Rechenzentrum
Universität Stuttgart
Prof. Dr.-Ing. R. Rühle
Allmandring 30, 70550 Stuttgart

Institut für
Computeranwendungen II
Abt. Computersimulation
und Visualisierung

Anwendungen der Informatik
im Maschinenwesen

EINE FINITE-VOLUMEN-METHODE IN ALLGEMEINEN ZELLEN FÜR DIE EULER-GLEICHUNGEN MIT INTEGRIERTER SELBST-ADAPTIVER GITTERGENERIERUNG

von der Fakultät Energietechnik der Universität Stuttgart
zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr.-Ing.)
genehmigte Abhandlung

vorgelegt von

Clemens Helf
aus Erlangen

Hauptberichter:	Prof. Dr.-Ing. R. Rühle
Mitberichter:	Prof. Dr.-Ing. E. Laurien
Tag der Einreichung:	26. Januar 2000
Tag der mündlichen Prüfung:	26. Oktober 2001

ISSN 0941 - 4665 Februar 2002 RUS - 53

Zusammenfassung

Die Erzeugung von Gittern für numerische Strömungssimulationen bereitet mit zunehmender Komplexität der untersuchten Geometrien immer größere Probleme.

Um diesen Problemen zu begegnen, wird eine höchst flexible Gitterstruktur entworfen, die Polyeder (3D) bzw. Polygone (2D) mit beliebiger Anzahl von Oberflächen als Maschen benutzt. Diese Maschen müssen nicht konvex oder einfach zusammenhängend sein, so dass eine beliebige Geometrie als Masche repräsentiert werden kann, sobald ihre gekrümmten Oberflächen diskretisiert wurden. Für diese Gitterrepräsentation werden geeignete Verfeinerungs- und Vergröberungstechniken entwickelt, die ohne Eingriff des Anwenders benutzt werden können.

Zur Approximation reibungsfreier, kompressibler Strömungen wird eine zell-zentrierte Finite-Volumen-Methode mit linearer Rekonstruktion benutzt. Durch problemangepasste Rekonstruktionstechniken erhält man ein robustes numerisches Verfahren.

Aufgrund der Flexibilität der Gitterrepräsentation, der Bereitstellung automatischer Gitteradaptionstechniken und der Robustheit des numerischen Verfahrens können Berechnungen auf einer einzigen, das Rechengebiet beschreibenden Masche beginnen. Durch die vollständige Integration der Gitteradaption in den numerischen Lösungsprozess können Gitter und numerische Lösung in selbst-adaptiver Weise zusammen erzeugt werden.

Das vorgeschlagene Verfahren wurde zusammen mit allen Komponenten der Gitteradaption für den Einsatz auf Rechnern mit verteiltem Speicher parallelisiert.

Abstract

The generation of grids for numerical flow simulations represents a major problem for the application of computational fluid dynamics.

In order to tackle this problem, a highly flexible grid data structure is developed, which applies polyhedra (3D) resp. polygons (2D) bounded by an arbitrary number of surfaces as mesh cells. Mesh cells are neither restricted to be convex nor simply connected. Therefore, an arbitrary geometry may be represented as a single mesh cell, as soon as its curved surfaces are discretized. Also, appropriate refinement and coarsening techniques were developed, which do not require any user interaction.

Inviscid, compressible flows are approximated by a cell-centered Finite-Volume scheme with linear reconstruction. The numerical scheme is robust through the use of appropriate reconstruction techniques.

Due to the flexibility of the grid representation, the availability of appropriate grid adaption techniques and the robustness of the numerical scheme, calculations may be started on a single control volume, representing the computational domain. The complete integration of grid adaption into the numerical solution enables us to compute together grid and numerical solution in a self-adaptiv way.

The numerical scheme and all components of the grid generation system are implemented for use on distributed memory parallel computers.

Danksagung

Für die Betreuung der vorliegenden Arbeit möchte ich mich herzlich bei Herrn Prof. Dr.-Ing. Rühle bedanken. Die offene Atmosphäre und die hervorragende Ausstattung am Rechenzentrum haben wesentlich zum Erfolg dieser Arbeit beigetragen.

Herrn Prof. Dr.-Ing. Laurien möchte ich für den Mitbericht und die gründliche Durchsicht des Manuskripts danken, die meiner Arbeit in der Schlussphase wichtige Impulse gegeben haben.

Die Anregung zu dieser Arbeit verdanke ich Herrn Uwe Küster, der mir über die ganze Zeit mit Rat und Fachkenntnis zur Seite stand. Mein besonderer Dank gilt Herrn Dr. Klaus Birken für die gemeinsame Forschungsarbeit an der Parallelisierung des Codes, sowie Herrn Dr. Thomas Schmidt und Frau Dr. Katina Warendorf für die langjährige Zusammenarbeit.

Mein Dank gilt auch den vielen Kolleginnen und Kollegen des Rechenzentrums, die mich in vielfältiger Weise bei der täglichen Arbeit unterstützten und begleiteten. Besonders erwähnen möchte ich Manuela Wossough, Anja Heinrich, Tünde Erdei und Ramadoss Magesh.

Für die Unterstützung und den Rückhalt, den ich von meiner Frau Angelika, meinen Töchtern Raffaela und Antonia, meinen Eltern und Schwiegereltern erfahren habe, möchte ich mich bei meiner Familie besonders herzlich bedanken.

Stuttgart, im Februar 2002 Clemens Helf

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Hintergrund

Die numerische Simulation von Strömungsvorgängen ("Computational Fluid Dynamics", CFD) hat sich in einer Reihe von Ingenieurdisziplinen als ein mächtiges Analyse- und Entwurfswerkzeug erwiesen. In traditionellen Anwendungsgebieten, wie dem Entwurf von Flugkörpern, Fahrzeugen und Turbomaschinen oder der Wettervorhersage, hat sich die numerische Strömungsmechanik bereits fest etabliert, doch auch in anderen Bereichen wird sie zunehmend eingesetzt, wie etwa beim Katastrophenschutz (Simulation von Schadstoffausbreitungsmodellen, Küstenschutz), im Bauwesen (Haus- und Raumklimatisierung) oder in der Medizin (Blutströmungen).

Der erfolgreiche Aufbau numerischer Modelle für komplexe Geometrien ist auch heute noch mit so vielen Schwierigkeiten bzw. so hohem Aufwand verbunden, dass diese Techniken eher im Bereich nationaler Forschungsprojekte als im Rahmen industrieller Produktion angesiedelt sind. So ist beispielsweise die numerische Simulation einer kompletten, dreistufigen Turbine eines der als 'Grand Challenge' definierten Projekte des ASCI-Programmes ("Accelerated Strategic Computing Initiative") der amerikanischen Regierung, obwohl diese bereits seit langem im Einsatz sind. Das heißt, dass die Fähigkeit, numerische Modelle für Probleme aus der industriellen Praxis aufzubauen, hinter der technischen Entwicklung deutlich zurückbleibt.

Die Erzeugung von Gittern für diese Modelle stellt dabei ein wichtiges Teilproblem dar. Der mit der Gittergenerierung verbundene Aufwand stellt für die Anwendung numerischer Techniken ein ernstzunehmendes Problem dar, weil die Bereitstellung geeigneter Gitter für komplexe Konfigurationen gegenwärtig eine so aufwändige und komplexe Aufgabe ist, dass die Anwendung numerischer Methoden häufig einfach unterbleibt. Der Zeitaufwand für die Erzeugung eines Gitters für industriell interessante Anwendungen kann den Aufwand zur Ermittlung einer Lösung um eine oder mehr Größenordnungen übersteigen. So benötigt ein eingearbeiteter Spezialist etwa vier Wochen, um ein Gitter für ein komplettes Verkehrsflugzeug zu erzeugen, während die numerische Lösung nur mehrere Tage in Anspruch nimmt. Ähnliches kann aber auch für weniger spektakuläre Geometrien, etwa kompliziert geformte Rohrverzweigungen oder den Motorraum ("underhood") eines KFZ gelten. Selbst in Anwendungsbereichen, in denen numerische Untersuchungen regelmäßig durchgeführt werden, ist die routinemäßige Anwendung auf industriell relevante Fragestellungen noch weit von der täglichen Praxis entfernt.

