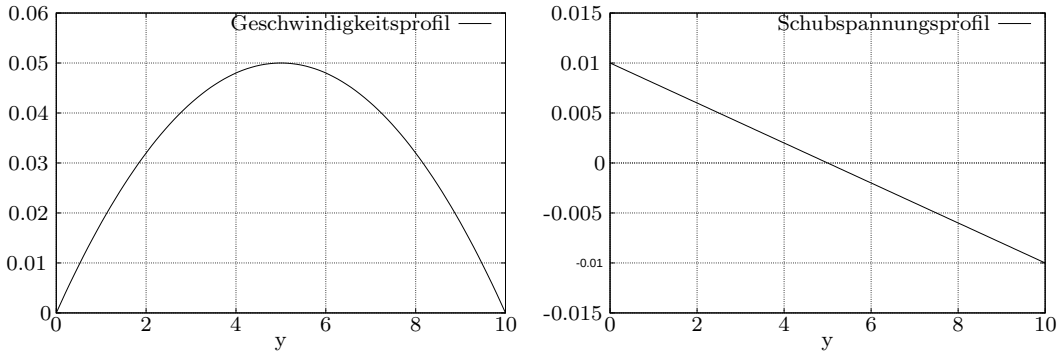

Abbildung 7.16: Poiseuille-Strömung: Geschwindigkeits- (links) und Schubspannungsprofil (rechts)

Im Inneren eines Strömungsfeldes ist die Aufgabe des Rekonstruktionsverfahrens, einen möglichst kontinuierlichen und harmonischen Verlauf der makroskopischen Größen nachzubilden. In Wandnähe zeigen Spannungen und Geschwindigkeit komplementäres Verhalten, weil die Geschwindigkeit verschwinden und die

Spannungen nicht. Der hier vorgestellte Ansatz basiert auf der Idee, die Rekonstruktion der verschiedenen makroskopischen Größen angepasst durchzuführen. Folgendes Vorgehen wird vorgeschlagen:

1. Dichte:
   Die Dichte auf der *Wandoberfläche* ist durch die Haftrandbedingung nicht definiert. Am hängenden Knoten H_l muss die Dichte durch lineare (oder konstante) Extrapolation bestimmt werden:
   $$\rho(H_l) = \frac{3}{2}\rho(F1_l) - \frac{1}{2}\rho(F2_l) \tag{7.10}$$

2. Geschwindigkeit:
   An der Oberfläche von Hindernissen verschwindet die Geschwindigkeit per Definition. Der wahre Abstand q der Wand vom ersten Fluid-Knoten ist bekannt. Somit kann die Geschwindigkeit am hängenden Knoten H_l linear interpoliert werden:
   $$u_\alpha(H_l) = \frac{q}{1+q}u_\alpha(F1_l) \tag{7.11}$$
   Mit den Annahmen für die Dichte und die Geschwindigkeit können nun die Gleichgewichtsverteilungen bestimmt werden (vgl. Kapitel 5.8.2).

3. Spannungen:
   Die Spannungen an der Wand sind unbekannt. Unter der Annahme, dass sich die Krümmung des Geschwindigkeitprofils in Wandnähe nahezu konstant ist, bietet sich eine Extrapolation der Spannungen an. Im Lattice-Boltzmann Schema sind diese linear von den Nichtgleichgewichtsanteilen abhängig. Eine lineare Extrapolation der Nichtgleichgewichtsanteile bietet sich an:
   $$f_i^{(1)}(H_l) = \frac{3}{2}f_i^{(1)}(F1_l) - \frac{1}{2}f_i^{(1)}(F2_l) \tag{7.12}$$

Die Rekonstruktion wird abgeschlossen, indem die Gleichgewichts- und Nichtgleichgewichtsverteilungen addiert werden. Dieses gemischte Schema, bestehend aus Interpolationen und Extrapolationen, zeigt deutlich glattere Ergebnisse als ein reines Extrapolationsschema.

Konzept der Haftbedingungen: Um das oben beschriebene Verfahren anzuwenden, muss die Datenstruktur erweitert werden. Für jeden Link[5], der einen Fluid-Knoten mit einem Solid-Knoten verbindet, ist der entsprechende Abstand q ebenso wie jene Fluid-Knoten, die in Interpolationsrichtung für das Boundary-Fitting-Schema benötigt werden, zu speichern. Hierfür wird die Klasse *WallElement* eingeführt, die folgende Attribute enthält (vgl. Abb. 7.17):

[5]Mit *Link* wird der Weg bezeichnet, den eine Verteilung während der Propagation zurücklegt.

- Ein Objekt der Klasse *QuadNode*:
  Dies ist der Solid-Knoten. Die interpolierten Verteilungen werden auf dem Speicher der *normalen* Verteilungssets abgelegt und während der Propagation auf den entsprechenden Fluid-Knoten kopiert.

- Ein 1D-Array für die Abstände q:
  Jeder Solid-Knoten kann im D2Q9-Modell theoretisch acht benachbarte Fluid-Knoten besitzen. Daher wird ein Feld für acht Abstandswerte q vorgehalten.

- Ein Objekt der Klasse *CQuadtreeElement*:
  Da von der Klasse *QuadNode* keine Verzeigerung zu den Elementen existiert, ist für die räumliche Zuordnung der Knoten dieser Verweis auf die Baumdatenstruktur notwendig. Dieses Element schneidet mindestens ein geometrisches Hindernisobjekt.

- Die Position des Knotens im Element:
  Von den vier möglichen Eckpunkten muss die richtige Ecke aus Gründen der räumlichen Zuordnung der Gitterknoten spezifiziert werden.

- Eine ID-Nummer:
  Die ID ist eindeutig und gibt die Möglichkeit, auf jenes Geometrie-Objekt zuzugreifen, welches das CQuadtreeElement schneidet.

- Ein 2D-Array für die Zeiger auf die Interpolations-Knoten:
  Ein 2×9-Array für zwei Interpolations-Knoten und neun Richtungen wird für jeden Solid-Knoten vorgehalten.

- Eine *Link*-Variable:
  Mittels einer Bit-Codierung wird festgelegt, welche Links vorhanden sind.

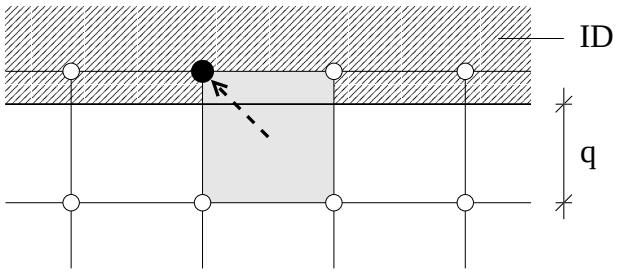

Abbildung 7.17: Implementierung: Modellierung eines *WallElements*

Aus Gründen der Effizienz sollten alle Attribute nur einmal in der Initialisierungsphase bestimmt werden. Für die entsprechenden Berechnungen sind Member-Funktionen vorgesehen.

Dieses Konzept hält auch den Anforderung für Gitterübergänge an Rändern stand. In Abbildung 7.18 wird zur Verdeutlichung das grobe vom feinen Gitter getrennt, um die vorhandenen Links besser darzustellen. Da die feinen Knoten auf

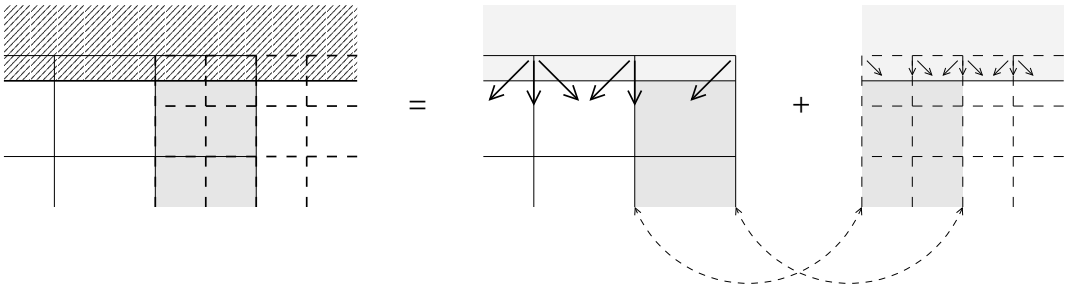

Abbildung 7.18: Reflektierte Verteilungen von Wand-Knoten am Interface

dem feinen Interface und die groben Knoten auf dem groben Interface rekonstruiert werden, muss auf dem jeweiligen Link, deren Zielknoten rekonstruiert wird, kein modifiziertes Bounce-Back durchgeführt werden. Das vorgestellte Konzept ist somit konsistent.

7.4.2.3 Rutschbedingung

Die Rutschbedingung, auch *Slip*-Randbedingung genannt, zeichnet sich dadurch aus, dass die Geschwindigkeitskomponenten senkrecht zur Wand ebenso wie die Ableitung der Geschwindigkeitskompononten parallel zur Wand in Normalenrichtung Null sind. Typischerweise wird diese Randbedingung eingesetzt, um symmetrische System zu modellieren oder um eine unendliche Ausdehnung des Strömungsgebietes nachzubilden. Im Lattice-Boltzmann Kontext lassen sich diese Randbedingungen sehr einfach implementieren, wenn die Berandung des der Randbedingung zugehörigen geometrischen Objekts parallel zu den Gitterachsen des Systems ist. Der algorithmische Ablauf funktioniert analog des Bounce-Back-Schemas. Auch hier werden die Solid-Knoten verwendet, um die Verteilungen für den Propagationsschritt vorzuhalten. Die Zuordnung der Ziel- und Quell-Knoten wird hierfür, ähnlich wie bei den *No-Slip*-Randbedingungen, mithilfe eines Filters realisiert. In Abbildung 7.19 ist die algorithmische Vorgehensweise illustriert:
Zunächst werden die Verteilungen vom Fluid-Knoten auf den orthogonalen Solid-Knoten kopiert (Abb. 7.19, links). Mithilfe des Filters werden dann die Verteilungen entsprechend der *Slip*-Bedingung ausgerichtet (Abb. 7.19, mitte) und stehen für den Propagationsschritt zur Verfügung (Abb. 7.19, rechts).

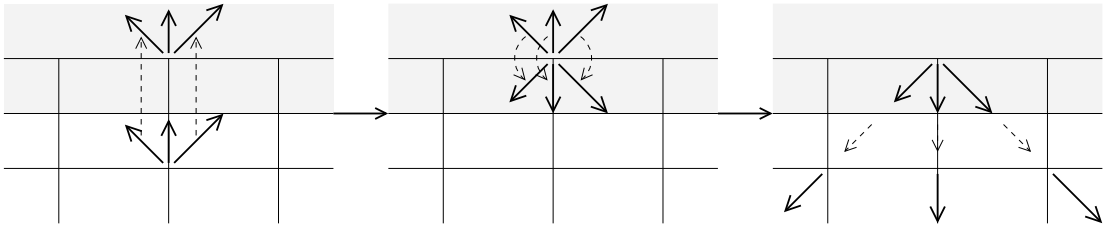

Abbildung 7.19: Ablauf der Slip-Randbedingung

7.5 Auswertung von Strömungen

Die Auswertung von Daten aus einer Strömungssimulation erfolgt durch Visualisierung zur qualitativen Beurteilung aber auch durch Evaluierung physikalischer Kenngrößen zur quantitativen Bewertung.

7.5.1 Evaluierungsgrößen

Im Folgenden werden die gängigen Evaluierungsgrößen erklärt und auf die Lattice-Boltzmann spezifischen Merkmale eingegangen.

7.5.1.1 Geschwindigkeits- und Druckfelder

Zur Visualisierung von Strömungen sind in adiabaten Strömungen an den diskreten Punkten des Rechengitters Geschwindigkeits- und Druckkomponenten anzugeben. Typischerweise werden die Datenfelder mittels Dateischnittstelle an das entsprechende Visualisierungswerkzeug weitergegeben. Für die Berechnung der Geschwindigkeitskomponenten wird Gleichung (5.42) und für die Berechnung des Druckes Gleichung (5.45) angewandt.

7.5.1.2 Druckdifferenzen

Oft werden Druckdifferenzen zur Analyse von Strömungsphänomenen berechnet, um beispielsweise Druckverluste aufzuzeigen. Um diese Druckdifferenzen auch für verschiedene Versuchsanordnungen vergleichbar zu machen, erhält man mittels einer Dimensionsanalyse folgende Formel zur Bestimmung des dimensionslosen Drucks $\widehat{p}$:

$$\widehat{p} = \frac{p}{\frac{1}{2} \cdot \rho \cdot \overline{U}^2} \tag{7.13}$$

Am Beispiel der Zylinderumströmung in Abbildung 7.20 wird die Druckdifferenz mithilfe zweier definierter Punkte A und B berechnet:

$$\Delta\widehat{p} = \frac{p(A) - p(B)}{\frac{1}{2} \cdot \rho \cdot \overline{U}^2} \tag{7.14}$$

7.5.1.3 Rezirkulationszone

Für viele Anwendungsfälle ist die Ausdehnung des Rezirkulationsgebietes von großer Bedeutung. Zunächst muss hierfür die Hauptstromrichtung eindeutig definiert werden. Bereiche, in denen auf die Hauptstromrichtung bezogen negative Geschwindigkeitskomponenten auftreten, befinden sich innerhalb des Nachlaufs eines Hindernisobjekts. Weiterhin muss ein Saatpunkt angegeben werden (z. B.

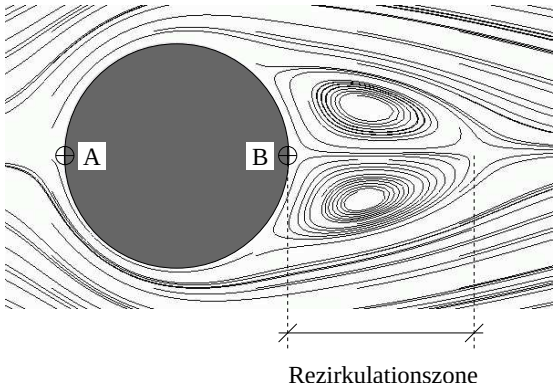

Abbildung 7.20: Rezirkulationszone einer Zylinderumströmung

in Abb. 7.20 die rechte Seite des Zylinders), von dem angenommen wird, dass er im Nachlaufgebiet liegt. Anschließend wird zellenweise nach rechts, rechts oben oder rechts unten gewandert, bis keine Gitterpunkte gefunden werden können, in denen die Geschwindigkeitskomponente der Hauptstromrichtung negativ ist. Der Abstand zum Bezugspunkt B stellt die Ausdehnung der Nachlaufzone dar.

7.5.1.4 Kraftbeiwerte

Die Kraftbeiwerte spielen vor allem in der Automobil- und der Flugzeugindustrie eine wichtige Rolle. Es ist zwischen dem Kraftbeiwert in ungestörter Anströmrichtung, dem Widerstandsbeiwert c_W, und dem Kraftbeiwert normal zur Strömrichtung, dem Auftriebsbeiwert c_A, zu unterscheiden[6]. Die Formel zur Berechnung der dimensionslosen Kennzahl lautet:

$$c_K = \frac{2F}{\rho \overline{U}^2 D} \tag{7.15}$$

F ist die Kraftkomponente, ρ die Fluid-Dichte, $\overline{U}$ eine Vergleichsgeschwindigkeit und D eine makroskopische Vergleichslänge. Typischerweise wird $\overline{U}$ in Bezug zur Anströmgeschwindigkeit (z. B. $\frac{2}{3}U_{max}$ am Einströmrand) und D in Bezug zu einer Abmessung des umströmten Körpers (z. B. Zylinderdurchmesser) gestellt.

[6]In der Literatur werden oft die englischen Abkürzungen c_D für den *drag coefficient* und c_L für den *lift coefficient* verwendet.

Die Berechnung der Kraft kann durch Integration des Drucktensors (vgl. Gleichung (5.4)) über die Berandung des Körpers erfolgen:

$$F_\alpha = \int_O (D_{\alpha\beta} \cdot n_{O,\alpha})\, dO \tag{7.16}$$

In [7, 72] wird berichtet, dass die Implementierung der Integrationsmethode einen enormen Aufwand darstellt. Dieser ist dadurch begründet, dass die Spannungen des den Körper umgebenden Strömungsfeldes auf die Oberfläche des Körpers projeziert werden müssen. Die Berandung muss gleichzeitig in diskreter Form vorliegen. Weiterhin müssen die Schnittfiguren zwischen Körperoberfläche und Fluidzellen bestimmt und im Speicher vorgehalten werden. Im gleichen Artikel wird eine neue, auf Impulsaustausch basierte Methode vorgestellt. Der Ansatz stützt sich auf den Gedanken, alle Teilimpulse, die beim *Aufprall* der Verteilungen auf das Hindernis im Propagationsschritt entstehen, aufzusummieren. Bezogen auf einen Einheitszeitschritt ist das genau die Kraft, die auf den Körper wirkt. Die qualitative und quantitative Übereinstimmung beider Berechnungsmethoden wurde in [72] untersucht und bestätigt.

Der Impuls einer Verteilung, der während der Propagation (Reflexion) mit einem Hindernis ausgetauscht wird, entspricht dem Impuls, den der Körper durch die eingehenden Verteilungen aufnimmt und dem Impuls, der durch die abgehenden Verteilungen vom Körper abgegeben wird.

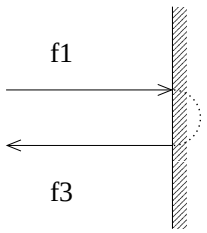

Abbildung 7.21: Impulsaustausch

Ein Teilimpuls P entlang eines Links i ergibt sich somit mit:

$$P = f_1 + f_3 = f_i + f_{\bar{i}} \quad \text{mit} \quad \vec{e_i} + \vec{e_{\bar{i}}} = \vec{0} \tag{7.17}$$

Die Gesamtkraft auf einen Körper ergibt sich als Summe aller Teilimpulse, die zwischen den Solid-Knoten x_b und deren benachbarten Fluid-Knoten ausgetauscht werden:

$$\vec{F} = \sum_{\vec{x_b}} \sum_{1 \neq 0} \vec{e_i} \left[f_i(\vec{x_b}, t) + f_{\bar{i}}(\vec{x_b} + \vec{e_i}\Delta x, t) \right] \cdot \Delta t \tag{7.18}$$

Die Berechnung erfolgt nach der Kollision und nach der *Boundary-Fitting*-Interpolation. Somit wird die *wahre* Position der Wand implizit berücksichtigt. Mithilfe der in Kapitel 7.4.2.2 beschriebenen Datenstruktur lässt sich diese Operation sehr gut durchführen.

7.5.1.5 Strouhal-Zahl

Oftmals werden in instationären Strömungen periodisch auftetende Strömungssequenzen, wie beispielsweise die alternierenden Wirbelablösungen der von-Karman-Straße, beobachtet. Die entsprechende Kenngröße ist die *Strouhal-Zahl* oder auch die *dimensionslose Frequenz* und wird wie folgt berechnet:

$$St = \frac{Df}{\overline{U}} \tag{7.19}$$

D entspricht wieder einer makroskopischen Vergleichslänge, f der Frequenz der Wirbelablösung und $\overline{U}$ einer Vergleichsgeschwindigkeit. Zur Bestimmung der Frequenz wird ein Signal benötigt, dass sich phasengleich mit der Strömung verändert. Hierfür wird typischerweise der Auftriebsbeiwert genommen. Es kann auch ein beliebig anderes frequenzabhängiges Signal, wie z. B. die y-Komponente des Geschwindigkeitsvektors an einem Ort $\vec{x}$, verwendet werden. Dieses Signal wird einer diskreten Fouriertransformation unterzogen, um die maßgebliche Frequenz zu erhalten.

7.5.2 Evaluierungskonzept

Die Mächtigkeit eines Simulationswerkzeugs zeigt sich nicht nur in seiner numerischen Stabilität und Effizienz des entsprechenden Lösers, sondern auch in seiner Flexibilität. Hierzu zählt auch die Handhabung der Auswertungslogik, welche sich in den Postprocessingmethoden, wie z. B. der Evaluierung des c_D-Wertes widerspiegelt. Mithilfe des objektorientierten Konzepts dieses Prototyps ist eine beliebige Anzahl und Kombination von Evaluierungen möglich. In Abbildung

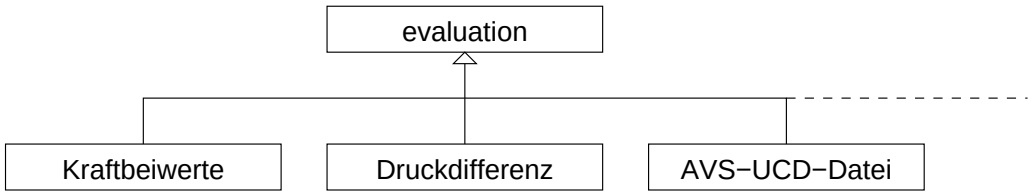

Abbildung 7.22: Evaluierung: Klassenkonzept

7.22 ist die flache Ableitungshierarchie des Evaluierungskonzepts dargestellt. Die abstrakte Klasse *evaluation* enthält Attribute, die sinnvoller Weise alle Evaluierungsklassen haben. Hervorzuheben ist das Zeitintervall, das angibt, in welchen

Abständen eine Evaluierung durchgeführt werden soll und der Zeitpunkt oder der Zeitschritt, an dem die nächste Evaluierung angesetzt ist. Das gemeinsame Interface wird durch die rein virtuellen Methoden *evaluate()* und *getNextEvaTime()* definiert. Die Funktion *getNextEvaTime()* gibt lediglich den nächsten Zeitschritt der Evaluierung zurück. Eine Erweiterung des Konzepts bezüglich Evaluierungen, die sich über ein Zeitintervall erstrecken, ist ohne weiteres möglich. Dies könnte beispielsweise bei Large-Eddy-Simulationen [59] erforderlich werden, um das Geschwindigkeitsfeld über definierte Zeitintervalle zu mitteln. In jeder abgeleiteten Klasse muss die virtuelle Funktion *evaluate* überladen und die im Kapitel 7.5.1 beschriebenen Auswertungen implementiert werden. Die Klasse *LBSimulation* enthält eine Template-Liste vom Typ *evaluation*, in der alle Objekte der Evaluierungen gehalten werden, womit diese alle in gleicher Weise aufgerufen werden können.

7.6 Verifizierung an ausgewählten Strömungen

Im Folgenden wird mithilfe von ausgewählten Strömungen gezeigt, dass das erweiterte Lattice-Boltzmann Verfahren und die Implementierungen gute Ergebnisse liefern. Sämtliche Visualisierungen wurden mit der Visualisierungsumgebung AVS/Express 6.0 durchgeführt [7] und nicht durch weitere Algorithmen modifiziert.

7.6.1 Poiseuille Strömung

In der ersten Untersuchung wird der Prototyp anhand der *Poiseuille Strömung* verifiziert. Es handelt sich hierbei um eine besonders einfache, stationäre Schichtenströmung mit dem großen Vorteil, dass eine analytische Lösung existiert.

Wenn die Strömung sich vollständig entwickelt hat, ist ihr Geschwindigkeitprofil in x-Richtung invariant. Ausgehend von einer laminaren, stationären und inkompressiblen Strömung wird von den Navier-Stokes Gleichungen folgende Gleichung für das Geschwindigkeitsprofil abgeleitet [74]:

$$u(y) = \frac{1}{2}\mu(h^2 - (y-h)^2) = \frac{1}{2}\rho\nu(h^2 - (y-h)^2)\frac{\Delta p}{L} \tag{7.20}$$

Der Maximalwert des Parabelprofils wird berechnet mit:

$$u_{max} = \frac{h^2}{2\rho\nu} \cdot \frac{\Delta p}{L} \tag{7.21}$$

Die Reynoldszahl ist definiert durch:

$$Re = \frac{u_{mittel} \cdot (2h)}{\nu} \quad \text{mit} \quad u_{mittel} = \frac{2}{3}u_{max} \tag{7.22}$$

Diese Gleichungen dienen den nachfolgenden Simulationen als Referenzlösungen.

[7] www.avs.com

7.6.1.1 Simulations-Setup

Für die Lattice-Boltzmann Simulation wird ein in x-Richtung periodisches Berechnungsgitter gewählt, um sämtliche Einflüsse, die durch die Einfluss- und Ausflussrandbedingungen entstehen könnten, zu eliminieren. Als treibende Kraft wird der Druckgradient durch eine Volumenkraft G ersetzt [43]. Die folgende Herleitung gilt jedoch nur für die Poiseuille-Strömung, da eine sehr einfache geometrische Berandung vorliegt und der Druckgradient konstant ist. Es gilt:

$$F_{\Delta p} = \Delta p \cdot L_y \quad \text{und} \quad F_G = \rho \cdot g \cdot L_y \cdot L_x \tag{7.23}$$

Die Volumenkraft G und die aus dem Druckgradienten resultierende Kraft muss gleich sein:

$$\Delta p \cdot L_y = \rho \cdot g \cdot L_y \cdot L_x \quad \Rightarrow \quad \frac{\Delta p}{L_x} = g \cdot \rho \tag{7.24}$$

Die volumenbezogene Kraft g wird nun ausgedrückt als Kraft $\widehat{G}$ bezüglich eines Kontrollvolumens Δx^2 und man erhält folgende Beziehung:

$$\frac{\Delta p}{L_x} = \frac{\widehat{G}}{\Delta x^2} \cdot \rho \quad \Leftrightarrow \quad \frac{\Delta p}{n_x \cdot \rho} = \frac{\widehat{G}}{\Delta x} \quad \text{mit} \quad L_x = n_x \cdot \Delta x \tag{7.25}$$

Der *Forcing*-Term muss für jedes Gitter entsprechend dem Quotienten $\frac{\widehat{G}}{\Delta x}$ skaliert werden. Die folgenden drei Gitter (vgl. Abb. 7.23-7.24) wurden für die Untersuchung konstruiert:

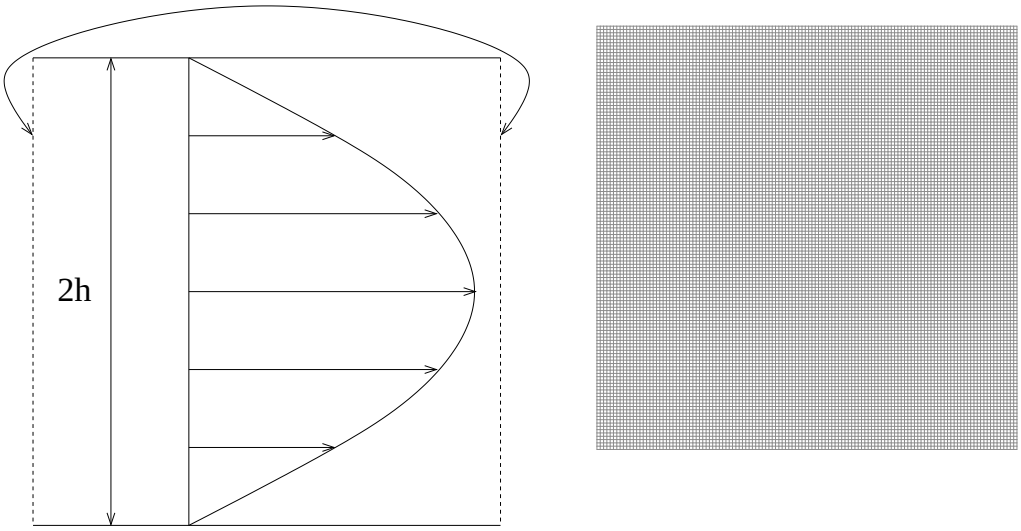

Abbildung 7.23: Poiseuille Strömung: Gitter-Setup (links) und Gitterkonfiguration 1 (rechts)

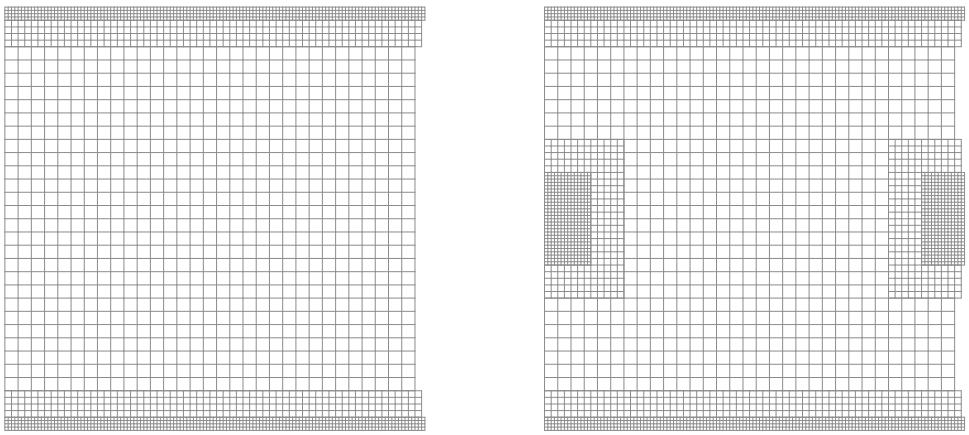

Abbildung 7.24: Poiseuille Strömung: Gitterkonfiguration 2 (links) und 3(rechts)

Eine Reynoldszahl von $Re = 50$ erfordert bei einem Kanaldurchmesser $d = 2h = 127$ eine kinematische Viskosität von $\nu = 0.05842$. Der Referenzwert der maximalen Geschwindigkeit beträgt $u_{max} = 0.0345054$.

7.6.1.2 Auswertung

Eine Konvergenz der drei Simulationen wurde angenommen, wenn die Abbruchbedingung[8] von $\epsilon = 10^{-9}$ unterschritten wurde. In der Abbildung 7.25 sind die analytische Lösung als durchgehende Linie und die diskreten Simulationswerte als Kreuze dargestellt. Auf die Grafik der uniformen Lösung wird verzichtet. Es ist zu beobachten, dass die Punkte der numerischen Lösung sehr gut mit der analytischen Lösung übereinstimmen. Zur Quantifizierung wird für alle drei Simulationen der relative Fehler der maximalen Geschwindigkeiten berechnet.

	Simulation 1	*Simulation 2*	*Simulation 3*
u_{max}	0.03450194	0.03450195	0.03450108
Relativer Fehler	0.00010027	0.00009998	0.00012519
Fluidknoten	16256	2336	3408
Knotenverhältnis	1	6.9589	4.7699

Tabelle 7.1: Poiseuille-Strömung: Auswertung

[8]Das Abbruchkriterium wird vereinfacht als Differenz der f_0-Verteilungen zwischen 500 Zeitschritten berechnet.

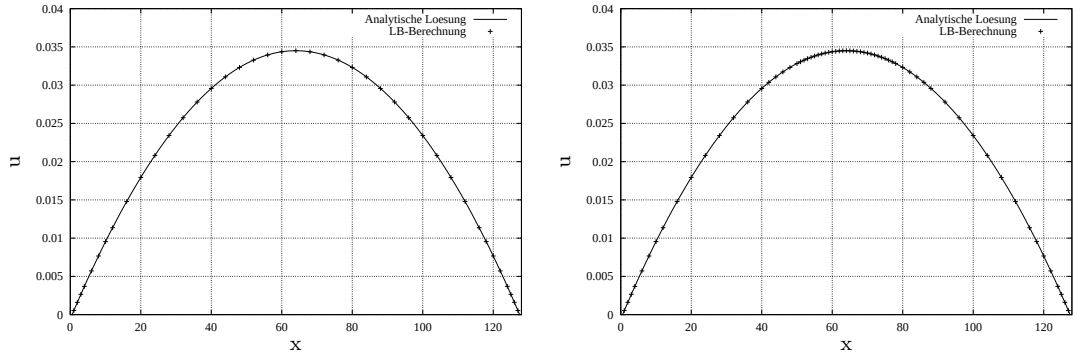

Abbildung 7.25: Poiseuille Strömung: Auswertung für Simulation 2 (links) und 3(rechts)

Die relativen Fehler aller drei Simulationen sind verschwindend gering und von der gleichen Größenordnung. Die Hauptfehlerquelle liegt in der Differenz des angenommenen Wandabstands, der mit $q = 0.5$ festgelegt wurde und des aus der Simulation resultierenden Wandabstands, der nur dann exakt ist, wenn die Viskosität gegen den Wert $\nu = 0\frac{\Delta x^2}{\Delta t}$ geht.

Damit ist der Prototyp zumindest für Strömungen, deren Profile parabolischer Natur sind und somit durch die Skalierungs- und Interpolationsschemata abgebildet werden können, verifiziert.

7.6.2 Taylor-Couette Strömung

In der zweiten Untersuchung wird der *abklingende Taylor-Wirbel* betrachtet, um zu prüfen, inwieweit die Konvergenzrate bezüglich des Diskretisierungs- und des Kompressibiltätsfehlers von den Interpolationsschemata am Gitterinterface abhängt. Eine genaue Versuchsbeschreibung des realen Experiments kann in [45, 76] eingesehen werden.

Der Strömungsverlauf und auch das Abklingverhalten des Taylor-Wirbels in einem 2π-periodischen System wird mit den folgenden Gleichungen exakt beschrieben:

$$u_x(x, y, t) = u_0[-e^{-2\nu t} \cdot cos(x) \cdot sin(y)] \tag{7.26}$$

$$u_y(x, y, t) = u_0[e^{-2\nu t} \cdot sin(x) \cdot cos(y)] \tag{7.27}$$

$$p(x, y, t) = u_0^2[-\frac{1}{4}e^{-4\nu t} \cdot cos(2x) \cdot cos(2y)] \tag{7.28}$$

u_0 stellt eine initiale Geschwindigkeit und t den zeitliche Fortschritt des transienten Strömungsverlaufs dar. Die Reynoldszahl wird mit

$$Re = \frac{n \cdot \Delta x_g \cdot u_0}{\nu} \tag{7.29}$$

berechnet, wobei n die Anzahl der Gitterabstände Δx_g in einer Raumdimension ist.

7.6.2.1 Simulations-Setup

Für die folgende Konvergenzstudie wird eine Reynoldzahl von 50 gewählt. In x- und in y-Richtung werden periodische Randbedingungen gesetzt. Damit ist das System vollständig ohne Randbedingungen modelliert und deren Einflüsse ausgeschlossen. In der Abbildung ist das grundlegende Rechengitter, überlagert mit Stromlinien des Wirbels, skizziert.

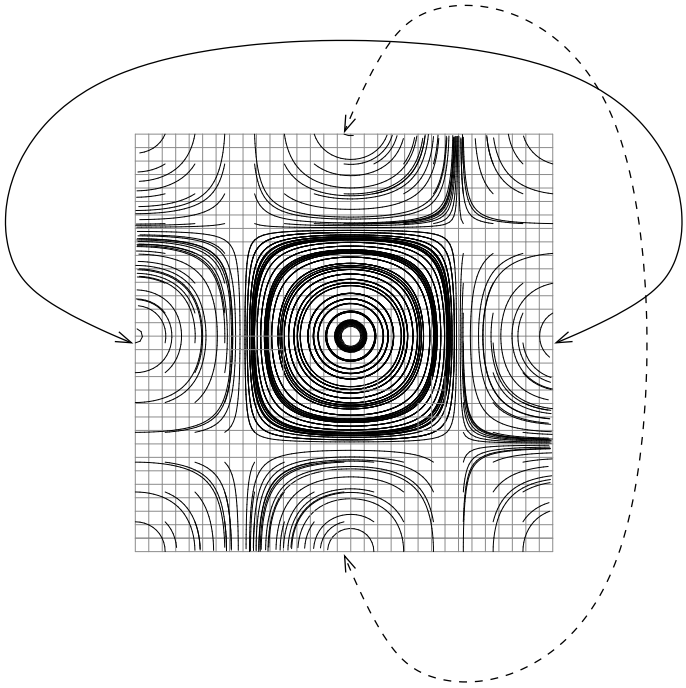

Abbildung 7.26: Taylor-Couette Strömung: Gitter-Setup (mit Stromlinien)

Das numerische Experiment besteht aus drei Simulationsläufen mit unterschiedlich vielen Gitterebenen[9]:

Gitter - $\Delta TD = 0$: Zur Verifikation der Konvergenzrate des grundlegenden Lattice-Boltzmann Algorithmus des Prototypen werden zunächst uniforme Gitter mit verschiedenen Gitterebenen ($l_0 = 4..9$) verwendet.