1.2 Simulation und Gittergenerierung

Die numerische Simulation eines Vorgangs der realen Welt beruht auf der Abstraktion eines Systemmodells aus den relevanten physikalischen und chemischen Vorgängen. Dieses Modell wird in der Regel durch ein System von algebraischen Gleichungen und partiellen Differentialgleichungen repräsentiert, die das (lokale) Verhalten des Systems beschreiben. Vervollständigt wird das Modell durch die Definition der Systemgrenzen, z. B. in Form räumlicher Beschreibungen und geeigneter Randbedingungen.

Damit ein solches Modell auf einer digitalen Rechenanlage dargestellt werden kann, ist eine Approximation notwendig, die die Zahl der Freiheitsgrade des mathematischen Modells verringert. Die Approximation des kontinuierlichen Modells durch endlich-dimensionale Funktionenräume ist im Wesentlichen gleichbedeutend mit der Wahl eines Gitters, welches die diskreten Funktionen einbettet in den Funktionenraum des Modells. Das heißt, dass die Erzeugung von Gittern ein fester Bestandteil aller numerischen Modelle ist, auch dann, wenn die Diskretisierungstechnik als "gitterlos" bezeichnet wird. Diese Techniken benutzen nur noch einen abstrakten Nachbarschaftsbegriff (Punktwolke) und Knoten im Rechengebiet, jedoch keine Kanten mehr und werden deshalb als "gitterlos" bezeichnet.

Durch die Notwendigkeit ein Gitter bereitzustellen, beinhalten numerische Modelle einen hohen Anteil problemspezifischer Information, die aufwändig erzeugt werden muss, die Lösung in empfindlicher Weise beeinflusst und nur schlecht wiederverwendet werden kann.

Von besonderer Bedeutung ist die Abhängigkeit zwischen der Diskretisierungstechnik für die Differentialoperatoren (z. B. Finite-Differenzen-, Finite-Volumen-, Finite-Elemente-Methode) und dem Gittertyp. Jede Technik stellt spezielle Anforderungen an die Gitterstruktur. Sind diese Anforderungen unzureichend kompatibel mit der Geometrie des Problems, wird die Gittererzeugung erschwert. Die Erzeugung strukturierter Gitter, wie sie für Finite-Differenzen-Verfahren notwendig sind, ist dafür ein Beispiel.

Darüberhinaus beeinflussen die Freiheitsgrade des Gitters (z. B. die Knotenpositionen) die entstehenden diskreten Gleichungssysteme. Eine ungeeignete Wahl des Gitters kann den numerischen Lösungsprozess (z. B. die Konvergenzgeschwindigkeit) und die Qualität der Lösung beeinträchtigen oder zusammenbrechen lassen. Für die Beurteilung des Einflusses, den ein Gitter auf die Lösung hat, und für die Steuerung selbst-adaptiver Verfahren gibt es bis jetzt keine abgesicherten Qualitätsmaßstäbe, die Problemzonen zuverlässig erkennen. Die unter dem Begriff "Gitterqualität" zusammengefassten Kriterien (z. B. Gitterglattheit, Winkelbedingungen) können ebenso auf Mängel in der Diskretisierungstechnik hindeuten.

Diese Problematik führt dazu, dass die Durchführung numerischer Simulationen Vorleistungen in Form von qualifizierter Arbeitskraft, Fachwissen und Erfahrung bei der Bedienung der Software-Werkzeuge erfordert, ohne gleichzeitig einen wirtschaftlichen Nutzen zu garantieren.

1.3 Zielsetzung und Inhalt der Arbeit

Aufgrund der Komplexität und Empfindlichkeit der beschriebenen Abhängigkeit zwischen Gitter und Diskretisierung wird die herkömmliche Trennung zwischen Preprocessing und Lösung – d. h. zwischen Gittergenerierung und Lösung des diskreten Gleichungssystems – in Frage gestellt. Die vorliegende Arbeit verfolgt den Ansatz, die Schwierigkeiten bei

der Gittergenerierung durch ihre vollständige Integration in den Simulationsprozess zu "lösen".

Um diese Integration zu ermöglichen, muss die Gittergenerierung ohne Eingriff des Benutzers ablaufen. Dazu soll eine Gitterstruktur gewählt werden, die alle Schwierigkeiten umgeht, die solche Eingriffe bisher erforderten. Ausgangspunkt dieser Arbeit (**Kapitel 2**) ist deshalb eine Analyse der Rahmenbedingungen und grundlegenden Vorgehensweisen bei der Gittergenerierung. Aus der Diskussion bekannter Gitterstrukturen und der zugehörigen Gittergenerierungstechniken werden kritische Merkmale dieser Ansätze herausgearbeitet. Von besonderer Bedeutung für die Komplexität der Gittergenerierung ist die *Beschränkung auf Maschen mit fester Anzahl von Oberflächen* und damit auf einfache Maschentypen. Legt man sich auf bestimmte Maschentypen fest (z. B. Tetraeder, Hexaeder), entsteht sofort die Notwendigkeit, eine gegebene Geometrie in einem Preprocessingschritt in die festgelegte Darstellung zu transformieren. Je einschränkender die Wahl der Maschenform ist, desto aufwändiger gestaltet sich diese Vorverarbeitung, desto mehr kritische Abhängigkeiten sind bereits vorgegeben, bevor die eigentliche Simulation beginnt und desto mehr Know-How seitens des Anwenders wird benötigt. Ist die zugrundeliegende Gitterstruktur ausreichend flexibel, alle denkbaren Geometrien direkt darzustellen, entfällt das Preprocessing, weil eine gegebene Geometrie bereits eine Masche ist.

Ausgehend von den in Kapitel 2 abgeleiteten Kriterien wird in **Kapitel 3** eine höchst flexible Gitterstruktur entworfen, die aus Polyedern mit beliebiger Anzahl von Oberflächen aufgebaut ist. Als Polyeder-Oberflächen sind Polygone mit beliebiger Anzahl von Segmenten zugelassen. Die Polyeder bzw. Polygone müssen nicht konvex sein und können mehrfach zusammenhängende Oberflächen haben. Darüberhinaus werden geeignete Mechanismen zur Verfeinerung und Vergröberung solcher Maschen beschrieben. Ein neuer Algorithmus zur Teilung solcher Maschen entlang von Schnittflächen wurde entwickelt und implementiert. Auf diese Weise kann eine beliebige Geometrie nach Diskretisierung ihrer gekrümmten Oberflächen sofort als Gitter repräsentiert werden und unter Kontrolle des Simulationssystems geeignet adaptiert werden.

Diese Techniken der Gitterrepräsentation und -adaption bilden die Basis für die Integration der Gittergenerierung in den Lösungsprozess. Mit Hilfe eines einfachen Adaptionsindikators, der aus einer Zwischenlösung Bereiche des Gitters identifiziert, deren Auflösung erhöht oder verringert werden muss, entsteht in selbst-adaptiver Weise eine Strömungslösung zusammen mit einem lösungsangepassten Gitter. Aus der Sicht eines CFD-Anwenders kann die vorgeschlagene Integration als Problemlösung verstanden werden, weil die zur Simulation notwendigen Gitter nun ohne sein Zutun entstehen.

Dagegen bestehen die Probleme, die von der Gitterabhängigkeit der Diskretisierung herrühren, unabhängig von der Integration weiter, mit dem Unterschied, dass man sich nicht mehr mit dem Hinweis auf einschlägige Software-Werkzeuge oder die Erfahrung und Intuition des Anwenders zufrieden geben kann. Ein Schwerpunkt dieser Arbeit liegt deshalb darin, die übrigen Komponenten des Simulationssystems so auszulegen, dass sie trotz der Flexibilität dieser Gitterdarstellung zuverlässig und genau arbeiten. **Kapitel 4** beschreibt die verwendete zell-zentrierte Finite-Volumen-Methode mit linearer Rekonstruktion, für die neue, problemangepasste Rekonstruktionstechniken entwickelt wurden.