Gitter - $\Delta TD = 1$: In der nächsten Versuchsreihe werden, wie in Abbildung 7.27 dargestellt, vier Patches[10] höherer Auflösung ($l_0 + 1$) symmetrisch im

[9]Beispiel: In einem Gitter mit zwei Gitterebenen liegen Knoten von genau zwei Baumtiefen TD (Tree-Depth) vor $\rightarrow \Delta TD = 1$.

[10]hier: rechteckige Zonen, in denen die Auflösung erhöht wird.

Rechengebiet plaziert.

Gitter - $\Delta TD = 2$: Die dritte Versuchsreihe enthält Gitterkonfigurationen, deren verfeinerte Regionen um zwei Gitterebenen höher aufgelöst wird, als die Grundebene.

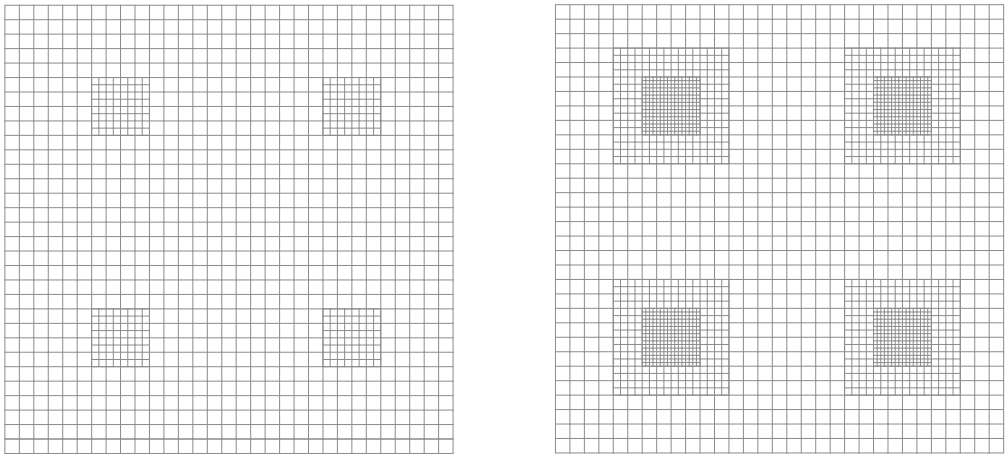

Abbildung 7.27: Taylor-Couette Strömung: Gitterkonfiguration 2 (links) und 3 (rechts)

Berechnungsablauf: Das Lattice-Boltzmann Verfahren konvergiert nur dann mit zweiter Ordnung, wenn sowohl der räumliche Diskretisierungsfehler durch eine höhere Gitterauflösung *und gleichzeitig* der Kompressibilitätsfehler durch entsprechende Abminderung der makroskopischen Geschwindigkeit reduziert werden. Das bedeutet, dass bezogen auf ein Gitter der Auflösung l_0 und einer Geschwindigkeit u_0 der Fehler um den Faktor *vier* reduziert wird, wenn die Auflösung verdoppelt wird und die Geschwindigkeit u_0 halbiert wird. Für die Auswertung von transienten Vorgängen zu einem bestimmten Zeitpunkt, wie es bei der Taylor-Couette Strömung der Fall ist, muss die Simulationsdauer entsprechend berechnet werden. Die *Halbierung* der Geschwindigkeit und die *Verdoppelung* der Gitterpunkte entlang einer Raumachse erfordert demnach eine *Vervierfachung* der Simulationszeit. Die Eingangsparameter für die drei Versuchsreihen werden in der nachfolgenden Tabelle für die Gitterebenen TD (Tree-Depth) angegeben:

TD	u_0	$m \cdot \Delta t_f$	$L = 2^{TD}$
4	0.15625000	100	16
5	0.07812500	400	32
6	0.03906250	1600	64
7	0.01953125	6400	128
8	0.00976563	25600	256
9	0.00488281	102400	512

Tabelle 7.2: Taylor-Couette Strömung: Eingangsparameter

Die Viskosität für die Versuchsreihen $\Delta TD = 0$ und $\Delta TD = 1$ wurde mit $\nu = 0.05$ angenommen. Die Versuchsreihe $\Delta TD = 2$ wurde mit einer kinematischen Viskosität von $\nu = 0.02$ durchgeführt, um gültige Relaxationszahlen zu erhalten.

Umlaufzeit: Als Maß für eine sinnvolle Laufzeit für die Simulation wird die Dauer einer vollständigen Durchwanderung des Rechengebietes angenommen. Die Gebietsgröße ist mit 2π gesetzt. Die reale Simulationsdauer ergibt sich somit mit:

$$T_{real} = \frac{2\pi}{u_0} \tag{7.30}$$

Die entsprechende Berechnungsdauer der Lattice-Boltzmann Simulation ist:

$$T_{LB} = \frac{n \cdot \Delta x}{u_0} \tag{7.31}$$

Diese Bezugszeiten dienen nun der Umrechnung von realer Laufzeit t_{real} auf die Simulationsdauer t_{LB}:

$$\frac{t_{LB}}{T_{LB}} = \frac{t_{real}}{T_{real}} \Longleftrightarrow t_{LB} = \frac{T_{LB}}{T_{real}} \cdot t_{real} = \frac{n \cdot \Delta x}{u_0} \cdot \frac{u_0}{2\pi} \cdot t_{real} = \frac{n \cdot \Delta x}{2\pi} \cdot t_{real} \tag{7.32}$$

Für den ersten Eintrag ($TD = 4$) in obiger Tabelle ergibt sich die Simulationsdauer für eine Gebietsdurchwanderung folgendermaßen:

$$t_{LB} = \frac{n \cdot \Delta x}{2\pi} \cdot t_{real} = \frac{n \cdot \Delta x}{2\pi} \cdot \frac{2\pi}{u_0} = \frac{16 \cdot \Delta x}{0.15625 \frac{\Delta x}{dt}} = 101,85\dot{\Delta}t \approx 100 \cdot \Delta t \tag{7.33}$$

7.6.2.2 Auswertung

Quantitative Bewertung: Als geeignetes Maß für die Beurteilung der Fehler der Lattice-Boltzmann Simulation wird die diskrete $L2$-Norm (euklidische Norm) gewählt:

$$E2(t) = \frac{\sqrt{\sum_i (u_{LB}(\vec{x}_i, t) - (u_{real}(\vec{x}_i, t))^2}}{\sqrt{\sum_i u_{real}(\vec{x}_i, t)^2}} \tag{7.34}$$

Mit dem Fehler zweier Gitter kann dann die Konvergenzrate ρ berechnet werden:

$$\rho = \log_{\frac{\Delta x_g}{\Delta x_f}} \frac{E_g}{E_f} \tag{7.35}$$

In [83] wird die Konvergenzrate von $\rho = 2$ für uniforme Verfeinerungen bestätigt. Die Auswertung der drei durchgeführten Simulationsläufe ist in der folgenden Tabelle angegeben:

TD	$\rho(\Delta TD = 0)$	$\rho(\Delta TD = 1)$	$\rho(\Delta TD = 2)$
5	2.054	1.831	2.185
6	2.047	1.906	2.758
7	1.979	2.068	2.209
8	2.024	2.039	1.799
9	2.045	2.017	2.663

Tabelle 7.3: Taylor-Couette-Strömung: Auswertung

In der ersten und der zweiten Simulationsreihe konnte die Konvergenzrate von 2 verifiziert werden. Der dritte Simulationslauf ist starken Schwankungen unterworfen.

Qualitative Bewertung: Für die qualitative Bewertung werden Isolinien der x-Komponente des Geschwindigkeitsvektors sowie Isolinien des Druck-Verlaufs herangezogen. Da sich das Interesse vor allem auf den Bereich des Gitterübergangs beschränkt, wird ein Patch (links unten) der Gitterkonfiguration 3 (vgl. Abb. 7.27) genauer betrachtet. Die Ergebnisse der Versuchsreihen 1 und 2 stellen weniger Ansprüche an die Gitterübergänge und werden daher nicht dargestellt. Wie in den Abbildungen 7.28 und 7.29 zu sehen ist, sind keine Unstetigkeitsstellen in den Verläufen zu beobachten.

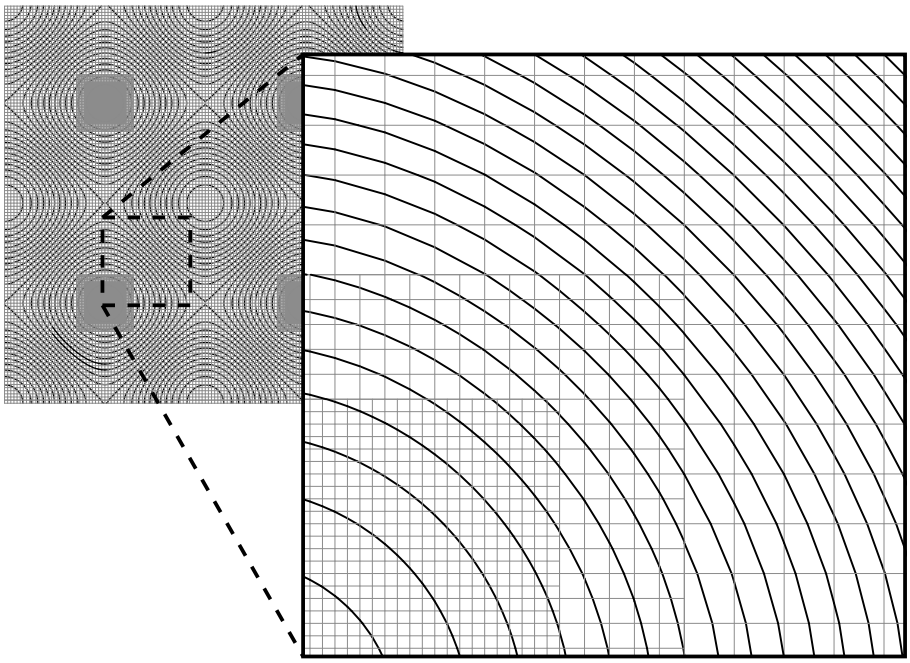

Abbildung 7.28: Taylor-Couette Strömung: Isolinien des Drucks

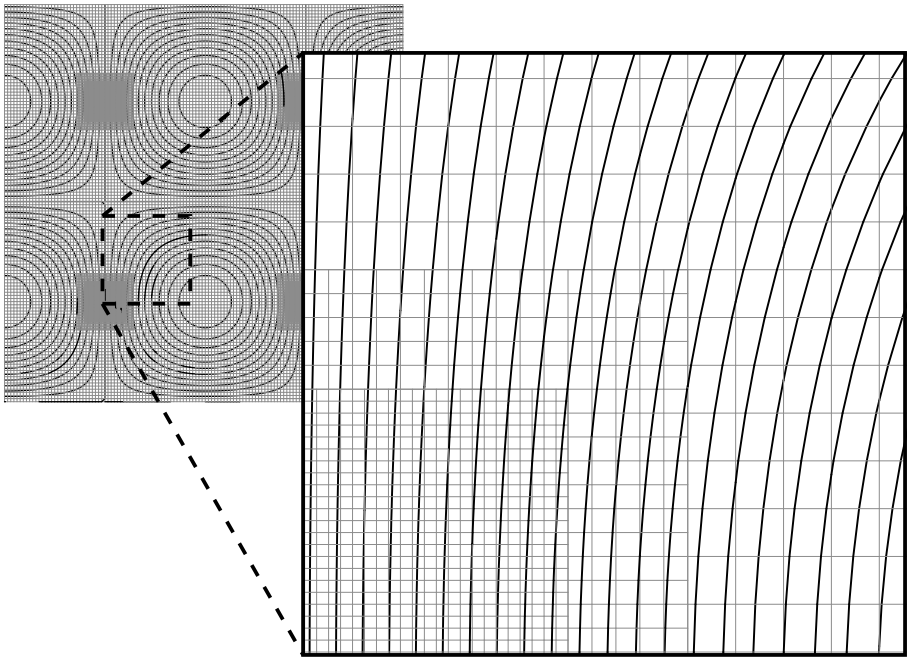

Abbildung 7.29: Taylor-Couette Strömung: Isolinien der x-Komponente des Geschwindigkeitsvektors

7.6.3 Zylinderumströmung: Re=20 und Re=100

Die zweidimensionale Zylinderumströmung im freien Strahl (keine Kanalwände) lässt sich im Wesentlichen in drei charakteristische Strömungsstrukturen einteilen: stationäre Strömung ohne Ablösung ($Re \leq 5$), stationäre Strömung mit zwei symmetrischen Wirbeln hinter dem Zylinder ($5 \leq Re \leq 46$) und instationäre Strömungen mit periodischer Wirbelablösung ($Re \geq 46$) [65]. Die Zylinderumströmung für sehr kleine Reynoldszahlen wird in Kapitel 8.7.1 behandelt, während die beiden Strömungen im Bereich der moderaten Reynoldszahlen mit einem modifizierten Simulationssetup im Folgenden erläutert wird.

Für die in [89] veröffentlichte und in Abbildung 7.30 dargestellte Zylinderumströmung gibt es keine analytische Lösung. Da es sich um eine Strömung handelt, die komplexe Strömungsstrukturen ausweist, eignet sich dieses Beispiel um die Tauglichkeit des Multiskalen Lattice-Boltzmann Ansatzes zu verfizieren. Als Ein-

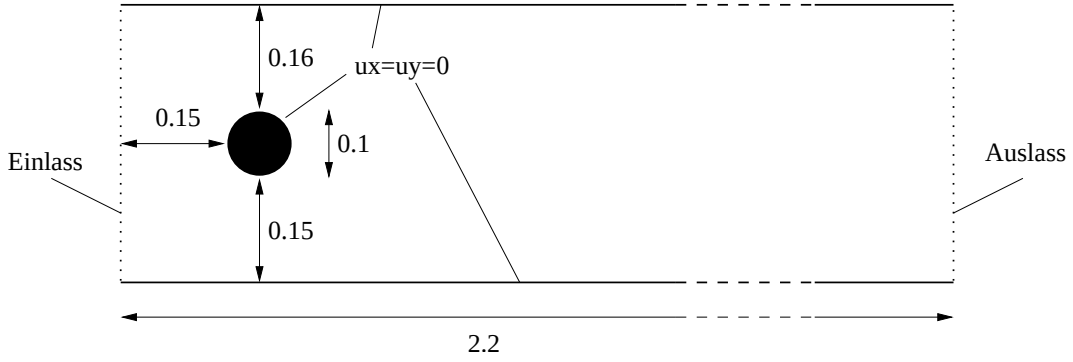

Abbildung 7.30: Zylinderumströmung: Geometrie und Randbedingungen aus [89]

flussrandbedingung ist ein Parabelprofil mit der Funktion

$$u_x(0, y) = 4 \cdot u_{max} \cdot y \cdot (H - y)/H^2, \quad u_y(0, y) = 0 \tag{7.36}$$

gewählt. Die Reynoldszahl ist durch

$$Re = \frac{\overline{u} \cdot D}{\nu} \quad \text{mit} \quad \overline{u} = \frac{2}{3} \cdot u_{max} \tag{7.37}$$

definiert. Der Zylinder ist asymmetrisch im Kanal positioniert. Gegenstand der Untersuchung ist zum einen eine stationäre Strömung bei $Re = 20$ und zum anderen eine transiente Umströmung bei $Re = 100$. Die Validierung mit der Referenzlösung aus [89] erfolgt an den in Kapitel 7.5.1 vorgestellten Größen Rezirkulationszone, Kraftbeiwerte, Druckdifferenz und Strouhalzahl.

7.6.3.1 Simulations-Setup

Für die Zylinderumströmung bei $Re = 20$ werden vier Simulationsreihen durchgeführt. In den ersten beiden Durchläufen wird der Kollisionsoperator im Ge-

schwindigkeitsraum (LBGK) mit dem Kollisionsoperator im Momentenraum (GLBE) verglichen. Hierfür wird das Gitter sukzessive uniform verfeinert. Im zweiten Durchlauf wird der Einlassbereich und der Zylinder sukzessive höher aufgelöst, während die Grundauflösung konstant gehalten wird. Die Haftrandbedingungen sind mit der Boundary-Fitting Methode umgesetzt und sollen für eine erhöhte Genauigkeit sorgen. Der Versuchsablauf kann den Ergebnistabellen 7.4 bis 7.7 entnommen werden.

Für die Zylinderumströmung bei $Re = 100$ werden vier Simulationen durchgeführt. Als Vergleichslösung wird auf einem uniformen Gitter mit der Baumtiefe 7 sowohl der BGK- als auch der GLBE-Kollisionsoperator angewandt. Demgegenüber wird mit dem Gitter (vgl. Abb. 7.31), bestehend aus den Baumtiefen 6 bis 9, die Simulation wieder mit beiden Kollisionsoperatoren berechnet.

7.6.3.2 Auswertung

Re=20: Ein Vergleich der Tabellen 7.4 und 7.5 zeigt, dass die Simulationen mit uniformer Verfeinerung unter Verwendung von beiden Kollisionsoperatoren äquivalente Ergebnisse aufweisen[11]:

TD	c_D	c_L	L_a	ΔP	$Knoten$
5	6.405	0.0108	7.000	1.5775	2437
6	5.631	0.0116	8.415	1.4183	8771
7	5.589	0.0115	8.361	1.3017	35416
8	5.575	0.0107	8.430	1.2968	142358
$Referenz-$	5.57	0.0104	8.42	1.3022	
$spektrum$	5.59	0.0110	8.52	1.3066	

Tabelle 7.4: Zylinderumströmung (LBGK): $Re = 20$ - Uniforme Gitter

TD	c_D	c_L	L_a	ΔP	$Knoten$
5	6.354	0.0185	7.000	1.5753	2437
6	5.654	0.0116	8.444	1.3667	8771
7	5.588	0.0115	8.365	1.2951	35416
8	5.579	0.0107	8.434	1.2929	142358
$Referenz-$	5.57	0.0104	8.42	1.3022	
$spektrum$	5.59	0.0110	8.52	1.3066	

Tabelle 7.5: Zylinderumströmung (GLBE): $Re = 20$ - Uniforme Gitter

[11] Die in den Tabellen abgedruckten Referenzspektren wurden aus [89] entnommen. Da es für diese Problemklasse keine analytischen Lösungen gibt, wurden ausgehend von den Ergebnissen von 15 Gruppen statistische Methoden verwendet, um diese Spektren abzuschätzen. Es ist daher nicht zwingend zu erwarten, dass alle Lösungswerte innerhalb der angegebenen Spektren liegen.