Um mit dem entwickelten Verfahren auch größere Probleme (z. B. Umströmung eines kompletten Flugzeugs) bearbeiten zu können, ist der Einsatz moderner Höchstleistungsrechner (Vektor- und Parallelrechner) notwendig, die hohe Rechenleistung und große Spei-

cher zur Verfügung stellen. In **Kapitel 5** wird deshalb die Implementierung grundlegender Programm- und Datenstrukturen diskutiert, wie sie von selbst-adaptiven numerischen Verfahren auf unstrukturierten Gittern benötigt werden, um den Anforderungen an die dynamische Änderbarkeit der Datenstrukturen und die geometrische Flexibilität gerecht zu werden. Darüberhinaus wird dort die Parallelisierung des vorgestellten Verfahrens für Rechnerarchitekturen mit verteiltem Speicher beschrieben. Sie beruht auf dem in [9] entwickelten Konzept eines verteilten Graphen, welches sich durch die effiziente Unterstützung adaptiver, verteilter Datenstrukturen auszeichnet.

Kapitel 6 behandelt die selbst-adaptive Lösungsstrategie, die von den Anfangsbedingungen auf dem einzelnen, die Geometrie beschreibenden Kontrollvolumen zu einer genauen, numerischen Lösung auf einem lösungsangepassten, feinen Gitter führt. Verschiedene Testfälle für die reibungsfreie Strömung idealer Gase in 2D belegen die Anwendbarkeit des vorgestellten Ansatzes. Erste Ansätze zur Bearbeitung von 3D-Gebieten werden gezeigt.

Die vorgelegte Arbeit schließt mit einer Zusamenfassung und einem Ausblick.

Kapitel 2

Darstellung von Rechengebieten für numerische Simulationen

Bevor ein numerisches Simulationsverfahren eingesetzt werden kann, müssen diskrete Anfangs- und Randbedingungen sowie eine geeignete, räumliche Diskretisierung der zu untersuchenden Geometrie bereitgestellt werden. Die dazu notwendigen Arbeitsschritte werden unter dem Begriff *Preprocessing* zusammengefasst.

Die Diskretisierung des Untersuchungsgebiets bildet die Grundlage für die Diskretisierung der Differentialgleichungen und zerlegt das Untersuchungsgebiet in der Regel in *Maschen*, deren Knoten und Kanten das *Gitter* bilden. Deshalb wird dieser Vorgang als *Gittergenerierung* bezeichnet. Das von den Maschen überdeckte Gebiet (*Rechengebiet*) beschreibt das Untersuchungsgebiet näherungsweise. Das Untersuchungsgebiet wird im Wesentlichen durch seine Oberflächen, die *Geometrie*, charakterisiert. Die am meisten verbreiteten *Maschentypen* in 2D sind Dreiecke und Vierecke und in 3D Hexaeder sowie Tetraeder, Prismen, und Pyramiden.

Aufgrund von Einschränkungen, die von der verwendeten Diskretisierungstechnik, den zur Verfügung stehenden Gittergenerierungswerkzeugen und der Komplexität der Implementierung herrühren können, wird zur Darstellung des Rechengebietes vorzugsweise nur ein Maschentyp (oder wenige Maschentypen) verwendet. Darüberhinaus können sich die zur Implementierung eingesetzten Datenstrukturen (*Gitterstrukturen*) für verschiedene Maschentypen wesentlich voneinander unterscheiden. Deshalb unterscheidet man zwischen verschiedenen *Gittertypen* nach dem Typ der verwendeten Maschen.

Dieses Kapitel untersucht den Einfluss des Maschentyps und der zugehörigen Gitterstruktur auf die Gittererzeugung. Dazu werden die Rahmenbedingungen (Abschnitt 2.1) und die Struktur (Abschnitt 2.2) der Gittergenerierung analysiert. Anhand eines Überblicks über bekannte, grundlegende Gittertypen (Abschnitt 2.3) und Kopplungstechniken für Gitterblöcke (Abschnitt 2.4) werden die Eigenschaften verschiedener Gittertypen im Hinblick auf die Darstellung komplexer Geometrien betrachtet. In Abschnitt 2.5 werden kritische Eigenschaften dieser Strukturen diskutiert und daraus die Grundzüge der in dieser Arbeit entwickelten Gitterrepräsentation abgeleitet, die in Kapitel 3 spezifiziert wird. Abschnitt 2.6 beschreibt abschließend die Struktur eines adaptiven Verfahrens.

2.1 Aufbereitung von Geometriedaten

Am Ausgangspunkt der Gittergenerierung steht die Aufbereitung einer in beliebiger Weise beschriebenen, eventuell unvollständigen Geometrie. Damit eine solche Geometrie mit Hilfe von Software-Werkzeugen bearbeitet werden kann, wird sie in eine *einheitliche* Darstellung gebracht. Dann können Fehler oder Unvollständigkeiten korrigiert werden, so dass eine *konsistente* Beschreibung entsteht, die bestimmten minimalen Anforderungen genügt. Unabhängig von der zu erzeugenden Gitterstruktur und den zur Gittergenerierung verwendeten Techniken bildet eine einheitliche, konsistente Beschreibung der Geometrie den Ausgangspunkt für die software-gestützte Erzeugung eines Initialgitters.

Das Untersuchungsgebiet wird durch seine Oberflächen und die Auszeichnung des "Inneren" beschrieben. Äußere Oberflächen grenzen dabei die Untersuchungsmenge ("Inneres") von der Menge der nicht betrachteten Punkte des R^d ("Äußeres") ab, innere Oberflächen repräsentieren beispielsweise Materialgrenzen innerhalb der Untersuchungsmenge. Dabei bezeichnet d die Raumdimension. In technisch relevanten Geometrien werden diese Ränder durch eine endliche Zahl stückweise glatter Kurven bzw. Flächen ("$d-1$-dimensionale Mannigfaltigkeiten") dargestellt.

2.1.1 Einheitliche Geometriebeschreibung

Die zu diskretisierenden Geometrien stammen aus einer Vielzahl verschiedener Quellen. Geometrien können beispielsweise durch analytische Funktionen, als Abtastwerte realer Körper, in Form von "Blaupausen" auf Papier oder in Form computergestützter Geometriemodelle vorliegen. Darüberhinaus existieren diverse Geometrie-Austauschformate, die als Schnittstelle zwischen CAD-Systemen und Werkzeugen der Gittergenerierung dienen, Darstellungen von Zwischenstufen aus solchen Werkzeugen und die "fertigen" Gitter selbst.

Soll der Prozess der Gittergenerierung durch Software-Werkzeuge unterstützt werden, so zieht man sich auf eine Teilmenge von Repräsentationen zurück, die einer einheitlichen Behandlung zugänglich sind. Durch diese Einschränkung wird eine Umwandlung der gegebenen Geometrie in die ausgewählte Repräsentation notwendig, die um so aufwändiger ist, je kleiner die Menge der unterstützten Darstellungen ist. Die Abbildung in die unterstützte Darstellung ist mit einem Approximationsfehler verbunden. Weitere Einschränkungen für die Wahl der Elemente der Beschreibung können von den Erfordernissen der Gittergenerierung oder dem numerischen Verfahren ausgehen.

Ein möglicher Ansatz ist die Darstellung der Geometrie durch NURBS ("non-uniform, rational B-splines"). NURBS sind parametrisierte Kurvendarstellungen, die durch Bildung kartesischer Produkte zur Darstellung von Flächen und Volumen eingesetzt werden können. Es sind eine Reihe günstiger Eigenschaften (z. B. lokale Kontrolle der Kurvenform, Verlauf der Kurve in der konvexen Hülle des Kontrollpolygons) und verschiedene Manipulationstechniken (z. B. Einfügen von Knoten, Aufteilen der Kurve) bekannt, die NURBS im Bereich des computergestützten Geometrieentwurfs ("computer aided geometry design", CAGD) populär gemacht haben. Darüberhinaus wurden Techniken zur Umwandlung gängiger CAD-Objekte in diese Darstellung entwickelt, z. B. [68].

2.1.2 Konsistente Geometriebeschreibung

Die Oberflächenbeschreibung ist genau dann konsistent, wenn die dargestellte Oberfläche zweiseitig und geschlossen ist, d. h. als Fläche keinen Rand hat. Diese Forderung läßt sich anschaulich als Wasserdichtigkeit ("water-tight surface") der Oberfläche beschreiben. Darüberhinaus kann das Innere des eingeschlossenen Gebietes als zusammenhängend angenommen werden, weil die Aufgabenstellung andernfalls in unabhängige Teilprobleme zerfällt.

Zweiseitige Oberflächen bilden die notwendige Voraussetzung, um das Innere vom Äußeren der Geometrie unterscheiden zu können. So sind beispielsweise Möbiusbänder einseitige Flächen des R^3 und zur Berandung einer 3D-Geometrie ungeeignet. Der Randbegriff der Oberfläche leitet sich mit Hilfe einer Homomorphismus zwischen der Oberfläche und einer Parametermenge ($\subset R^{d-1}$) aus der Topologie des R^{d-1} ab. Ein Homomorphismus ist eine bijektive Abbildung, die zusammen mit ihrer Umkehrabbildung stetig ist, d. h. sie induziert eine Topologie auf der Oberfläche und damit einen Randbegriff.

Geometriebeschreibungen, die in diesem Sinne *inkonsistent* sind, treten häufig auf. So ist es beispielsweise üblich, Tragflügelprofile durch eine Fläche bzw. Kurve zu beschreiben, welche an der Hinterkante nicht geschlossen ist. Weil die Art und Weise in der die Fläche geschlossen wird Einfluss auf die Ergebnisse reibungsfreier Strömungsberechnungen hat, wird für Referenzfälle eine Schließungsvorschrift angegeben [3].

Eine weitere Quelle inkonsistenter Geometriebeschreibungen bilden fehlerhafte Umwandlungsprogramme [30].

Eine andere Schwierigkeit tritt bei komplexen, technischen Konfigurationen auf, die aus verschiedenen Komponenten zusammengesetzt sind. Als Beispiel kann eine Konfiguration aus einem Tragflügel mit Triebwerksaufhängung und -gondel dienen. Diese Komponenten werden getrennt entworfen und produziert, so dass ihre Beschreibungen ebenfalls getrennt vorliegen und zuerst geeignet verschnitten werden müssen, um eine konsistente Beschreibung zu erhalten. In [54] wird eine Technik zur Verschneidung von Oberflächentriangulierungen beschrieben. Dieses Problem tritt noch verstärkt bei Konfigurationen auf, bei denen die Komponenten nicht starr verbunden sind und Relativbewegungen eine Rolle spielen, etwa bei der Einfahrt eines Hochgeschwindigkeitszuges in einen Tunnel.

2.1.3 Geometriedaten aus CAD-Systemen

Die Aufbereitung von Geometriebeschreibungen aus CAD-Systemen stellt ein wichtige Aufgabenstellung für die Gittergenerierung dar, weil technische Konfigurationen aufgrund ihrer Komplexität häufig mit Computerunterstützung entworfen werden.

Da CAD-Systeme in erster Linie mit dem Ziel entwickelt wurden, optische Darstellungen (Baupläne) zu liefern, beschreiben gängige CAD-Systeme Geometrien durch ein Flächenmodell. Dabei werden die Berührungen verschiedener Flächen nicht explizit repräsentiert, sondern Lücken und Schnitte unterhalb der Wahrnehmungsgrenze zugelassen. Außerdem gibt es keine Unterscheidung zwischen dem Inneren und dem Äußeren der Geometrie. Auch die als Schnittstelle benutzten Datenaustauschformate (z. B. STL, VDA-FS, IGES) ignorieren dieses Problem und erhöhen teilweise noch die Redundanz in den Daten. So werden beispielsweise Flächen in der STL-Beschreibung durch eine Triangulierung repräsentiert,

die ihre Knoten nicht durch eindeutige Indizes, sondern durch ihre Koordinaten referenziert. Da bei der Erzeugung der Koordinaten rundungsfehlerbehaftete Operationen eine Rolle spielen, können "gleiche" Knoten geringfügig verschiedene Koordinaten erhalten.

Enthält eine Geometrie verschiedene Längenskalen, können solche Ungenauigkeiten in der Flächendarstellung die Identifikation einer das Rechengebiet berandenden, geschlossenen Oberfläche zu einer Aufgabe machen, die nicht ohne graphische Werkzeuge und Interaktion mit dem Benutzer zu lösen ist. Letztendlich bedeutet dies, dass die Flächenbeschreibung vom Anwender "interpretiert" werden muss, um intentionale Lücken von Ungenauigkeiten zu unterscheiden. Aufgrund dieser Probleme implementieren neuere CAD-Systeme Volumenmodelle, in denen die Schnitte verschiedener Flächenstücke explizit repräsentiert sind und ein ausgezeichnetes Inneres existiert. Moderne Schnittstellen wie die STEP Geometrieschnittstelle machen diese Information auch nach außen sichtbar und bieten eine geeignete Schnittstelle für die Gittergenerierung.

2.2 Gittererzeugung

Im Prozess der Gittererzeugung lassen sich häufig die folgenden Teilaufgaben identifizieren: die Zerlegung des Rechengebietes in *Blöcke*, innerhalb derer nur ein bestimmter Maschentyp erzeugt wird, für jeden Block die Erzeugung eines *Oberflächengitters* als Randbedingung der darauf folgenden Erzeugung des *Raumgitters* sowie die Behandlung der Blockgrenzen. Diese Schritte werden in der Regel in einen Optimierungsprozess eingebettet, der die Qualität der erzeugten Gitter verbessert.

Aufgrund ihrer Unterschiedlichkeit müssen für jeden einzelnen Gittertyp geeignete Gittergenerierungstechniken identifiziert und implementiert werden. Für Verfahren, die lokale Bereiche niedriger Qualität nicht ausschließen, wurden interaktive, graphische Werkzeuge entwickelt, um das erzeugte Gitter zu kontrollieren und zu modifizieren. Mehrere Werkzeuge diesen Typs werden z. B. in [63] vorgestellt.

2.2.1 Hintergrundgitter

Um aus der Geometriebeschreibung die Oberflächen- und Raumgitter des gewünschten Typs abzuleiten, müssen geometrische und/oder topologische Fragestellungen behandelt werden, z. B. die Bestimmung von Schnitten und Abständen, die Darstellung noch nicht bearbeiteter Bereiche der Geometrie, die Erkennung degenerierter Maschen, usw. Zur Erzeugung strukturierter Gitter werden Gleichungssysteme für die Koordinaten auf der Geometriebeschreibung oder Teilen davon gelöst. Häufig treten Suchprobleme in den Algorithmen auf, die sich ohne zusätzliche Maßnahmen nicht mit vertretbarem Aufwand durchführen lassen. Diese Fragestellungen sind mittels der Geometriebeschreibung gar nicht oder nur schlecht zu bearbeiten. Zudem treten sie zu einem Zeitpunkt auf, zu dem das zu erzeugende Gitter noch nicht oder nicht vollständig zur Verfügung steht, also zur Bearbeitung dieser Aufgaben nicht eingesetzt werden kann.

Gibt es keine systematischen Lösungsansätze für solche Probleme, dann bleiben sie dem Anwender überlassen. So verlangen beispielsweise Gittergeneratoren für strukturierte Gitter eine manuelle Parkettierung des Rechengebietes, weil diese topologische Information

über das durch die Geometriebeschreibung eingeschlossene Gebiet nicht eindeutig abgeleitet werden kann. Aufbauend auf der Parkettierung und der Geometriebeschreibung kann dann ein wesentlicher Teil der Gittergenerierung automatisiert werden.