Die Simulationsdurchläufe 3 und 4 (Tabellen 7.6 und 7.7), basierend auf nicht-uniformen Gittern, zeigen quantitativ ähnlich gute Ergebnisse, teilweise sogar besser als die Ergebnisse der uniformen Verfeinerung. Hervorzuheben ist allerdings, dass die Multiskalen Simulationen deutlich weniger Gitterknoten benötigten, als die uniformen Simulationen. Das Multiskalen-Gitter ($TD = 6..9$) bestand aus 18316 und das uniforme Vergleichsgitter ($TD = 8$) aus 142358 Fluidknoten. Dies entspricht einer Reduktion des Knotenanzahl um einen Faktor von 7.77. Es wird deutlich, dass ein feines Gitter im Interessenbereich ausreicht, um gute Ergebnisse zu erzielen.

TD	c_D	c_L	L_a	ΔP	$Knoten$
$6-7$	5.627	0.01181	8.504	1.3369	9386
$6-8$	5.591	0.0110	8.405	1.3082	11657
$6-9$	5.585	0.0107	8.383	1.3068	18276
$Referenz-$	5.57	0.0104	8.42	1.3022	
$spektrum$	5.59	0.0110	8.52	1.3066	

Tabelle 7.6: Zylinderumströmung (LBGK): $Re = 20$ - Nicht-uniforme Gitter

TD	c_D	c_L	L_a	ΔP	$Knoten$
$6-7$	5.609	0.0119	8.497	1.3413	9386
$6-8$	5.588	0.0110	8.394	1.3076	11657
$6-9$	5.584	0.0108	8.382	1.3068	18276
$Referenz-$	5.57	0.0104	8.42	1.3022	
$spektrum$	5.59	0.0110	8.52	1.3066	

Tabelle 7.7: Zylinderumströmung (GLBE): $Re = 20$ - Nicht-uniforme Gitter

Re=100: Die Ergebnisse der vier Simulationen in Tabelle 7.8 sind vergleichbar bezüglich Kollisionsoperatoren und Gitterkonfiguration. Die Strouhalzahl stimmt mit den zu erwartenden Wert gut überein.

	$c_{D,max}$	$c_{L,max}$	St	ΔP	$Knoten$
$LBGK_{uniform} : Td = 7$	3.2650	0.9492	0.3076	1.1674	35416
$GLBE_{uniform} : Td = 7$	3.2754	0.9549	0.3076	1.1468	35416
$LBGK : Td = 6..9$	3.2645	1.0709	0.3050	1.1174	18316
$GLBE : Td = 6..9$	3.2566	1.0642	0.3050	1.1164	18316
$Referenz-$	3.2200	0.9900	0.2950	1.0933	
$spektrum$	3.2400	1.0100	0.3050	1.1111	

Tabelle 7.8: Zylinderumströmung: $Re = 100$

Qualitative Bewertung: Die qualitative Bewertung erfolgt wieder anhand der Isolinien-Darstellung des Drucks, einer Geschwindigkeitskomponente und einer Spannungskomponente. Die Beobachtungen beschränken sich auf einen Bereich in der Nähe des Zylinders, in der auch die Gitterübergänge vorhanden sind. Beispielhaft wird nur ein Datensatz der Simulation $TD = 6..9, GLBE, Re = 20$ herangezogen.

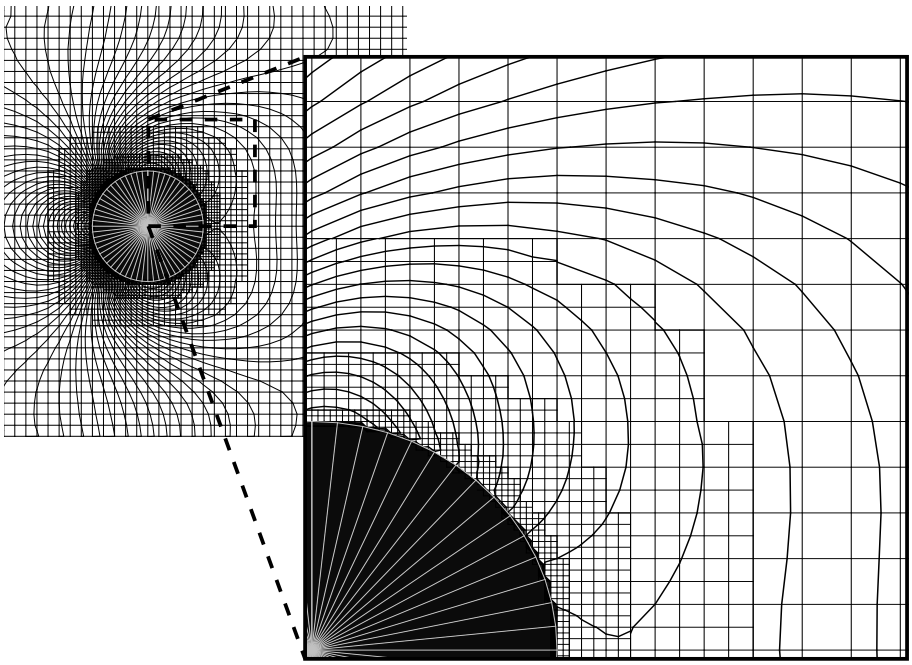

Abbildung 7.31: Zylinderumströmung: Isolinien des Drucks

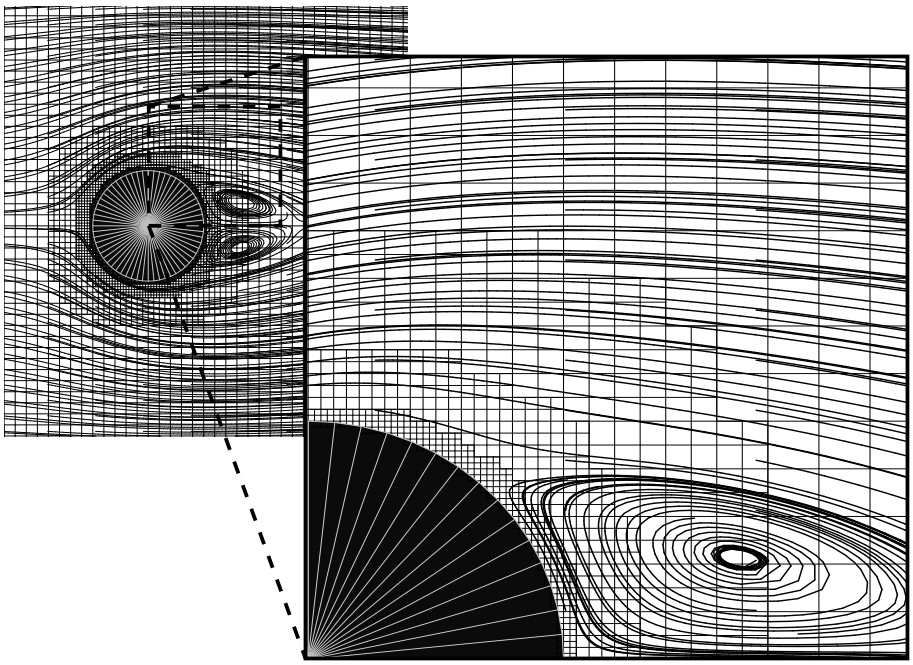

Abbildung 7.32: Zylinderumströmung: Stromlinien

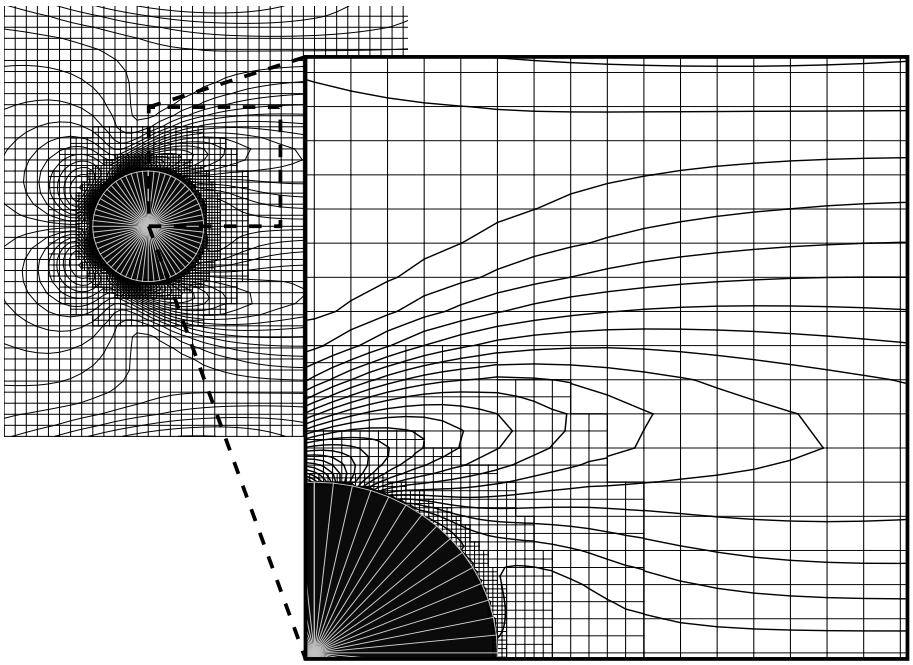

Abbildung 7.33: Zylinderumströmung: Isolinien der xy-Komponente des viskosen Spannungstensors

Alle drei Darstellungen zeigen keine Unstetigkeiten im Bereich des Gitterüberganges. Eine Kontinuität von Druck, Impuls und Spannungen ist somit zumindest grafisch nachgewiesen.

7.7 Zusammenfassung

Der größte Aufwand bei der Implementierung des Prototypen liegt in der abschliessenden Aufbereitung des Gitters, in der die Datenstruktur in eine Form gebracht und erweitert wird, um den Multiskalen LB-Algorithmus durchführen zu können. Als wichtigste Punkte sind die *modifizierte Glättung* sowie die Initialisierung der zeitabhängigen Simulationsgrößen zu nennen. Sowohl die Implementierung der Randbedingungen als auch die Implementierung der evaluierenden Funktionalität hängt in starkem Maße von der Datenstruktur ab.

Zum Abschluss diesen Kapitels wurde der Prototyp einer ausführlichen Verifizierung unterzogen. Mithilfe der Poiseuille-Strömung und der Taylor-Couette-Strömung konnte die Richtigkeit des Skalierungsansatzes und die Beibehaltung der Konvergenzrate bestätigt werden. Der kontinuierliche Verlauf der Masse, des Impulses und der Spannungen wurde im letzten Beispiel, der Zylinderumströmung, mit entsprechenden Isolinien-Plots nachgewiesen. Hierbei konnte die Tauglichkeit des Multiskalen Lattice-Boltzmann Algorithmus basierend auf dem LBGK-Ansatz und dem GLBE-Ansatz für stationäre Strömungen ($Re = 20$) und für instationäre Strömungen ($Re = 100$) verifiziert werden.

Kapitel 8

Adaptive LB-Simulationen für stationäre Strömungen

Im Rahmen dieser Arbeit wurde erstmals ein adaptiver Lattice-Boltzmann Strömungslöser für stationäre Strömungsphänomene basierend auf Baumdatenstrukturen entwickelt und implementiert [20]. Die Grundlagen bezüglich der adaptiven Strömungssimulation sind Gegenstand diesen Kapitels. Neben der Simulationsstrategie werden Sensoren und Kriterien zur Gitterverfeinerung sowie die notwendigen Mechanismen der Gitteradaption ausführlich beschrieben. Die Besonderheiten, die sich durch das Lattice-Boltzmann Verfahren ergeben, werden ebenfalls herausgestellt.

8.1 Einführung und Motivation

Adaptive Simulationen haben das Ziel, die Qualität einer numerischen Lösung eines mathematischen Modells zu verbessern. Während in den Finite-Elemente Methoden für die Adaption des Lösungsvorgangs die Anpassung der Netzdichte (h-Adaptivität) und des Polynomgrads der Ansatzfunktionen (p-Adaptivität) [25] denkbar ist, muss man sich bei den Finite-Differenzen Methoden, zu denen auch die Lattice-Boltzmann Verfahren zählen, auf die h-Adaptivität beschränken. Es wird immer angestrebt die Dichte der Freiheitsgrade problemgerecht möglichst optimal auf dem Berechnungsnetz zu verteilen und somit die Netzabhängigkeit der Lösung zu minimieren. Durch die Ordnung des Verfahrens von zwei bezüglich der Raum- und der Zeitdiskretisierung im Lattice-Boltzmann Verfahren, bewirkt eine Verfeinerung der Gitterdichte um den Faktor Zwei eine Reduktion des Fehlers um den Faktor Vier.

In allen h-adaptiven Verfahren besteht die Notwendigkeit, das Rechengitter zur Laufzeit entsprechend der gewählten Verfeinerungsstrategie anzupassen. Wegen der kartesischen Gitterstruktur des in dieser Arbeit vorgestellten Multiskalen Lattice-Boltzmann Verfahrens zeigt sich der Einsatz von Baumdatenstrukturen

als Berechnungsgrundlage als vorteilhaft, weil somit eine dynamische Erweiterung des Gitters jederzeit möglich ist. Bereits in den Arbeiten von Wang et. al. und De Zeeuw [103, 104, 111] wurden Baumdatenstrukturen in adaptiven Strömungssimulationen basierend auf der Diskretisierung der Navier-Stokes Gleichungen eingesetzt. Deister stellte in seinen Arbeiten [22, 23] ein Applikationsmodell vor, in dem ein vorhandener Strömungslöser mit einem selbstorganisierenden Gitter vereint wird. Alle Arbeiten versuchen durch Verwendung geeigneter Verfeinerungskriterien die Benutzerinteraktion zur Verteilung der Freiheitsgrade während der Simulation auf ein Minimum zu reduzieren. Der Grund für die Bestrebungen liegt in dem Verlangen der Industrie und der Ingenieure, die Dauer der Simulationszyklen im Rahmen einer nahezu vollkommen automatisierten Arbeitsweise zu verkürzen und die Anwendung von Simulationswerkzeugen zu ermöglichen, ohne sich vorher verfahrensspezifisches Expertenwissen aneignen zu müssen.

8.2 Simulationsstrategie: Der adaptive Zyklus

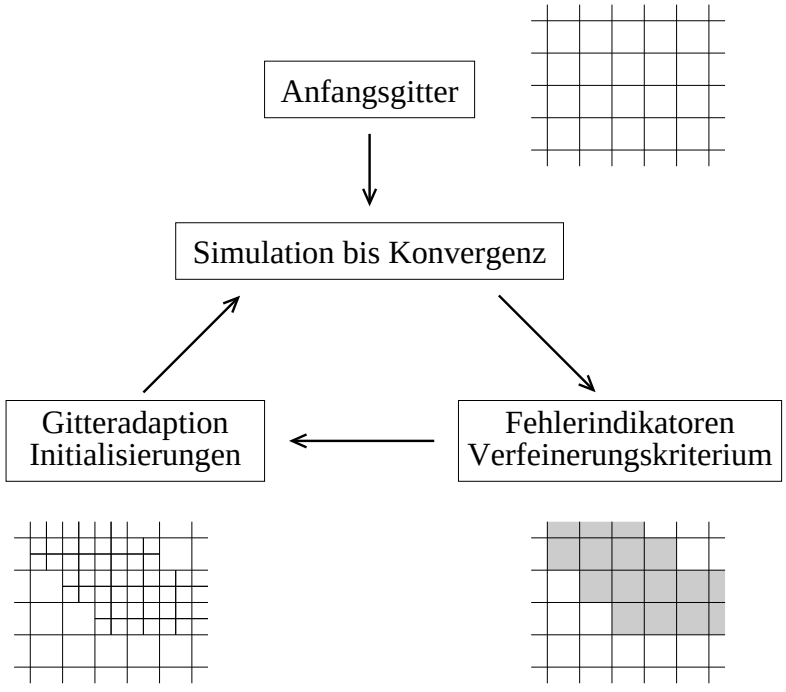

Abbildung 8.1: Der adaptive Zyklus

Der adaptive Simulationszyklus ist in Abbildung 8.1 dargestellt. Typischerweise beginnt die Simulation mit der Generierung eines kartesischen Anfangsgitters, dessen lokale Auflösung nur von der geforderten anfänglichen Oberflächendiskretisierung abhängt. Diese wird unter anderem durch Stabilitätsüberlegungen, wie

sie in Kapitel 6.6.2 diskutiert wurden, eingeschränkt. Eine Anfangslösung wird berechnet und mithilfe von Fehlerindikatoren werden jene Gitterzellen markiert, welche für eine Verfeinerung vorgesehen sind. Als nächstes muss das Gitter entsprechend erweitert werden und die neu hinzugekommenen Gitterknoten initialisiert werden. Die Simulation wird fortgeführt, bis erneut ein Konvergenzkriterium erfüllt ist. Dieser adaptive Zyklus wird solange fortgeführt, bis die Qualität der Lösung zufriedenstellend ist oder vom Verfeinerungsmechanismus keine Zellen mehr markiert werden.

Die gesamte Simulationsdauer kann reduziert werden, indem das Konvergenzkriterium selbst mit zunehmender Iterationsanzahl verfeinert wird. Zunächst wird ein relativ großer Schwellenwert für die Unterbrechung der Simulation gewählt. Bei jedem adaptiven Iterationsschritt wird dieser sukzessive verkleinert. Hierdurch kann es zwar vorkommen, dass durch zu grobe Konvergenzkriterien mehr Gitterzellen als notwendig markiert werden, aber dadurch, dass diese Bereiche früher verfeinert werden, wird auch früher eine bessere Lösung erzielt.

Im Gegensatz zu anderen numerischen Diskretisierungsmethoden (z. B. FEM, FVM) ist man bei den Lattice-Boltzmann Verfahren bezüglich der möglichen Auflösungsunterschiede eingeschränkt (vgl. Kapitel 6.6.2). Noch bevor eine adaptive Simulation gestartet wird, muss sich der Benutzer bewusst sein, welche maximale Gitterebene angestrebt wird, damit die Relaxationzahlen im numerisch stabilen und hydrodynamisch validen Bereich bleiben und entsprechend initialisiert werden können. Alternativ kann dies auch durch einen Algorithmus im Quellcode abgefangen werden.

8.3 Sensor-basierte Fehlerindikatoren

Die Wahl der verwendeten Fehlerschätzer oder Fehlerindikatoren ist von entscheidender Bedeutung für den Erfolg der adaptiven Strategie. In den Finite-Elemente Verfahren liegt der Ansatz in der Minimierung der *potentiellen Energie*, die mit der Dehnungsenergie korrespondiert. In [25] werden erweiterte adaptive FE-Verfahren vorgestellt, in denen Fehlerschätzer verwendet werden, die näherungsweise den Fehler in der Energienorm berechnen können. Hierbei kann eine Verteilung der Fehler im Berechnungsgebiet angegeben werden. Für Strömungssimulationen werden häufig Finite-Volumen, Finite-Differenzen und zunehmend Lattice-Boltzmann Verfahren eingesetzt. Fehlerschätzer, wie sie für die Finite-Elemente Methode entwickelt wurden, existieren hier nicht [32]. Die Berechnung eines Residuen-Fehlers, welcher bei Verfahren höherer Ordnung, per Definition aus Ableitungen hoher Ordnung bestehen muss, führte bisher zu keinen numerisch oder physikalisch sinnvollen Erkenntnissen [32]. Daher müssen andere Möglichkeiten gefunden werden, um Diskretisierungsfehler zu quantifizieren oder zumindest zu indizieren.

8.3.1 Fehlerindikations-Strategien

Prinzipiell gibt es drei Ansätze zur Formulierung von Fehlerindikatoren:

1. Residuen-Berechnung:
   Wie schon erwähnt, bedingt die Fehlerbestimmung basierend auf den Residuen oft die Berechnung höherer Ableitungen und stellt damit einen beachtlichen Rechenaufwand dar. Der aufrauhende Effekt der Differentiation verhindert meist die Erzeugung von physikalisch interpretierbaren Ergebnissen [32]. Zudem kann es selbst bei stationären Strömungsphänomenen zu lokalen instationären Effekten kommen und somit eine Schranke des Residuums nicht unterschritten werden, obwohl die Lösung bereits physikalisch akzeptabel wäre.

2. Gitterkonvergenz:
   Das Indizieren von Fehlern mithilfe der Gitterkonvergenz basiert auf dem Vorgehen, im ersten adaptiven Schritt das gesamte Berechnungsgebiet zu verfeinern und die beiden Lösungen zu vergleichen. Man erhält eine mögliche Verteilung des Fehlers. Dies führt allerdings a priori zu einem stark erhöhten Rechenaufwand, da zuerst verfeinert und dann über die weitere Verteilung der Freiheitsgrade entschieden wird.

3. Sensoren:
   Zu einer weit verbreiteten Methode hat sich die Anwendung von sogenannten *phänomenologischen Sensoren* entwickelt. Es handelt sich hierbei um eine empirische Vorgehensweise, in der physikalisch interpretierbare Werte im Rahmen eines Postprocessing-Schrittes bewertet werden. In den Arbeiten von Hentschel [44] und Deister [24] werden einige Sensoren detailiert beschrieben und in deren Applikationen eingesetzt.

Da der in dieser Arbeit entwickelte Prototyp ebenfalls Sensor-basierte Fehlerindikatoren verwendet, werden die Grundlagen diesbezüglich im Folgenden vorgestellt.

8.3.2 Definition der phänomenologischen Sensoren

Die Aufgabe eines Sensors ist das Aufspüren bestimmter strömungsmechanischer Phänomene. Er wird als gut beurteilt, wenn er einen Bereich, der für den Benutzer von Interesse ist, erfassen und eine Präzisierung der geometrischen Ausdehnung des strömungsmechanischen Phänomens bewirken kann. Bezugnehmend auf die Bestimmung der Rezirkulationszone einer Zylinderumströmung (vgl. Kapitel 7.5.1.3) könnte ein Sensor Bereiche anzeigen, in denen negative Geschwindigkeitskomponenten in x-Richtung auftreten:

$$\Phi_R = \begin{cases} 0 & \text{falls } u_x > 0, \\ 1 & \text{sonst} \end{cases} \tag{8.1}$$

Typischerweise sind Sensoren Funktionen von physikalischen Strömungsgrößen, wie Geschwindigkeit und Druck oder Gradienten derselben.

Sensoren sollten folgende Anforderungen erfüllen:

- Einzelne Sensoren sollten immer nur einzelne Phänomene erfassen, um gezielte Untersuchungen durchführen zu können. Für eine umfassende Untersuchung können natürlich mehrere Sensorenuntersuchungen überlagert werden.
- Die Berechnung der Sensorfunktion sollte numerisch effizient umsetzbar sein, um den Mehraufwand durch den adaptiven Zyklus gering zu halten.
- Für die Erfassung von allgemeinen Strömungsphänomenen, unabhängig von den einzelnen Koordinatenrichtungen, soll der Sensor isotropes Verhalten aufweisen.
- Die Empfindlichkeit des Sensors soll von der lokalen Gitterdichte abhängen. Damit soll erreicht werden, dass Strömungsphänomene in sehr grob aufgelösten Teilgebieten nicht übersehen werden und gleichzeitig fein aufgelöste Gebiete nicht unnötig weiter verfeinert werden.
- Die Sensorfunktion soll in *glatten Gebieten* unempfindlich und in Bereichen mit hohen Lösungsgradienten der Strömungsvariablen sehr empfindlich reagieren. Als Lösungsgradienten können u.a. Gradienten des Geschwindigkeitsfeld (z. B. Vortizität) oder höhere Ableitungen der Primärvariablen (z. B. Gradienten der Spannungskomponenten) verwendet werden.

8.3.3 Formulierung der Sensoren

In dieser Arbeit wurden nur Phänomen-basierte Sensoren implementiert. Die Sensorfunktion Φ, basierend auf dem physikalischen Phänomen P und der lokalen Gitterweite Δx, ist wie folgt definiert:

$$\Phi = (|P| \cdot \Delta x^{\alpha})^{\beta} \tag{8.2}$$

Der Exponent α steuert die Empfindlichkeit des Sensors bezüglich der Gitterweite und wird meist mit $\frac{3}{2}$ angenommen. Für eine Verstärkung der Gradienten der Sensorfunktion dient der Exponent β, wobei der resultierende Effekt nur schwach ausgeprägt ist und β meist zu Eins gesetzt wird.

Mehrere Sensoren wurden eingesetzt:

Divergenz: Der Divergenz-Sensor überprüft die Kontinuitätsgleichung für die Masse und wird mit

$$\Phi_{div} = \left| \frac{\partial u_i}{\partial x_i} \right| \cdot \Delta x^{\frac{3}{2}} \tag{8.3}$$

definiert. Es handelt sich hierbei tatsächlich um einen Residuen-Sensor, dessen Funktion idealerweise an jedem Ort im Rechengebiet verschwinden sollte. Residuen bezüglich der Massenerhaltung treten vor allem an scharfen Festkörperkanten auf und in jenen Bereichen, in denen die Massenerhaltung per Definition nicht zwingend eingehalten wird, wie zum Beispiel bei den Gitterinterfaces in Multiskalen-Simulationen. Das Lattice-Boltzmann Verfahren erweist sich hier als besonders vorteilhaft, weil die Divergenz proportional zur Spur des Spannungstensors ist, welcher völlig lokal berechnet wird, und daher keine räumlichen Differenzen gebildet werden müssen.

Rotation: Zur Lokalisierung von Scherschichten in Bereichen von Wänden oder Wirbelzonen dient der Rotations-Sensor.

$$\Phi_{rot} = \left| \epsilon_{ijk} \frac{\partial u_k}{\partial x_j} \right| \cdot \Delta x^{\frac{3}{2}} \quad \text{mit} \quad \epsilon_{ijk} = \begin{cases} 1 & \text{i,j,k zyklisch} \\ -1 & \text{i,j,k antizyklisch} \\ 0 & \text{sonst} \end{cases} \tag{8.4}$$

Im Gegensatz zum Divergenz-Sensor müssen hier die Ableitungen mit Finiten-Differenzen berechnet werden, was vor allem bei nicht-uniformen Gittern einen enormen Implementierungs- und Rechenaufwand bedeutet.

Entropie-Gradient: Der Entropie-Gradient-Sensor ist prinzipiell als allgemeingültig anzusehen, da er Gebiete anspricht, die durch hohe Variation des Entropieniveaus gekennzeichnet sind [44]. Im Lattice-Boltzmann Kontext lässt sich die *relative Entropie S* nach Koelman völlig lokal mit

$$S = -\frac{k}{m} \sum_i f_i^{(0)} ln \frac{f_i^{(0)}}{w_i} \tag{8.5}$$

berechnen [106]. Der Faktor $\frac{k}{m}$ kann in isothermen Systemen für die Sensorfunktion zu Eins gesetzt werden. w_i sind die Wichtungsfaktoren (vgl. Kapitel 5.8.2). Die Berechnung der Sensorenwerte erfolgt mit:

$$\Phi_{entropy} = \left| \frac{\partial S}{\partial x_i} \right| \cdot \Delta x^{\frac{3}{2}} \tag{8.6}$$

Die Gradienten müssen wieder durch Finite-Differenzen bestimmt werden.

Absolute Geschwindigkeitsdifferenz: In diesem Sensor werden lediglich absolute Geschwindigkeitsdifferenzen innerhalb einer Gitterzelle berechnet:

$$\Phi_{velocity} = max(|\vec{u}|) - min(|\vec{u}|) \tag{8.7}$$

Dieser Sensor unterscheidet sich von den anderen bisher vorgestellten Sensoren in seiner Heuristik, die hier sicherlich am größten ist. Ziel wird hier sein, für den Geschwindigkeitsunterschied innerhalb einer Gitterzelle eine obere absolute Schranke zu definieren. In [111] wird hierfür 5% des maximalen im Gebiet befindlichen Geschwindigkeitsunterschieds gewählt.

8.4 Verfeinerungskriterium

Nachdem für alle Gitterpunkte die Berechnung der Sensorfunktion erfolgt ist, muss eine Entscheidung darüber getroffen werden, ob eine Gitterzelle verfeinert wird. Hierfür müssen zunächst Kriterien definiert werden, die sich in sensorbasierte, geometriebedingte und Zwangskriterien unterscheiden lassen.

Sensorbasiertes Kriterium: Typischerweise wird die Standardabweichung der Sensorenwerte mit

$$\sigma = \sqrt{\frac{\sum_{j=1}^{n} \Phi^2(j)}{n} - \overline{\Phi}} \approx \sqrt{\frac{\sum_{j=1}^{n} \Phi^2(j)}{n}} \tag{8.8}$$

berechnet [24, 96]. Der Erwartungswert $\overline{\Phi}$ wird vereinfachend mit Null angenommen. Das Adaptionskriterium τ ergibt sich durch Wichtung der Standardabweichung.

$$\tau = \lambda \cdot \sigma \quad \text{mit} \quad \lambda \approx 1 \tag{8.9}$$

Der Wertebereich von λ für einen Verfeinerungsschritt liegt erfahrungsgemäß im Intervall $[0.7, 1.3]$. Liegen keine Anhaltspunkte vor, ist die Wahl $\lambda = 1$ vorzunehmen. Eine Verfeinerung ist dann vorzunehmen, wenn

$$\Phi > \tau \tag{8.10}$$

gilt.

Geometriebedingtes Kriterium: Für einige Simulationen ist es nicht zweckmäßig, das Verfeinerungskriterium auf das gesamte Strömungsgebiet anzuwenden. Als Beispiel ist hierfür eine Simulation zu nennen, dessen potentielles Nachlaufgebiet extrem lang modelliert wurde. Ist man lediglich am c_D-Wert interessiert, genügt es sicherlich, einen angemessenen Bereich um das entsprechende Hindernis anzugeben. In der Implementierung dieser Arbeit müssen daher mögliche Verfeinerungszonen VZ definiert werden.

Zwangskriterium: Wie bereits in Kapitel 7.2 beschrieben, unterliegt die Gittergenerierung einigen Einschränkungen. Einfluss auf eine Zwangsverfeinerung ZV hat vor allem die *modifizierte Glättung*. Es ist daher möglich, dass Zellen in Bereichen verfeinert werden, deren Sensorenwerte nicht das Verfeinerungskriterium erfüllen und deren Lage sich nicht mit den Verfeinerungszonen überschneiden.

Eine Gitterzelle GZ wird letzten Endes dann verfeinert, wenn folgende Gesamtbedingung erfüllt ist:

$$\Big(\Phi(GZ) > \tau \quad \text{UND} \quad GZ \in VZ\Big) \quad \text{ODER} \quad ZV \tag{8.11}$$

8.5 Mechanismus der Gitteradaption

Im Folgenden werden die Teilschritte einer adaptiven Gittererweiterung exemplarisch anhand des Gitterausschnittes in Abbildung 8.2 erklärt.

1. **Markierung der zu verfeinernden Zellen:**
   Im ersten Schritt wird die Sensorfunktion für alle Zellen des Strömungsgebietes berechnet und gegebenenfalls markiert (vgl. Kapitel 8.4). In Abbildung 8.2 wird davon ausgegangen, dass die Sensorenwerte der grau hinterlegten Zellen das Kriterium erfüllt haben und verfeinert werden sollen.

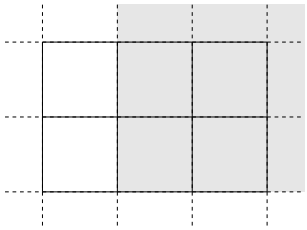

Abbildung 8.2: Markierung der zu verfeinernden Zellen (grau)

2. **Eigenschaften zurücksetzen:**
   In dieser Implementierung werden Knoten, die skaliert, interpoliert, etc. werden müssen, in Listen gehalten. Durch eine Verfeinerung kann es vorkommen, dass einige Knoten, die im ursprünglichen Gitter Teil des Interfaces waren, jetzt außerhalb des Interfaces liegen und entsprechend nicht mehr gesondert behandelt werden müssen. Gleichzeitig kann es passieren, dass neue Interfaces erzeugt werden. Da die Schritte Klassifizierung der Knoten, Füllen der Listen für die Rekonstruktion des Gitters (Skalierungen, Interpolationen) und der Vorbereitung für die Haftbedingungen im Vergleich zur Berechnung der Strömung selbst nur kaum ins Gewicht fallen, werden die entsprechenden Listen und Werte zurückgesetzt und nach der Gittererweiterung neu gefüllt.

3. **Verfeinerung der markierten Zellen:**
   Als nächstes folgt die eigentliche Zellverfeinerung (vgl. Abb. 8.3). Alle Zellen, die markiert wurden, werden verfeinert und in das Zeigergeflecht der Baumdatenstruktur eingebunden (vgl. Kapitel 3.3.4).

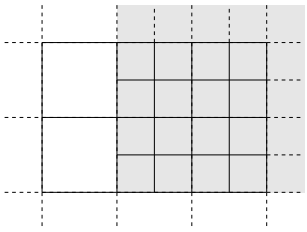

Abbildung 8.3: Verfeinerung der markierten Zellen

4. **Gitterrekonstruktion:**
   Ähnlich wie im Kapitel 6 beschrieben, muss die Konsistenz des Gitters gewährleistet werden. Im Gegensatz zu den Rekonstruktionsschritten, die während der Strömungsberechnung am Interface notwendig sind, muss hier das Gitter und die Simulation bezüglich der Konsistenz der Datenstruktur und des Simulationsablaufs erweitert werden.

   (a) **Horizontale Verzeigerung korrigieren:**
   Durch eine Verfeinerung von Gitterzellen, die sich mit geometrischen Objekten schneiden, können Gitterzellen entstehen, die vollständig innerhalb der geometrischen Objekte liegen. Diese Zellen werden im Laufe der Gitterkorrektur ignoriert und erhalten keine Gitterknoten. Um zu verhindern, dass auf diese Knoten im weiteren Verlauf zugegriffen wird, muss die horizontale Verzeigerung auf jene Gitterzellen beschränkt werden, die zumindest in direktem Kontakt mit dem Fluid stehen, also mindestens einen Zeiger auf einen Fluid-Knoten haben.

   (b) **Glättung:**
   Wegen der im Wesentlichen durch die Strömungslösung vorgegebenen Verfeinerung, kann es zu sehr komplexen Gitterkonstellation kommen. Dies stellt eine wirkliche Herausforderung an die Glättungsfunktionen dar, die die Aufgabe haben die Datenstruktur bezüglich den Anforderungen des Simulationsalgorithmus konsistent zu halten.

   (c) **Gitterknoten erzeugen:**
   In der Abbildung 8.4 sind die neu erzeugten Knoten als gefüllte Kreise dargestellt. Diese müssen natürlich erst erzeugt und dann entsprechend in die Knotenlisten eingetragen werden.

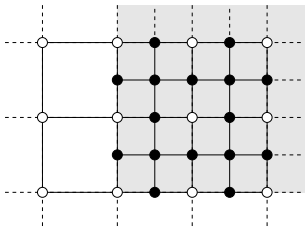

Abbildung 8.4: Neue Gitterknoten (gefüllte Kreise)

(d) **Simulationsspezifische Rekonstruktionen:**
Alle Listen, die für den Multiskalen-Algorithmus (vgl. Kapitel 6.4) notwendig sind, müssen neu angelegt werden. Dies betrifft in der Hauptsache Knoten und Zellen, die sich im Bereich des Interfaces befinden (vgl. Abb. 8.5).

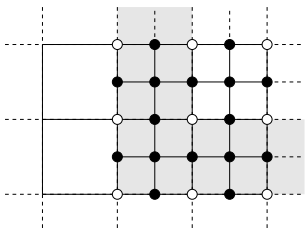

Abbildung 8.5: Verfeinertes Gebiet mit Interface (grau)

Zudem müssen die Verteilungen der Gitterknoten in jenen Bereichen, die verfeinert wurden, zusätzlich angepasst werden:

i. **Skalierung:**
Die Baumtiefe aller Knoten[1] der Abbildung 8.5, die durch nicht gefüllte Kreise dargestellt sind, wurde um eins erhöht. Somit sind die Verteilungen der Knoten nun dem höher aufgelösten Gitterlevel zugehörig und müssen gemäß den Gleichungen 6.17 und 6.18 skaliert werden.

ii. **Räumliche Interpolation:**
Die schwarz gefüllten Gitterknoten sind neu und werden durch lineare Interpolation, wenn der neue Knoten genau zwischen zwei alten Knoten liegt, und durch bilineare Interpolation[2], wenn der neue Knoten im Schwerpunkt der verfeinerten Zelle liegt, rekonstruiert.