Würde stattdessen die Komplexität der Geometriebeschreibung so weit reduziert, dass sich die gewünschten Aufgaben bearbeiten lassen, so könnten nur noch triviale Geometrien dargestellt werden, da die meisten Gittergenerierungstechniken für einfache Maschentypen (z. B. Tetraeder, Prismen, Quader) entwickelt wurden und nur für diese anwendbar sind. Andere Schwierigkeiten sind mit gekrümmten Kurven bzw. Flächen verknüpft, auf deren Repräsentation nicht verzichtet werden kann.

Ein wesentlich tragfähigerer Ansatz ist die Verwendung sogenannter "Hintergrundgitter", die die gegebene Geometrie unter dem Gesichtspunkt der Gittergenerierung geeignet repräsentieren. Interessanterweise werden dafür ausschließlich dynamische, adaptive Strukturen eingesetzt, die sich ohne Benutzerintervention aus der Geometriebeschreibung generieren lassen (z. B. Quad- bzw. Octrees, verkettete Listen, etc.). Aufgrund der Verschiedenartigkeit der Fragestellungen können unterschiedliche Hintergrundgitter zum Einsatz kommen, etwa eines für die Umwandlung der gegebenen Geometrie in die einheitliche, konsistente Beschreibung, eines für die Erzeugung des Oberflächengitters, eines für die Erzeugung des Raumgitters und eines zur Bearbeitung von Blockgrenzen.

2.2.2 Zerlegung in Teilgebiete

Durch die Zerlegung des Rechengebietes in Teilgebiete entsteht ein Gitter, dessen Maschen (Makromaschen) die Teilgebiete sind. Die Aufteilung wird dabei so gewählt, dass sich die Erzeugung der Gitter für die Teilgebiete vereinfacht. Dieser Ansatz erlaubt es beispielsweise, Rechengebiete, für die ein strukturiertes Gitter nicht einfach erzeugt werden kann (z. B. mehrteilige Tragflügelprofile) in Teilgebiete zu zerlegen, für die das einfacher möglich ist.

Durch Verwendung verschiedener Gittertypen in jedem Teilgebiet können die Vorteile (und Nachteile) verschiedener Gittertypen für die Gittergenerierung miteinander verknüpft werden (z. B. Hybridgitter). Da die für ein Teilgebiet erzeugten Gitterblöcke organisatorische Einheiten des Programmes bilden, erlaubt eine Zerlegung in Teilgebiete Einschränkungen zu umgehen, die für einen einzelnen Gitterblock zwingend sind. Diese Technik erlaubt eine grobgranulare Organisation von Adaptivität [45] oder die Partitionierung des Gebietes für datenparallele Algorithmen, wenn die Struktur eines einzelnen Gitterblocks dafür nicht die notwendigen Voraussetzungen bereitstellt.

Eine wichtige Frage ist, in welcher Weise die Nachbarschaftsbeziehungen der Makromaschen repräsentiert werden.

Bilden die Makromaschen selbst ein Gitter mit strukturierten Nachbarschaftsbeziehungen, so kann durch kompatible Wahl der Teilgitter in den Makromaschen – durch übereinstimmende Knoten an den Blockgrenzen – ein strukturiertes Gitter für das gesamte Rechengebiet erzeugt werden, d. h. die Erzeugung eines strukturierten Gitters aus Makromaschen ist im Wesentlichen gleichbedeutend mit der Erzeugung eines strukturierten Gitters für das Gesamtgebiet.

Geht man von der Annahme aus, dass die Erzeugung eines solchen strukturierten Gitters für das gesamte Rechengebiet nicht möglich, zu aufwändig oder aus anderen Gründen nicht

wünschenswert ist, so müssen zur Repräsentation der Nachbarschaftsbeziehungen der Makromaschen unstrukturierte Techniken eingesetzt werden, die durch die explizite Beschreibung der Nachbarschaftsbeziehungen charakterisiert sind. Dadurch steigt die Komplexität der Implementierung und erhöht die Bedeutung moderner Ansätze des Software-Entwurfs für numerische Anwendungen, vgl. Kapitel 5.

2.2.3 Erzeugung von Oberflächengittern

Für die Erzeugung eines Raumgitters stellt die Geometriebeschreibung eine Randbedingung dar: die Oberfläche des zu erzeugenden Raumgitters soll eine geeignete Diskretisierung der Geometriebeschreibung darstellen. Daher gehen die Algorithmen zur Erzeugung von Raumgittern von einer Diskretisierung der Oberflächen aus. Besonders auffällig ist dies bei "Advancing-Front"-Methoden. Das heißt, dass aus der Geometriebeschreibung ein Oberflächengitter hergestellt werden muss, das zur Technik der Raumdiskretisierung kompatibel ist.

Dabei beeinflusst das Oberflächengitter die Eigenschaften des entstehenden Raumgitters und kann seine Erzeugung verhindern. So existiert z. B. in 3D keine "constrained-Delaunay-Tetrahedrierung", d. h. eine Oberflächentriangulierung und Raumpunkte können so gegeben sein, dass sich die Delaunay-Tetrahedrierung der Raumpunkte nicht so modifizieren lässt, dass sie die gegebene Oberfläche enthält [38]. In diesem Fall muss das Oberflächengitter im Verlauf der Erzeugung des Raumgitters, z. B. durch Einfügen, Entfernen oder Verschieben von Knoten, verändert werden. Dazu muss eine vom Gittertyp und dem Oberflächengitter unabhängige, Repräsentation der Oberfläche zusätzlich bereitgestellt werden.

Die dadurch entstehende, wechselseitige Abhängigkeit zwischen Raumgitter, Oberflächengitter und Geometriebeschreibung kompliziert den Prozess der Gittergenerierung.

2.2.4 Erzeugung von Raumgittern

Für die Erzeugung von Raumgittern ist die Auswahl zulässiger Maschentypen der erste entscheidende Schritt. Diese Auswahl bestimmt im Wesentlichen, welche Schwierigkeiten mit der Repräsentation komplexer Geometrien verbunden sind und in welchem Umfang der Anwender mit ihnen belastet wird. Die Raumgittererzeugung hat zwei Aufgaben zu bewältigen: die Bereitstellung einer Menge von Gitterknoten sowie den Aufbau von zulässigen Maschen aus diesen Knoten. Dabei soll das entstehende Gitter bestimmten maschentyp-spezifischen Qualitätskriterien genügen.

Ist das Untersuchungsgebiet einer der gewählten Maschenformen ähnlich, kann das Gebiet also einfach auf eine Masche abgebildet werden, so vereinfacht sich der Vorgang der Gittergenerierung wesentlich, weil das Gebiet direkt in der gewünschten Gitterstruktur repräsentiert werden kann. Die Erzeugung des Raumgitters kann dann durch die der Struktur angepasste Verfeinerungstechnik bewirkt werden, die die Raumknoten und deren Konnektivität in natürlicher Weise bereitstellt. In diesem Fall sind außer der Gitterstruktur selbst keine weiteren Darstellungen der Geometrie zu implementieren. Dies ist zum Beispiel der Fall, wenn das Untersuchungsgebiet (2D) selbst dreieckig oder viereckig ist. Ein solches Gebiet kann in der Implementierung direkt als Masche repräsentiert werden. Die Masche kann dann beispielsweise durch wiederholtes halbieren der Kanten verfeinert werden.