[1] Jedem Knoten wird als Attribut eine Baumtiefe, stellvertretend für die Auflösung, zugewiesen.

[2] In diesem speziellen Fall entartet die bilineare Interpolation zu einer Mittelwertbildung.

Nach Abschluss des Rekonstruktionsmechanismus kann die Simulation mit dem modifizierten Gitter fortgeführt werden.

8.6 Klassenkonzept

Die adaptive Simulation ist eine echte Obermenge der auf einer a priori Verfeinerung basierten Multiskalen-Simulation. Alle wichtigen Funktionen des Strömungslösers und auch viele Initialisierungsroutinen können übernommen werden. Gleichzeitig muss der Rahmenalgorithmus um die Adaptivität erweitert werden. In der objektorientierten Modellierung des Prototypen wurde daher eine neue Klasse *adaptiveSimulation* von der Klasse *LBSimulation* abgeleitet (vgl. Abb. 8.6). Auf diese Art und Weise konnten bereits implementierte und auch

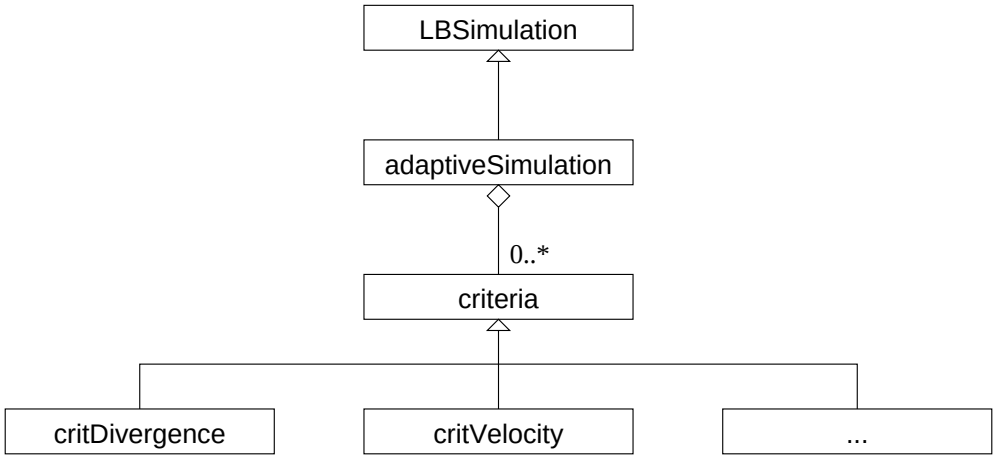

Abbildung 8.6: Adaptiver LB-Simulator: Klassenkonzept

häufig angewandte Funktionen übernommen werden.

8.7 Beispiele

8.7.1 Stokes-Strömung

In der folgenden adaptiven Simulation wird die Stokes-Strömung, auch Kriechströmung genannt, untersucht. Kriechströmungen zeichnen sich dadurch aus, dass die viskosen Kräfte im Fluid dominieren. Die Reynoldszahl ist typischerweise sehr klein ($Re <= 1.0$). Als Beispiel wird ähnlich wie in Kapitel 7.6.3 beschrieben eine Zylinderumströmung im freien Kanal gewählt. Exemplarisch wird die Simulation bei einer Reynoldszahl von 1.0 durchgeführt.

Bei dieser Strömungsart besteht die Problematik darin, dass aufgrund der großen viskosen Kräfte, der Einfluss von Randbedingungen oder allgemeiner gesagt, von Begrenzungen der Fluid-Domäne auf den Zylinder sehr groß ist und eine *ungestörte* Berechnung von Auswertedaten, wie beispielsweise der $c_D - Wert$, nur dann möglich ist, wenn der Durchmesser des Berechnungsgebiets bezogen auf den Zylinderdurchmesser sehr groß gewählt wird. In [65] wurden diese Effekte studiert und das Verhältnisse aus Kanalhöhe H zu Zylinderdurchmesser D so angegeben, dass der Fehler $\epsilon^{cD}_{\frac{H}{D}}$ in der c_d-Wert-Berechnung unterhalb der Schranke von 1% liegt:

$$\frac{H}{D} > 320 \cdot Re^{-0.8} \quad \text{für} \quad \epsilon^{c_D}_{\frac{H}{D}} < 1\% \tag{8.12}$$

8.7.1.1 Simulations-Setup

Am Einlaufrand wird ein über die Höhe des Kanals konstantes Geschwindigkeitsprofil vorgegeben. Der Auslaufrand wird mit einem konstanten Druck belegt. Die obere und die untere Kanalbegrenzung kann mit einer Slip-Randbedingung versehen werden. Da allerdings davon ausgegangen werden kann, dass in diesem Bereich nur sehr kleine Gradienten in den Strömungsvariablen vorliegen, werden die beiden Ränder in Form einer periodischen Randbedingungen miteinander verbunden. Der Vorteil liegt darin, dass keine numerischen Fehler durch die Randbedingung selbst induziert werden können.

Mit der Gleichung (8.12) wird für $Re = 1.0$ das Verhältnis $\frac{H}{D}$ zu

$$\frac{H}{D} = 320 \cdot 1.0^{-0.8} = 320 \tag{8.13}$$

bestimmt. Für eine aussagekräftige c_D-Wert-Berechnung (vgl. Kapitel 7.5.1.4) sollte der Zylinderdurchmesser auf dem feinsten Gitter mindestens $D = 50 \cdot \Delta x_{TD_{max}}$ betragen. Die Kanalhöhe H ergibt sich damit zu:

$$H = 50 \cdot 320 \cdot \Delta x_{TD_{max}} = 16000 \cdot \Delta x_{TD_{max}} \tag{8.14}$$

Unter der Annahme, dass genau ein Wurzelelement vorliegt, wird die Anzahl der notwendigen Baumtiefen (TD_{max}) mit

$$2^{TD_{max}} = H = 16000 \quad \Rightarrow \quad TD_{max} = 13,96 \approx 14 \tag{8.15}$$

berechnet. Die minimale Baumtiefe wird mit

$$TD_{min} = 7 \tag{8.16}$$

abgeschätzt. Das Berechnungsgebiet besteht demnach aus acht verschiedenen Gitterebenen. Die Viskosität ν wird so gewählt, dass die Relaxationszahlen auf dem

feinsten Gitterlevel 1.0 ergeben. Hierfür wird zunächst Formel 6.32 nach ν aufgelöst und $\Delta TD = 7$ und $s = 1.0$ eingesetzt:

$$\nu = \frac{1}{3} \cdot \left(\frac{1}{2}\right)^{\Delta TD} \cdot \left(\frac{1}{s} - \frac{1}{2}\right) = 1.3020833333 \cdot 10^{-3} \tag{8.17}$$

Auf dem am gröbsten aufgelösten Gitter ($TD = 7$) resultiert somit eine Relaxationszahl von

$$s_{max} = 1.9845 \tag{8.18}$$

Damit befindet man sich bereits in einem Bereich, der sehr schnell zu Instabilitäten aufgrund der hohen Relaxationzahl neigt. Da die Gradienten jedoch sehr gering sind, bleibt das Verfahren für diesen konkreten Fall stabil. Die Momentenmethode bringt hier keinen Vorteil, da die minimalen Relaxationszahlen $s_{min} = 1.0$ bereits minimal und damit am stabilsten gewählt wurden (vgl. Kapitel 6.6.2). Der Zylinderdurchmesser ergibt sich aufgrund der gewählten Eingabekoordinaten zu $D = 49.152 \cdot \Delta x_{TD_{max}}$. Die Geschwindigkeit u_0 wird mit

$$u_0 = \frac{Re \cdot \nu}{D \cdot \Delta x_{TD_{max}}} \cdot 2^{\Delta TD} = 3.3908420139 \cdot 10^{-3} \tag{8.19}$$

berechnet.

Die adaptive Gittererweiterung wird vom *Divergenz-Sensor* gesteuert.

8.7.1.2 Auswertung

Die Abbildung 8.7 zeigt das initiale Gitter. Wie in der gezoomten Darstellung zu sehen ist, kann die Geometrie des Zylinders nur durch die geometrische Annäherung mit vielen Gitterebenen ausreichend gut diskretisiert werden.

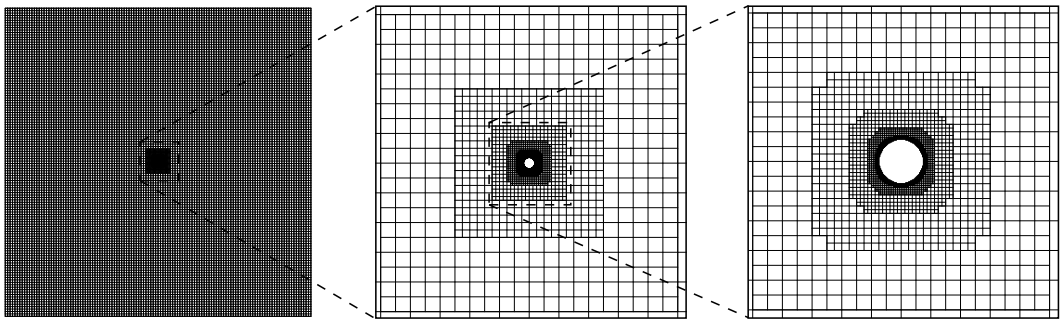

Abbildung 8.7: Stokes-Strömung: Initiales Gitter

Im Verlauf der adaptiven Simulation erfolgen vier Adaptionsschritte, deren Ergebnisse in Tabelle 8.1 nachzulesen sind. Die Berechnung der Widerstandsbeiwerte erfolgt mit der Momentenmethode. Die Referenzlösung des c_D-Wertes ist in [65] mit $c_D = 10.408(\pm 1\%)$ angegeben:

Adaptionen	$c_D[]$	$E_{rel}[\%]$	$Fluid - Knoten$
nicht-adaptiv	10.091	3.045	19515
1	10.405	0.028	24166
2	10.460	0.499	30179
3	10.464	0.538	31036
4	10.465	0.547	31537

Tabelle 8.1: Stokes-Strömung: c_D-Werte, relativer Fehler und Knotenzahlen

Nach der ersten Adaption ist die beste Lösung erhalten worden. Dies ist jedoch lediglich ein Zwischenschritt. Der Fehler der Lösung konnte letztendlich von 3.0% bis zu 0.5% reduziert werden und befindet sich damit innerhalb der geforderten 1%-Schränke Die Gitterkonfiguration des adaptiven Gitters nach der letzten Adaption ist der Abbildung 8.8 zu entnehmen.

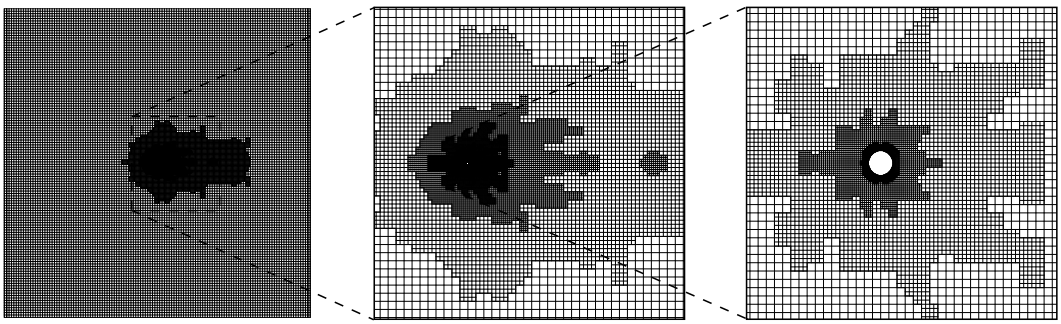

Abbildung 8.8: Stokes-Strömung: Adaptives Gitter

Qualitative Bewertung: Bei Betrachtung von Abbildung 8.9 wird deutlich, dass das Druckfeld bei Verwendung eines lediglich geometrisch angepassten Gitters große Diskontinuitäten aufzeigt. Dies ist dadurch begründet, dass das Gitter nicht in der Lage ist, verstärkt durch die extremen Relaxationparameter, den wahren Verlauf der Strömung zu rekonstruieren. Nach dem adaptiven Algorithmus dagegen wurden diese Zonen durch den Divergenz-Sensor detektiert und verfeinert. Das Ergebnis ist der Abbildung 8.10 zu entnehmen, in welcher die Isolinien des Druckes einen sehr glatten Verlauf haben. In Abbildung 8.11 sind die Diskontinuitäten im Druckverlauf in einer orthogonale Ansicht eines *Surfplots*[3] dargestellt. Sehr gut zu sehen ist, dass das adaptive Gitter den Druckgradienten in einer harmonischen Art und Weise wiedergibt.

[3]Die Überhöhung entspricht dem Wert des Druckes an der entsprechenden Position im Gitter.

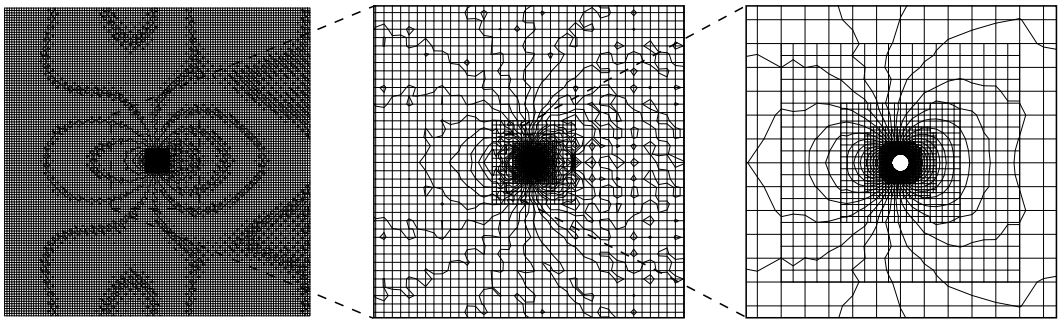

Abbildung 8.9: Stokes-Strömung: Isolinien der Berechnung auf einem geometrisch angenäherten Gitter ohne adaptiven Zyklus

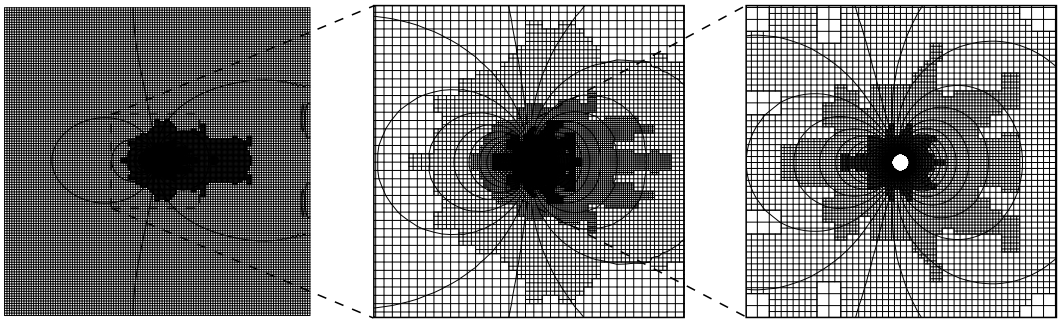

Abbildung 8.10: Stokes-Strömung: Isolinien der adaptiven Simulation

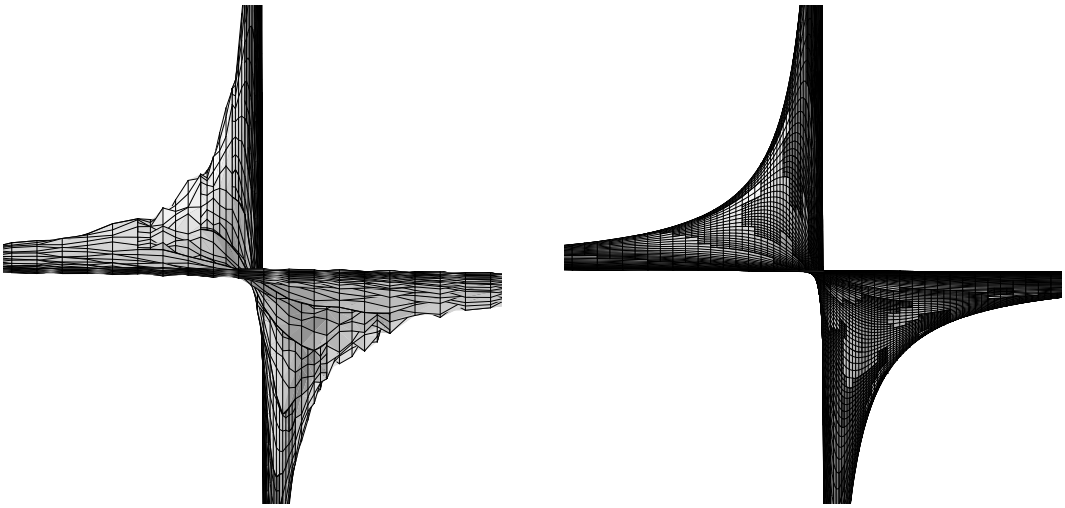

Abbildung 8.11: Stokes-Strömung: Orthogonale Darstellung (Surfplot) des Druckverlaufs in Zylindernähe; links: geometrisch angenähertes Gitter, rechts: adaptives Gitter

Zu bemerken sei an dieser Stelle, dass die Berechnung auf einem uniformen Gitter circa $268 \cdot 10^6$ Knoten erfordern würde. Dies ist selbst auf Höchstleistungsrechnern derzeit nicht oder nur schwer umsetzbar.