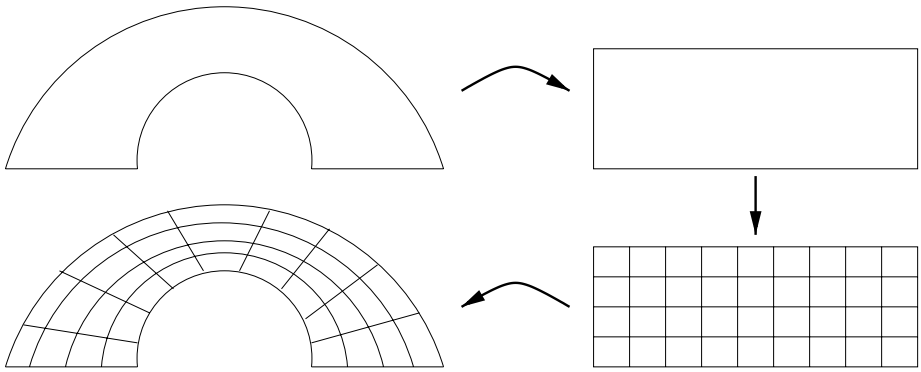


Abbildung 2.1: Gittergenerierung durch Ähnlichkeitsabbildung

Für die Erzeugung strukturierter Gitter besteht die wesentliche Schwierigkeit in der Ermittlung dieser Abbildung, welche beispielsweise durch algebraische Ansätze oder durch Lösung von partiellen Differentialgleichungen für die Koordinaten der Knoten bereitgestellt werden kann (vgl. Abschnitt 2.3.2, vgl. [63]). Ist eine solche Abbildung nicht verfügbar, so muss das betrachtete Gebiet in unstrukturierter Weise solange in einfachere Teile (Blöcke oder Maschen) zerlegt werden, bis eine solche Abbildung gefunden werden kann.

Bei unstrukturierten Gittern kann diese Zerlegung durch Abspalten zulässiger Maschen vom betrachteten Gebiet realisiert werden ("advancing front"). Dazu wird die Grenze ("front") zwischen dem bereits bearbeiteten Teil des Rechengebietes und dem noch unbearbeiteten Teil explizit repräsentiert, die anfänglich aus der Oberfläche des Rechengebietes besteht. Neue Gitterknoten werden dabei mit Hilfe einer lokalen Normalenrichtung der Gitterfront in den noch unbearbeiteten Teil des Rechengebietes eingefügt, bis das gesamte Rechengebiet bearbeitet ist. Besondere Sorgfalt erfordert dieser Ansatz bei der Behandlung von Geometrien mit ein- oder ausspringenden Ecken [40] sowie bei der Implementierung der notwendigen Suchvorgänge.

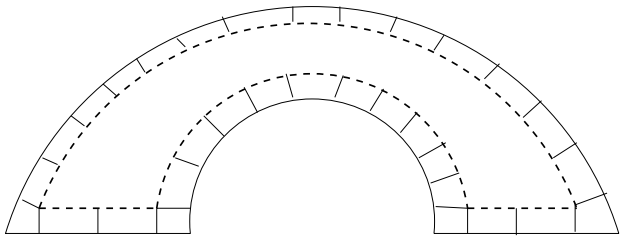


Abbildung 2.2: Gittergenerierung durch Abspalten von Maschen (Front gestrichelt)

Eine andere Möglichkeit zur Erzeugung eines unstrukturierten Raumgitters ist die lokale Verfeinerung eines die Geometrie überdeckenden, trivialen Gitters. Dazu werden einige Maschen bestimmt, die die gegebene Geometrie überdecken. Dieses Gitter wird solange verfeinert, bis die Geometrieoberflächen geeignet repräsentiert sind. Danach können die Bereiche des Raumgitters außerhalb der Geometrie entfernt werden und die von der Geometrieoberfläche entfernten Bereiche des Raumgitters innerhalb der Geometrie so verfeinert werden, dass die Maschen auch dort den gewünschten Qualitätskriterien [38, 16]. Für die Berechnung reibungsbehafteter Strömungen wurde vorgeschlagen, kartesische Gitter mit einem prismatischen Oberflächengitter zu koppeln, [15].

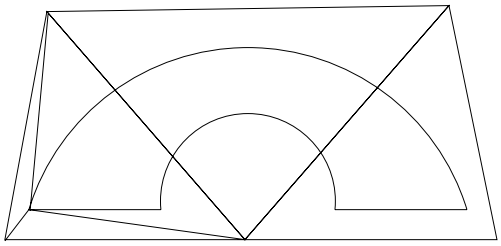


Abbildung 2.3: Gittergenerierung durch Verfeinerung einer Überdeckung (nach Einfügen eines Randknotens

2.2.5 Verknüpfung der Gitterblöcke

Können die aneinandergrenzenden Maschenoberflächen benachbarter Gitterblöcke kompatibel gewählt werden, so bleibt die Diskretisierung von der Existenz der Blockgrenze unbeeinflusst. Diese wünschenswerte Eigenschaft ist für den Übergang zwischen strukturierten und unstrukturierten Blöcken aus Dreiecken und Vierecken bzw. Tetraedern und Prismen gegeben. Für Übergänge zwischen Tetraedern und Hexaedern sind dagegen besondere Vorkehrungen erforderlich, weil Tetraeder von Dreiecken, Hexaeder aber von Vierecken begrenzt werden. Durch Erzeugung einer vermittelnden Pufferzone aus Pyramiden können die Oberflächen von Hexaedern und Tetraedern aneinander angepasst werden [53].

Für die Implementierung des Blockübergangs werden häufig sogenannte Geister-, Halo- oder Dummy-Zellen herangezogen. Für jede äußere Oberfläche des Gitterblocks wird in der Darstellungstechnik dieses Blocks eine zweite Masche erzeugt, die als Platzhalter für die angrenzende Masche des benachbarten Gitterblocks dient. Die Daten dieser Platzhalter werden aktualisiert, bevor sie benutzt werden. Diese Aktualisierung wird über Nachbarschaftslisten kontrolliert, welche die unstrukturierte Nachbarschaft an der Blockgrenze repräsentieren. Aufgrund des durch diese Überlappung induzierten Aufwandes sollten die Blöcke nicht zu klein gewählt werden.

Der Implementierung einer Blockgrenze durch Überlappung kann man sich nur entziehen, wenn keiner der angrenzenden Blöcke strukturiert ist und Zeiger zur Darstellung unstrukturierter Nachbarschaftsbeziehungen benutzt werden. Unter diesen Voraussetzungen kann auf die Aufteilung in Blöcke verzichtet werden.

Ist eine Anpassung der Oberflächen aneinandergrenzender Blöcke nicht möglich, etwa bei aneinandergrenzenden strukturierten Gitterblöcken, so müssen im numerischen Verfahren Interpolationstechniken vorgesehen werden, um die Blockgrenze zu behandeln. Da der Einfluss der Interpolation auf die Genauigkeit der Lösung nicht zuverlässig quantifizierbar ist, ist die Intuition und Erfahrung des Anwenders bei der Wahl der Blockgrenzen gefragt.

2.3 Bekannte Gittertypen

Komplexe Geometrien werden häufig nicht direkt aus Maschen aufgebaut, sondern Maschen werden in Teilgittern organisiert (“Mikroblöcke”, “Makroblöcke”, “Blöcke”, “Partitionen”), die dann zur Gesamtgeometrie zusammengefügt werden. Diese hierarchische Organisation erlaubt Einschränkungen zu umgehen, die für die Teilgitter zwingend sind.

Oder Teilgitter dienen als organisatorische Einheiten für die Gittergenerierung oder die Datenverteilung auf parallelen Rechnerarchitekturen.

(Teil-)Gitter werden in der Regel aus einer einzigen oder wenigen verschiedenen, grundlegenden Maschentypen aufgebaut, die sich einheitlich implementieren lassen. So kommen beispielsweise Dreiecke und Vierecke in 2D sowie Tetraeder, Prismen, Pyramiden und Hexaeder in 3D zum Einsatz. Durch die Beschränkung auf wenige Maschentypen wird die Implementierung der Datenstrukturen vereinfacht und die Möglichkeit geschaffen, Regelmäßigkeiten in den Datenstrukturen zur Optimierung der Rechenleistung zu nutzen.

Die hier getroffene Unterscheidung zwischen *grundlegenden Gittertypen* und *gekoppelten Gittertypen* orientiert sich an der Frage, in welcher Weise Daten aus benachbarten Maschen in das Diskretisierungsschema eingehen. Kommunizieren alle Maschen eines (Teil-)Gitters durch direkten Zugriff, so liegt nach der hier getroffenen Unterscheidung ein grundlegender Gittertyp vor. Ein gekoppelter Gittertyp ist dadurch ausgezeichnet, dass im Gitter Blockgrenzen existieren, an denen die Diskretisierung durch besondere Maßnahmen aufrecht erhalten wird oder wechselt. Der Datenaustausch zwischen Maschen, die über eine Blockgrenze hinweg benachbart sind, wird häufig durch zusätzliche Kopien oder durch Interpolationstechniken bewerkstelligt.

Auch andere Unterscheidungsmerkmale können zur Klassifikation herangezogen werden. Eine Unterscheidung nach dem Maschentyp bietet sich beispielsweise an, wenn maschentyp-spezifische Gittergenerierungstechniken im Mittelpunkt stehen. Unterscheidet man nach Maschentyp, so stellt sich ein unstrukturiertes Gitter mit mehreren Maschentypen (z. B. Tetraeder und Prismen) als gekoppelter Gittertyp dar. Nach der oben vorgeschlagenen Klassifikation liegt nur dann ein gekoppeltes Gitter vor, wenn eine uneinheitliche Implementierung der Maschen vorliegt, die die Implementierung einer Blockgrenze erforderlich macht.

2.3.1 Übersicht: Grundlegende Gittertypen

Folgende Gittertypen sollen hier kurz vorgestellt werden:

- Strukturierte Gitter (Vierecke/Hexaeder)
- Kartesische Gitter (Rechtecke/Quader)
- Unstrukturierte Gitter aus Dreiecken/Tetraedern oder Vierecken/Hexaedern (semistrukturierte Gitter)
- Unstrukturierte Gitter mit mehreren Maschentypen ("Hybridgitter")

Strukturierte Gitter (SG)

SG sind der älteste und einfachste Gittertyp, der im Bereich der numerischen Simulation partieller Differentialgleichungen angewendet wird. Sie besitzen eine Reihe günstiger numerischer Eigenschaften und werden deshalb noch immer bevorzugt eingesetzt. SG beruhen auf einer (Koordinaten-) Transformation des Rechengebietes auf einen regelmäßig unterteilten Einheitswürfel. Die Kreuzungspunkte der Koordinatenlinien werden als Knoten des Gitters aufgefasst. In realistischen Anwendungsfällen wird eine nicht-triviale Transformation benutzt, man spricht dann von strukturierten Gittern mit krummlinigen bzw. oberflächenangepassten Koordinaten. SG werden in Multiblock- und Hybridgittern als Teilgitter eingesetzt.

Kartesische Gitter

Kartesische Gitter werden aus rechteckigen bzw. quaderförmigen Maschen fester Größe aufgebaut. Die zu untersuchende Geometrie wird in das Gitter "eingeschnitten", so dass an der Geometrieoberfläche Maschen entstehen, die nicht vollständig im Rechengebiet enthalten sind. Diese Maschen bedürfen einer besonderen Behandlung im Hinblick auf die Diskretisierung der Oberfläche, die Art des Schnittes zwischen Oberfläche und Masche, bei der Implementierung des numerischen Verfahrens und der Randbedingungen.

Kartesische Gitter können im Hinblick auf den Maschentyp als Spezialfall strukturierter Gitter aufgefasst werden. Ein für die Anwendung, Implementierung und Rechenleistung bedeutsamer Unterschied ist jedoch, dass ein strukturiertes Gitter entlang jeder Koordinatenlinie gleichviele Knoten besitzt, während im kartesischen Gitter die Knotenabstände festgelegt sind. Wird der Knotenabstand von der notwendigen Auflösung an der Oberfläche diktiert, so wird eine für viele Anwendungen unrealistisch hohe Auflösung des gesamten Rechengebietes erforderlich. Daher wird dieser Gittertyp zusammen mit Techniken der lokalen Gitteradaption verwendet, die eine effiziente Auflösung der Oberflächen des Rechengebietes ermöglichen [45, 69].

Unstrukturierte Gitter

Unstrukturierte Gitter zeichnen sich durch die explizite Repräsentation der Nachbarschaftsbeziehungen zwischen den Maschen des Gitters aus, d. h. es gibt keine globale Sicht (Struktur) dieser Beziehungen. Im Gegensatz dazu sind die Nachbarschaftsbeziehungen bei strukturierten Gittern implizit durch die Organisation der Maschen in Koordinatenlinien und bei kartesischen Gittern durch hierarchische Datenstrukturen repräsentiert.

Ursprünglich wurden unstrukturierte Gitter ausschließlich aus dreieckigen bzw. tetraederförmigen Maschen aufgebaut. Die Oberfläche der zu untersuchenden Geometrie wird dazu durch Kanten bzw. Dreiecke diskretisiert, aus denen dann ein Gitter im Inneren des Gebietes abgeleitet wird. Eine Kombination zwischen strukturierten und unstrukturierten Gittertypen stellen die "semi-strukturierten" Gitter dar. Sie verbinden die Maschentypen strukturierter Gitter (Vierecke bzw. Hexaeder) mit der expliziten Darstellung der Nachbarschaftsbeziehungen unstrukturierter Gitter. Die Kombination von Tetraedern und Prismen in einem unstrukturierten Gitter wird mittlerweile als "Hybridgitter" bezeichnet [21]. Ursprünglich wurde mit diesem Begriff die Kopplung von strukturierten Oberflächengittern mit unstrukturierten Gittern im Fernfeld bezeichnet.

Unter dem Gesichtspunkt der verwendeten Datenstruktur ist die Verwandtschaft zu strukturierten Gittern bzw. Hybridgittern nur vordergründig durch die Verwendung von viereckigen oder hexaederförmigen Maschen gegeben. Die Eigenschaften einer solchen Gitterstruktur werden jedoch durch die unstrukturierte Repräsentation der Nachbarschaftsbeziehungen dominiert.

2.3.2 Strukturierte Gitter

Aufgrund ihrer Regelmäßigkeit können SG als mehrdimensionale Felder implementiert werden, die Nachbarschaftsbeziehungen werden implizit durch Indexrechnungen ausgedrückt und müssen deshalb nicht gespeichert werden. Größe, Schwerpunkt und Integrationspunkte können elementargeometrisch ermittelt werden. Dadurch beschränkt sich der

Aufwand zur Repräsentation der Geometrie auf die Felddimensionen und die Darstellung der Transformation durch die Koordinaten der Gitterpunkte. Da Felder als grundlegende Datenstrukturen sowohl von der Rechnerhardware als auch den Programmiersprachen und ihren Compilern effizient unterstützt werden, können numerische Verfahren auf SG einfach implementiert werden und erreichen hohe Rechenleistungen.

In gekoppelten Ansätzen werden SG bevorzugt als Teilgitter in oberflächennahen Bereichen der Geometrie eingesetzt. Die Oberfläche und ihre Normalenrichtung bilden in natürlicher Weise ein Koordinatensystem ("SG mit oberflächenangepassten Koordinaten"), dem das SG ohne Schwierigkeiten folgen kann, solange die Oberfläche glatt ist. Da diese Richtungen auch in den Lösungen reibungsbehafteter Strömungen ("boundary layer") dominant sind, passen Gitter und Lösung optimal zusammen. Da diese Bereiche sehr genau aufgelöst werden und deshalb eine hohe Anzahl von Maschen enthalten, wirkt sich die gute Rechenleistung, die auf SG erreicht werden kann, vorteilhaft aus.

Auf strukturierten Gittern lassen sich die bekannten Diskretisierungsansätze, Finite-Differenzen, Finite-Elemente und Finite-Volumen in natürlicher Weise anwenden.