8.7.2 Generisches, poröses Medium

In [4, 9] wird ein zweidimensionaler Testfall verwendet, um die Eigenschaften von Finite-Volumen Navier-Stokes und Lattice-Boltzmann Methoden hinsichtlich ihrer Genauigkeit und Effizienz zu bewerten. Es handelt sich um ein generisches, poröses Medium, wie es in Abbildung 8.12 dargestellt ist. Da in y-Richtung eine

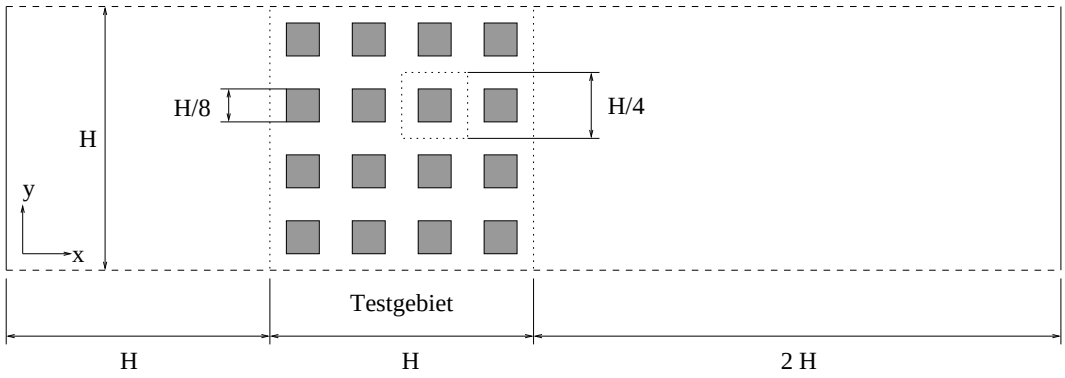

Abbildung 8.12: Poröses Medium: Geometrie

TD	$\Delta p[]$	$E_{rel}[\%]$	Knoten
7: uniform	14.645	0.000	61040
4-7: 0 Adaptionen	14.075	3.892	8080
4-7: 1 Adaptionen	14.628	0.116	13632
4-7: 2 Adaptionen	14.632	0.088	15364
4-7: 3 Adaptionen	14.631	0.095	15572

Tabelle 8.2: Poröses Medium (LBGK): Angabe des Druckverlustes, des relativen Fehlers und der Knotenanzahl

periodische Randbedingung gesetzt wird, kann nur zwischen dem porösen Medium und dem Fluid ein Impulsaustausch stattfinden, welcher zu einem Druckgradienten in x-Richtung führt. Die sogenannten *Ergun'sche Reynoldszahl* [48] für diesen Testfall wird mit

$$Re_E = \frac{u_0 \cdot D_p}{\nu \cdot (1 - \epsilon)} \tag{8.20}$$

berechnet. Die Porösität ϵ definiert den Anteil des Fluids am Gesamtvolumen und wird für dieses poröse Medium zu 0.75 berechnet. u_0 ist die konstante Einströmgeschwindigkeit und D_p der Partikeldurchmesser, welcher mit $\frac{H}{8}$ angenommern wird.

In dieser Untersuchung wird der Genauigkeitsgewinn, bedingt durch die Adaptivität des vorgestellten Verfahrens, betrachtet. Hierfür dient in erster Linie der in y-Richtung lokal gemittelte Druckverlauf, welcher dann als eindimensionale Kurve dargestellt werden kann.

8.7.2.1 Simulations-Setup

Zunächst wird eine Referenzlösung mit einem uniformen Gitter der Auflösung $TD = 7$ erzeugt, anhand derer in der Auswertung die relativen Fehler berechnet werden. Die adaptive Simulation beginnt mit einem nicht-uniformen Gitter mit den Gitterebenen 4 bis 7, wobei die Hindernisobjekte a priori verfeinert sind. Der obere und der untere Rand des Strömungsgebietes sind durch die periodischen Randbedingungen verbunden. Die Ergun-Reynoldszahl wird zu $Re_E = 50$ gewählt. Als Verfeinerungsindikator dient der Divergenzsensor.

8.7.2.2 Auswertung

Die Tabelle 8.2 zeigt, dass der relative Fehler schon nach dem ersten Verfeinerungsschritt drastisch reduziert wurde. Es wurden annähernd gleichwertige Lösungen erzeugt, wobei der adaptive Simulationslauf nur etwa ein Viertel der Knoten benötigte wie der Simulationslauf der uniformen Referenzlösung. In Abbildung 8.13 sind die gemittelten Druckverläufe dargestellt. Da hier der Druck im Auslauf definitionsgemäß Null ist, kann die dimensionlose Druckdifferenz im Bereich vor dem porösen Medium auf der Ordinate abgelesen werden.

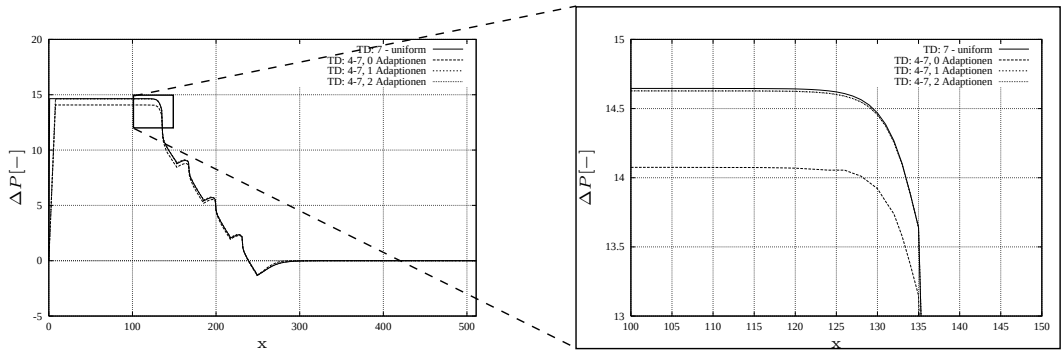

Abbildung 8.13: Poröses Medium (LBGK): gemittelter Druckverlauf

Das anfängliche, geometrisch approximierte Gitter und die Gitter nach der ersten und der zweiten Adaption werden in den Abbildungen 8.14 bis 8.16 dargestellt.

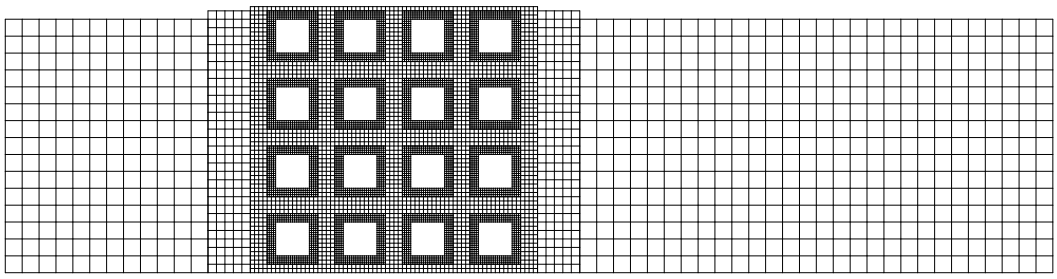

Abbildung 8.14: Poröses Medium: Anfangsgitter

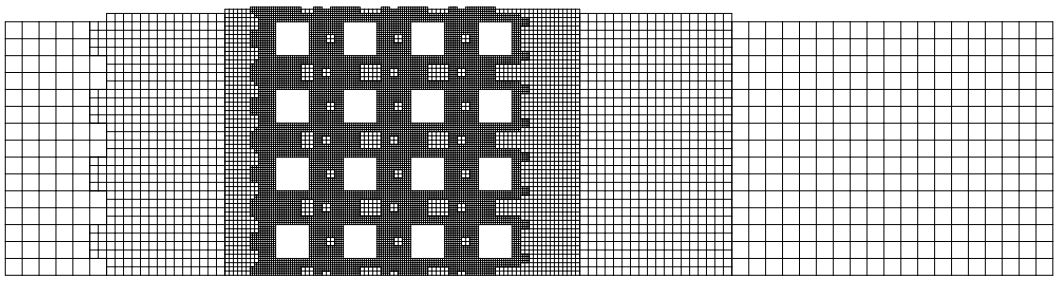

Abbildung 8.15: Poröses Medium: Gitter nach erster Adaption

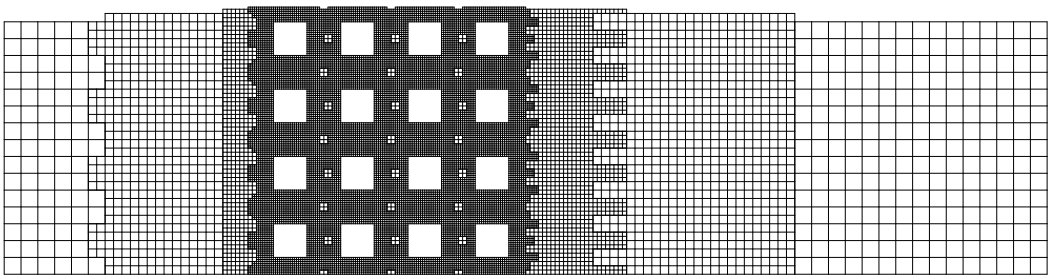

Abbildung 8.16: Poröses Medium: Gitter nach zweiter Adaption

Qualitative Bewertung: Für die qualitative Bewertung wird die Darstellung des Druckes in überhöhter Darstellung (Surfplot) gewählt. Im linken Bild der Abbildung 8.17 sind deutlich Oszillationen im Bereich der Gitterübergänge der gröberen Gitter zu erkennen. Diese sind, wie im rechten Bild der Abbildung 8.17 zu sehen ist, durch die Adaptionen im Wesentlichen eliminiert.

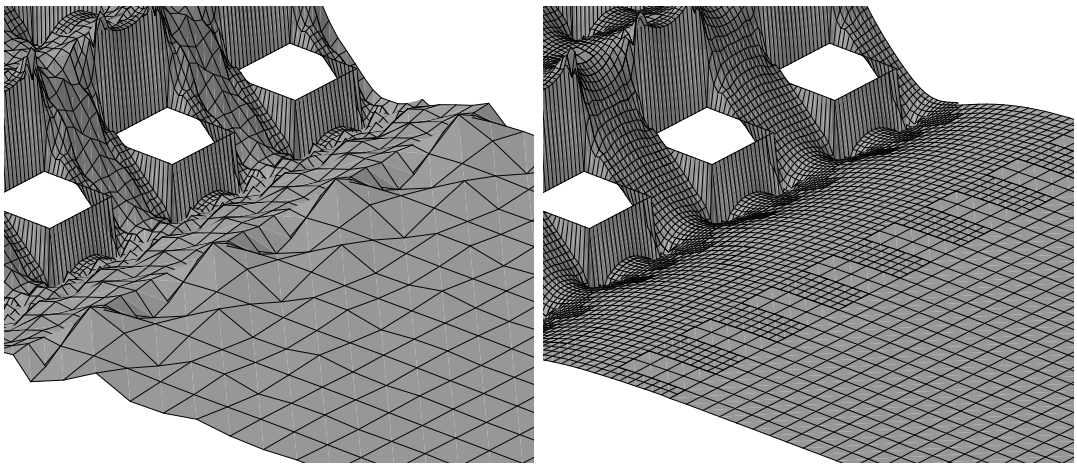

Abbildung 8.17: Poröses Medium: Überhöhte Darstellung des Druckverlaufs im Bereich des porösen Medium; links: Druckverlauf des nicht-adaptiven Gitters (vgl. Abb. 8.14); rechts: Druckverlauf der adaptiven Endlösung (vgl. Abb. 8.16)

8.8 Zusammenfassung

In diesem Kapitel wurde ein adaptiver Lattice-Boltzmann Algorithmus vorgestellt. Hierfür war zunächst die Einführung von Sensoren und Kriterien erforderlich, um die Verfeinerung zu steuern. Der Mechanismus der Gitteradaption besteht im Wesentlichen aus den folgenden Schritten:

- Zellen des Gitters müssen in Abhängigkeit von einer Zwischenlösung mithilfe von Sensoren und Kriterien für eine Verfeinerung markiert werden.
- Die markierten Zellen werden verfeinert und in die vorhandene Datenstruktur eingebunden.
- Das Gitter, vor allem die simulationsspezifischen Listen, wie z. B. Skalierungs- und Interpolationslisten müssen gegebenenfalls initialisiert und aktualisiert werden.
- Die Simulation kann jetzt fortgeführt werden.

Das selbstorganisierende Gitter ist mithilfe der Sensoren und des adaptiven Zyklus in der Lage, jene Bereiche im Gitter, in denen eine unzureichende Gitterdichte zu numerischen Oszillationen führen kann, aufzuspüren und zu eliminieren und somit die Qualität des Gitters, der Verteilung der Freiheitsgrade und der Lösung verbessern. Weiterhin wurde der Grad an Expertenwissen, das meist für anspruchsvolle Simulationen benötigt wird, durch die Selbstorganisation des Gitters reduziert.

Kapitel 9

Zusammenfassung und Ausblick

In dieser Arbeit wurde ein Integrationskonzept entworfen, entwickelt und implementiert, welches den Simulationsablauf im Bereich der Strömungsmechanik optimieren kann. Der Nutzen macht sich vorrangig in der Kopplung der Teilprozesse in Strömungssimulationen, dem Preprocessing, der Berechnungsphase und dem Postprocessing bemerkbar. Als zentrales Bindeglied zwischen den Teilprozessen fungieren die sogenannten Spacetrees, welche durch ihr raumorganisierendes Prinzip die gegenseitigen räumlichen Abbildungen der verschiedenen geometrischen Modelle sehr gut unterstützen. Gleichzeitig eignen sich die Baumdatenstrukturen hervorragend zur Darstellung von kartesischen Gittern sowie zu deren Generierung. Es wurde gezeigt, dass alle Schritte der Gesamtablaufskette, beginnend mit dem attributierten CAD-Modell bis zur Simulation vollständig automatisierbar sind. Hierbei wurde zunächst ein attributiertes Facettenmodell vom CAD-Modell abgeleitet und daraus anschließend ein Octree-basiertes Berechnungsgitter erzeugt. Da alle relevanten Informationen durchgängig in den Teilmodellen abrufbar sind, ist eine Erweiterung dieses Rahmenwerkes in zukünftigen Arbeiten, beispielsweise in Form eines bidirektionalen Informationsaustauschs mit anderen Simulationswerkzeugen im Rahmen einer Fluid-Struktur-Interaktion denkbar.

Im zweiten Teil dieser Arbeit wurden die Grundlagen für Lattice-Boltzmann Simulationen auf nicht-uniformen Gittern für zweidimensionale Probleme detailliert beschrieben. Der explizite Lattice-Boltzmann Algorithmus wurde für Simulationen, basierend auf Multiskalen-Gittern, unter Beibehaltung des Grundkonzepts, erweitert. Die Gitterabhängigkeiten wurden sowohl für den LBGK-Algorithmus als auch für den GLBE-Algorithmus untersucht. Im Design und in der Implementierung des prototypischen Simulators wurde auf objektorientierte Programmiertechniken Wert gelegt, um eine notwendige Kapselung der Teilaufgaben des Simulators zu erreichen und den Code erweiterbar zu gestalten. Es wurde deutlich gemacht, dass die Umsetzung der Randbedingungen und der Evaluierungsmethoden im starkem Maße von der Implementierung der Datenstruktur des Berechnungsgitters abhängt. Das modifizierte Verfahren wurde für zweidi-

mensionale Strömungen verifiziert und zeigt für geometrisch angepasste Gitter sehr gute Ergebnisse.

Das vorgestellte objektorientierte Konzept unterstützt die Erweiterung des Simulators für adaptive Simulationen. Die Verwendung von selbstadaptiven Gittern verbessert die Ergebnisse und ermöglicht sie teilweise sogar erst, da aufgrund von Instabilitäten, die bei Verwendung sehr vieler Gitterlevel auftreten, zwar Lösungen erzielt werden können, diese aber qualitativ unakzeptabel sein können. Das Ingenieurwesen verlangt nach adaptiven Simulationen, um Computerresourcen ohne Expertenwissen besser und effizienter einsetzen zu können. Diese Arbeit hat einen ersten Schritt in diese Richtung getan und gezeigt, dass sich die Lattice-Boltzmann Methode grundsätzlich für eine Erweiterung in Richtung Adaptivität eignet. Aufbauend auf dieser Arbeit ist der nächste Schritt die Transition der vorgestellten Algorithmen auf dreidimensionale Lattice-Boltzmann Modelle. Es kann davon ausgegangen werden, dass der Effizienzgewinn in adaptiven Strömungssimulationen in 3D noch deutlich höher ist, als in zweidimensionalen Fragestellungen.

Literaturverzeichnis

[1] Introduction to AVS/Express. Schulungsmappe, 1996.

[2] J. J. Barton and L. R. Nackman, editors. *Scientific and Engineering C++*. Addison-Wesley, 1995.

[3] M. Bender, S. Dendorfer, K. Kießlinger, and J. Stobinski. Parallelization of the calculation kernel of an interactive tool for 2D-fluid flow simulations using MPI. Jahrespraktikum, Lehrstuhl für Bauinformatik, TU München, 2003.

[4] J. Bernsdorf, G. Brenner, and F. Durst. Numerical analysis of the pressure drop in porous media flow with lattice Boltzmann (BGK) automata. *Computer Physics Communications*, 129:247 – 255, 2000.

[5] P. Bhatnagar, E. Gross, and M. Krook. A model of collision processes in gases. *Physical Review Letters*, 94(511-525), 1954.

[6] M. Bouzidi, D. dHumieres, P. Lallemand, and L.-S. Luo. Lattice Boltzmann Equation on a Two-Dimensional Rectangular Grid . *Journal of Computational Physics*, 172:704 – 717, 2001.

[7] M. Bouzidi, M. Firdaouss, and P. Lallemand. Momentum transfer of a Boltzmann-lattice fluid with boundaries. *Physics of Fluids*, 13:3452 – 3459, 2001.

[8] P. Breitling, H.-J. Bungartz, and A. Frank. Hierarchical Concepts for Improved Interfaces between Modelling, Simulation and Visualization. In *Preprint, SFB-438-9912*, 1999.

[9] G. Brenner. Numerische Simulation komplexer fluider Transportvorgänge in der Verfahrenstechnik. Habilitationsschrift, 2002.

[10] H. Bröker. *Integration von geometrischer Modellierung und Berechnung nach der p-Version der FEM.* Dissertation, Lehrstuhl für Bauinformatik, TU München.

[11] H. J. Bungartz, M. Griebel, and C. Zenger. *Einführung in die Computergraphik*. Vieweg Verlag, Wiesbaden, 1996.

[12] S. Chapman and T. Cowling. *The Mathematical Theory of Non-Uniform Gases*. Cambridge University Press, 1990.

[13] H. Chen. Persönliche Mitteilung, August 2002.

[14] H. Chen, C. Teixeira, and K. Molvig. Realization of Fluid Boundary Conditions via Discrete Boltzmann Dynamics. *International Journal of Modern Physics C*, 9(8):1281–1292, 1998.

[15] B. Crouse. Automatische Gittergenerierung für zweidimensionale Gebiete unter Verwendung von Quadtree-Datenstrukturen. Diplomarbeit, Lehrstuhl für Bauinformatik, TU München, 1999.

[16] B. Crouse. Baumdatenstrukturen in strömungsmechanischen Simulationen. In *Forum Bauinformatik 2000*, Berlin, 2000.

[17] B. Crouse and S. Kühner. Visualisierung von Finite-Element-Berechnungsergebnissen unter Benutzung des Softwarepaketes AVS/Express. Master's thesis.

[18] B. Crouse and S. Kühner. Simulationen und Visualisierung von Windströmungen um Bauwerke. In *Forum Bauinformatik 1999*, Darmstadt, Germany, 1999.

[19] B. Crouse, M. Krafczyk, S. Kühner, E. Rank, and C. van Treeck. Indoor air flow analysis based on lattice Boltzmann methods. *Energy and Buildings*, 34:941 – 949, 2002.

[20] B. Crouse, M. Krafczyk, J. Tölke, and E. Rank. A LB-based approach for Adaptive Flow Simulations. *International Journal of Modern Physics B*, 2003.

[21] D. d 'Humieres, I. Ginzburg, M. Krafczyk, P. Lallemand, and L.-S. Luo. Multiple-relaxation-time lattice Boltzmann models in three dimensions. *Philosophical Transactions of the Royal Society of London A*, 360:437 – 451, 2002.

[22] F. Deister and E. H. Hirschel. Self-Organizing Hybrid Cartesian Grid / Solution System with Multigrid. Reno, NV, 2002.

[23] F. Deister, F. Waymel, E. H. Hirschel, and F. Monnoyer. Self-Organizing Hybrid Cartesian Grid Generation and Application to External and Internal Flow Problems. *Numerical Flow Simulation III. Notes on Numerical Fluid Mechanics and Multidisciplinary Designs*, 82:18 – 29, 2002.

[24] F. J. Deister. *Selbstorganisierendes hybrid-kartesisches Netzverfahren zur Berechnung von Strömungen um komplexe Konfigurationen.* Dissertation, Institut für Aero- und Gasdynamik, Universität Stutgart, 2002.

[25] A. Düster. *High order finite elements for three-dimensional, thin-walled nonlinear continua.* Dissertation, Lehrstuhl für Bauinformatik, TU München, 2002.

[26] A. Dupuis. *From a lattice Boltzmann model to a parallel reusable implementation of a virtual river.* Dissertation, University of Geneve, Departement of informatics, 2001.

[27] J. H. Ferziger and M. Peric, editors. *Computational Methods for Fluid Dynamics.* Springer Verlag, Berlin/Heidelberg, 1999.

[28] O. Filippova and D. Hänel. Boundary-Fitting and local Grid Refinement for Lattice-BGK Models. *International Journal of Modern Physics C*, 9:1271–1279, 1998.

[29] O. Filippova and D. Hänel. Grid refinement for Lattice-BGK models. *Journal of Computational Physics*, 147, 1998.

[30] O. Filippova and D. Hänel. A novel Lattice BGK approach for low mach number combustion. *Computational Physics*, 2(158):139–160, 2000.

[31] O. Filippova, S. Succi, F. Mazzocco, C. Arrighetti, G. Bella, and D. Hänel. Multiscale Lattice Boltzmann Schemes with Turbulence Modeling. *Journal of Computational Physics*, 170:812–829, 2001.

[32] J. Fischer. *Selbstadaptive, lokale Netzverfeinerung für die numerische Simulation kompressibler, reibungsbehafteter Strömungen.* Dissertation, Institut für Aero- und Gasdynamik, Universität Stutgart, 1993.

[33] A. Frank. *Organisationsprinzipien zur Integration von geometrischer Modellierung, numerischer Simulation und Visualisierung.* Dissertation, Institut für Informatik, TU München, 2001.

[34] S. Freudiger. Effiziente Datenstrukturen für Lattice-Boltzmann-Simulationen in der computergestützen Strömungsmechanik. Diplomarbeit, Lehrstuhl für Bauinformatik, TU München, 2001.

[35] R. Geiger. 3D Simulation of Granular Materials. Master's thesis, Lehrstuhl für Bauinformatik, TU München, 2002.

[36] P. L. George. *Automatic Mesh Generation - Application to Finite Element Methods.* John Wiley & Sons, Chichester, 1991.

[37] M. Griebel, T. Dornseifer, and T. Neunhoeffer, editors. *Numerische Simulation in der Strömungsmechanik*. Vieweg Verlag, Braunschweig/Wiesbaden, 1995.

[38] A. Gun, V. Neugebauer, and J. Tondorf. Datenreduktion mithilfe der Quadtree-Datenstruktur. Jahrespraktikum, Lehrstuhl für Bauinformatik, TU München, 2000.

[39] A. Halfmann. *Ein geometrisches Modell zur numerischen Simulation der Fluid-Struktur-Interaktion windbelasteter, leichter Flächentragwerke*. Dissertation, Lehrstuhl für Bauinformatik, TU München, 2002.

[40] P. Hardt. Entwicklung eines Moduls zur Definition von strömungsmechanischen Randbedingungen als Attribute von 3D CAD-Geometrien. Diplomarbeit.

[41] P. Hardt and B. Crouse. Präprozessor für einen computergestützten Windkanal. In *Forum Bauinformatik Bochum*, Düsseldorf, 2002.

[42] X. He and L.-S. Luo. Lattice Boltzmann Model for the Incompressible Navier-Stokes Equation. *Journal of Statistical Physics*, 88:927 – 944, 1997.

[43] X. He, Q. Zou, L.-S. Luo, and M. Dembo. Analytic Solutions of Simple Flows and Analysis of Nonslip Boundary Conditions for the Lattice Boltzmann BGK Model. *Journal of Statistical Physics*, 87:115–136, 1995.

[44] R. Hentschel. *Entwicklung und Anwendung eines dreidimensionalen selbstadaptiven Verfahrens zur Simulation viskoser Strömungen auf der Basis strukturierter Gitter*. Dissertation, Institut für Aero- und Gasdynamik, Universität Stutgart.

[45] T. Hodohara, J. Matsumoto, and M. Kawahara. Analysis of the Taylor Vortex Flow. 2000.

[46] S. Hou, J. Sterling, S. Chen, and G. Doolen. A lattice subgrid model for high Reynolds number flows. *Fields Institute Communications*, 6:151–166, 1996.

[47] S. Jaksch. Facettierung dreidimensionaler Gebiete und Gittergenerierung unter Verwendung von Octree-Datenstrukturen. Diplomarbeit, Lehrstuhl für Bauinformatik, TU München, 2001.

[48] M. Kaviany, editor. *Principles of Heat Transfer in Porous Media*. Springer Verlag, Berlin/Heidelberg, 1995.

[49] A. Kela. Hierarchical octree approximations for boundary representation-based geometric models. *Computer-Aided Design*, 21(6):355–362, 1989.

[50] S. Kühner. Visualisierung von Ergebnissen numerischer Simulationen von Luftströmungen in und um Bauwerke in einer Virtual Reality Umgebung. Diplomarbeit, Lehrstuhl für Bauinformatik, TU München, 1999.

[51] S. Kühner. *Virtual Reality basierte Analyse und interaktive Steuerung von Strömungssimulationen im Bauingenieurwesen*. Dissertation, Lehrstuhl für Bauinformatik, TU München, 2003.

[52] S. Kühner and B. Crouse. Visualisierung von Ergebnissen numerischer Simulationen in der Strömungsmechanik unter Verwendung von hierarchischen Datenstrukturen. In *Forum Bauinformatik in München*, Düsseldorf, 2001.

[53] S. Kühner, B. Crouse, M. Krafczyk, J. Tölke, and E. Rank. From a building product model to the visualization of simulation data: Computation of indoor flow based on Lattice-Boltzmann-Methods. *eingereicht bei: CACAIE*, 2003.

[54] S. Kühner and M. Krafczyk. VirtualFluids - An environment for integral visualization and analysis of CAD and simulation data. In *Vision, Modeling, and Visualization 2000*, Saarbrücken, 2000.

[55] S. Kühner, E. Rank, and M. Krafczyk. Efficient reduction of 3D simulation results based on spacetree data structures for data analysis in Virtual Reality environments. In *Applied Virtual Reality in Engineering and Construction*, pages 128–135, Gothenburg, Schweden, 2001.

[56] S. Kühner and R. Romberg. Manipulation von IFC Produktmodelldaten mit "eurostep IFC Toolbox". In *Forum Bauinformatik in München*, Darmstadt, 2001.

[57] O. Klaas and M. S. Shephard. Automatic generation of octree-based three-dimensional discretizations for Partition of Unity methods. *Computational Mechanics*, 25:296–304, 2000.

[58] M. Krafczyk. *Simulation von Strömungen mit Gittergasmethoden*. Dissertation, Fakultät für Bauingenieurwesen, Universität Dortmund.

[59] M. Krafczyk. Gitter-Boltzmann-Methoden: Von der Theorie zur Anwendung. Habilitationsschrift, 2001.

[60] M. Krafczyk, M. Cerrolaza, M. Schulz, and E. Rank. Analysis of 3D transient blood flow passing through an artificial aortic valve by Lattice-Boltzmann methods. *Journal of Biomechanics*, 31:453–462, 1998.

[61] M. Krafczyk, J. Tölke, and L.-S. Luo. Large-Eddy Simulations with a Multiple-Relaxation-Time LBE Model. *International Journal of Modern Physics B*, 2002.

[62] R. Krishnan, A. Das, and B. Gurumoorthy. Octree Encoding of B-Rep Based Objects. *Computers and Graphics*, 20(1):107–114, 1996.

[63] P. Lallemand and L.-S. Luo. Theory of the lattice Boltzmann method dispersion, dissipation, isotropy, Galilean invariance, and stability. *Phys. Rev. E*, 61:6546–6562, 2000.

[64] P. Lallemand and L.-S. Luo. Lattice-boltzmann methods for moving boundaries. *Preprint*, 2001.

[65] C. F. Lange. *Numerical Prediction of Heat and Momentum Transfer from a Cylinder in Crossflow with Implications to Hot-Wire Anemometry.* Dissertation, Technische Fakultät der Universität Erlangen-Nürnberg, 1997.

[66] R. Löhner. Robust, vectorized search algorithms for interpolation on unstructured grids. *Journal of Computational Physics*, 118:380–387, 1995.

[67] R. Lietz, S. Mallick, S. Kandasamy, and H. Chen. Exterior Airflow Simulations Using a Lattice Boltzmann Approach. *Society of Automotive Engineers, Inc.*, 2002.

[68] F. Lin. Right first time design for automotive water pumps, 1999.

[69] L.-S. Luo. Theory of the lattice Boltzmann Equation. Lecture Notes for "Workshop (IV) on Soft Matters (Complex Fluids)", 2000.

[70] D. McKinney. Analysis of Subway Fire Ventilation, 1999.

[71] R. Mei, W. Shyy, D. Yu, and L.-S. Luo. Lattice Boltzmann Method for 3-D Flows with Curved Boundary. *Journal of Computational Physics*, 161:680 – 699, 2000.

[72] R. Mei, D. Yu, W. Shyy, and L.-S. Luo. Force evaluation in the lattice Boltzmann method involving curved geometry. *Physical Review E*, 65, 2002.

[73] R. Miller, G.Strumolo, E. Hytopoulos, S. Remondi, and S. Watson. High Performance Computing: Analytical Aerodynamics for Automotive Vehicles. 1999.

[74] K. Muralidhar and G. Biswas, editors. *Advanced Engineering Fluid Mechanics.* Narosa Publishing House, London, 1996.

[75] S. Nölting. The Lattice Boltzmann Method in Industrial Applications. In *18. CAD-FEM Users Meeting*, Friedrichshafen, 2000.

[76] M. N. Noui-Mehidi, N. Ohmura, and K. Kataoka. An Experimental Investigation of Flow Mode Selection in a Conical Taylor-Couette System. *International Journal of Fluid Dynamics*, 5, 2001.

[77] B. Oestereich. *Objektorientierte Softwareentwicklung: Analyse und Design.* R. Oldenburg Verlag, München, Wien, 1997.

[78] M. A. Oliver and N. E. Wiseman. Operations on Quadtree Leaves and Related Image Areas. *The Computer Journal*, 26(4), 1983.

[79] A. Pernpeintner. Aeordynamik der Bauwerke. Vorlesungsskript 1997/1998, 1998.

[80] Y. H. Qian, D. d 'Humieres, and P. Lallemand. Lattice BGK models for Navier-Stokes equation. *Europhys. Letters*, 17(6):479 – 484, 1992.

[81] E. Rank, B. Crouse, and C. van Treeck. Numerical Simulation of Air Flow for Civil Engineering Constructions on the basis of a product data model. In *The Ninth International Conference on Computing in Civil and Building Engineering*, Taipei, Taiwan, 2002.

[82] L. E. Reichl. *A modern course in statistical physics.* Univ. of Texas Press, Austin, 1980.

[83] M. B. Reider and J. D. Sterling. Accuracy of Lattice Boltzmann Methods for the Simulation of the Incompressible Navier-Stokes Equations. *Computers and Fluids*, 24(4), 1993.

[84] M. Rohde, J. J. Derksen, and H. E. A. V. den Akker. Volumetric method for calculating the flow around moving objects in lattice-Boltzmann schemes. *Physical Review E*, 65, 2002.

[85] J. Sahlmen and H. J. Niemann. Windkanalversuche als Grundlage für sturmsichere Bauwerke. *Rubin*, (2), 1997.

[86] H. Samet. Algorithms for the Conversion of Quadtrees to Rasters. *Computer Visions, Graphics, and Image Processing*, 26:1–16, 1984.

[87] H. Samet and R. E. Webber. Storing a Collection of Polygons Using Quadtrees. *ACM Transactions on Graphics*, 4(3):182–222, 1985.

[88] M. Schelkle. *Lattice-Boltzmann-Verfahren zur Simulation dreidimensionaler Zweiphasenströmungen mit freien Oberflächen.* Dissertation, Institut für Thermodynamik der Luft- und Raumfahrt, Universität Stuttgart.

[89] M. Schäfer and S. Turek. Benchmark computations of laminar flow around cylinder. *Notes on Numerical Fluid Mechanics*, 52:547–566, Vieweg Verlag, Braunschweig, 1996.

[90] H. Schlichting and K. Gersten, editors. *Grenzschicht-Theorie*. Springer Verlag, 1997.

[91] K.-J. Schneider, editor. *Bautabellen für Ingenieure*. Werner Verlag, Düsseldorf, 11 edition, 1994.

[92] M. Schulz. *Eine parallel LB-Applikation zur Durchströmung von porösen Medien*. Dissertation, Lehrstuhl für Bauinformatik, TU München, 2003.

[93] V. Seidl, S. Muzaferija, and P. M. Parallel computation of unsteady separated flows using unstructured, locally refined grids. *Notes on Numerical Fluid Mechanics*, 64:37–50, 1998.

[94] R. Shock, S. Mallick, H. Chen, V. Yakhot, and R. Zhang. Recent simulation results on 2D NACA airfoils using a lattice Boltzmann based algorithm. Exa Corporation, 2001.

[95] P. Steer and D. Haslar. Cfx, a validated design tool for marine hydordynamics, 1996.

[96] D. Stoyan, editor. *Stochastik für Ingenieure und Naturwissenschaftler*. Akademie Verlag, Berlin, 1993.

[97] J. Striegnitz. Writing Efficient Programs with C++. LRZ München, Tutorial, 2001.

[98] B. Stroustrup, editor. *Die C++-Programmiersprache*. Addison-Wesley, 3. auflage edition, 1998.

[99] S. Succi. *The Lattice Boltzmann equation for fluid dynamics and beyond*. Clarendon Press, Oxford, 2001.

[100] K. K. Tamma. Computer Graphics Aided Geometric Modeling and Mesh Generation Schemes for Preprocessing and Finite Element Analysis. *Engineering with Computers*, 3:157–167, 1988.

[101] J. F. Thompson and B. Hamann. A survey of grid generation techniques and systems with emphasis on recent developments. *Surveys on Mathematics for Industry*, 6:289–310, 1997.

[102] J. Tölke. *Gitter-Boltzmann-Verfahren zur Simulation von Zweiphasenströmungen*. Dissertation, Lehrstuhl für Bauinformatik, TU München, 2001.

[103] Z. Wang, R.F.CPhen, N.Hariharan, and A. Przekwas. A 2^N Tree Based Automated Viscous Cartesian Grid Methodology For Feature Capturing. 1999.

[104] Z. J. Wang. A Quadtree-Based Adaptive Cartesian/Quad Grid Flow Solver for Navier-Stokes Equation. *Computers and Fluids*, 27:529 – 549, 1998.

[105] H. Wengle. Numerische Berechnung turbulenter Strömungen. Vorlesungsskript 2000/2001, 2000.

[106] D. Wolf-Gladrow. *Lattice-Gas Cellular Automata and Lattice Boltzmnn Models*. Springer Verlag, Germany, 2000.

[107] J. R. Woodwark. The Explicit Quad Tree as a Structure for Computer Graphics. *The Computer Journal*, 25(2), 1982.

[108] M. A. Yerry and M. S. Shephard. Automatic Three-Dimensional Mesh Generation By The Modified-Octree Technique. *International Journal for numerical Methods in Engineering*, 20:1965–1990, 1984.

[109] M. A. Yerry and M. S. Shephard. Automatic Mesh Generation For Three-Dimensional Solids. *Computers and Structures*, 20:31–39, 1985.

[110] D. Yu, R. Mei, and W. Shyy. A multi-block Lattice-Boltzmann method for fluid flows. Denver, 2000.

[111] D. D. Zeeuw and K. G. Powell. An Adaptively Refined Cartesian Mesh Solver for the Euler Equations. *Journal of Computational Physics*, 104:56 – 68, 1992.