So wie die Regelmäßigkeit strukturierter Gitter ihre wesentliche Stärke ist, stellt diese Eigenschaft auch das schwierigste Hindernis bei der Diskretisierung realistischer Geometrien dar. Weil solche Geometrien nicht würfelförmig sind bzw. nicht einfach auf einen Würfel abgebildet werden können, wurden immer aufwändigere Techniken entwickelt, um die Flexibilität zu erhöhen. So können durch Manipulationen an den Gitterrändern einfache Profile dargestellt werden (sogenannte C-Gitter, O-Gitter, H-Gitter). Hat eine Geometrie jedoch ein- bzw. ausspringende Ecken, die sich nicht auf eine Ecke des Würfels abbilden lassen (siehe z. B. [40]) oder sind die gegenüberliegenden Ränder gegeneinander verschoben (z. B. Gitter für Turbinen), entstehen sehr diffizile Optimierungsaufgaben für die Transformation bzw. die Punktverteilung im Inneren der Geometrie, die mit aufwändigen Lösungsverfahren behandelt werden, z. B. [57, 18].

Bei der Diskretisierung der Differentialgleichung auf dem Einheitsquadrat geht die Transformation bzw. ihre Ableitungen bis zur Ordnung der Differentialgleichung in die Diskretisierung mit ein (Kettenregel) und beeinflusst so den Diskretisierungsfehler. Deshalb ist es wünschenswert, dass der Abstand der Gitterlinien sich langsam ändert ("grid smoothness") und dass die Gitterlinien sich orthogonal schneiden ("grid skewness", "orthogonality"). Besondere Bedeutung kommt diesen Kriterien an den Geometrieoberflächen zu, an denen reibungsbehaftete Strömungen untersucht werden, sowie an den Ecken des Rechengebietes, vgl. z. B. [11].

Die Transformation wird entweder durch algebraische Ansätze (d. h. durch parametrisierte Kurvendarstellungen) für die Gitterlinien oder durch Lösung geeigneter (hyperbolischer oder elliptischer) Differentialgleichungen für die Transformation bestimmt. Diese Verfahren benutzen Tuningparameter in Form von Gewichtsfunktionen oder Quelltermen. Eine ganze Reihe solcher Verfahren sind in [63] beschrieben. Einerseits wird nun versucht, robuste Ansätze zu finden, die geeignete Gitter mit möglichst wenigen Parametern liefern, andererseits werden immer flexiblere Verfahren vorgeschlagen, um unbefriedigend diskretisierte Bereiche durch Einführung zusätzlicher Parameter zu beseitigen. Da auf diesem Weg Degenerationen nicht ausgeschlossen werden können, kommen interaktive Systeme zum Einsatz, die eine optische Kontrolle der Gitter oder die Manipulation von Gewichtsfunktionen ermöglichen, um Einfluss auf die Gittergenerierung zu nehmen. Solche Systeme

unterstützen die Anwendung erheblich, erfordern aber sowohl einen beträchtlichen Entwicklungsaufwand als auch ein hohes Maß an Erfahrung und Zeit seitens des Anwenders. In gekoppelten Ansätzen fällt dieser Aufwand für jedes Teilgitter an. Zusätzlich können Änderungen in einem Teilgitter Auswirkungen auf die Parameter in benachbarten Blöcken haben.

Die Koordinatentransformation läßt sich auf Kosten der Glattheit und Orthogonalität des Gitters so beeinflussen, dass sich die Gitterlinien verdichten oder Lösungsphänomenen folgen. Eine andere Schwierigkeit besteht aber darin, dass Verfeinerungen nur durch Hinzufügen ganzer Gitterlinien erreicht werden können. Dadurch wird eine Verfeinerung auch dort erzwungen, wo sie nicht benötigt wird. Alternativ können hierarchische Multiblockgitter eingesetzt werden, um lokale Verfeinerungen zu ermöglichen [45].

Eine wesentlich höhere Geometriekomplexität läßt sich beherrschen, wenn strukturierte Gitter im Rahmen gekoppelter Ansätze (Multiblock-, Chimera- oder Hybridgitter) eingesetzt werden. Diese werden in Abschnitt 2.4 diskutiert.

2.3.3 Kartesische Gitter

Kartesische Gitter können z. B. als Bereichssuchbaum (Quadtree, Octree) oder als hierarchisches Multiblockgitter implementiert werden. Dies ist notwendig, um durch Adaption die geometrische Inflexibilität der Maschen aufzufangen und den Aufwand der Oberflächendarstellung auf randnahe Bereiche zu begrenzen. Andernfalls würde das kleinste Detail der Geometrieoberfläche die Mindestauflösung des gesamten Gitters kontrollieren. Dadurch unterscheiden sich kartesische Gitter in ihrer Anwendung deutlich von strukturierten Gittern, obwohl bei beiden Gittertypen Vierecke bzw. Hexaeder als Maschen benutzt werden.

Durch die Orientierung an den Koordinatenlinien des Untersuchungsraumes wird die Gittererzeugung wesentlich vereinfacht, die Erzeugung des Raumgitters ist trivial. Dagegen ist das Einschneiden der Geometrieoberfläche in das Raumgitter vor Beginn der Simulation ein schwieriger Vorgang. Soll dies für beliebige Geometrien automatisch, d. h. ohne Intervention durch den Benutzer funktionieren, so muss einige Mühe aufgewendet werden. Eine Teilaufgabe dabei ist beispielsweise der Schnitt von Koordinatenlinien mit der Oberflächenbeschreibung der Geometrie und die Klassifikation dieser Schnitte. Die verwandte Aufgabe, eine Ebene mit einer beliebigen, polygonal berandeten Masche unter Berücksichtigung kleiner Objekte zu schneiden, ist eines der Resultate dieser Arbeit. Um das Auffinden sich schneidender Maschen bzw. Oberflächenstücke effizient zu gestalten, ist eine geeignete Suchstruktur notwendig. Die eventuell bereits verwendete Baumstruktur bietet dafür die notwendigen Voraussetzungen.

Da von vorne herein bereits geeignete Techniken der lokalen Adaption bereitgestellt wurden, um die Darstellung der Oberflächen zu ermöglichen, stellt auch die lösungsabhängige Verfeinerung kein grundsätzliches Problem dar. Eine Orientierung der Maschengeometrie an Lösungsphänomenen, die nicht parallel zu Koordinatenlinien sind, ist jedoch nicht möglich. Wird das kartesische Gitter durch ein hierarchisches Multiblockgitter dargestellt, so muss beachtet werden, dass die Adaption schräg zum Gitter verlaufender Lösungsphänomene, z. B. Schockfronten, entweder hohen Verschnitt oder sehr kleine Blöcke bedingen. Zwar enthalten 3D selbst Blöcke mit geringer Kantenlänge ausreichend Maschen um Nutzen für die Rechenleistung ziehen zu können, aber der Aufwand zur Verwaltung der Blockgrenzen steigt dann wesentlich an. So hat ein Block der Kantenlänge vier bereits $4^3 = 64$

Maschen, aber auch an jeder seiner sechs Seiten sechzehn Oberflächen, d. h. zur Kopplung des Blocks mit seiner Umgebung werden weitere $6 * 4^2 = 96$ Maschen benötigt, in denen Randbedingungen gespeichert werden müssen.

Um die gute Rechenleistung auch wirklich zu erhalten, müssen die verschachtelten Schleifen für die Bearbeitung aller Maschen eines Blockes gegebenenfalls manuell ersetzt werden. Dies belastet die Lesbarkeit der Programme über die komplexe Implementierung der benötigten Strukturen hinaus. Werden Baumstrukturen verwendet und für Finite-Volumen-Verfahren die Aufwärtskonnektivität von Seiten zu Kontrollvolumen nicht implementiert, so erfordert jede Flussberechnung einen Suchvorgang durch den Baum, um die benachbarten Maschen aufzufinden [69]. In diesem Fall ist eine Vektorisierung der Flussberechnung nicht möglich.

Da die Schnitte zwischen dem Bereichssuchbaum und der Geometrieoberfläche exakt ausgeführt werden, entstehen Schwierigkeiten mit sehr kleinen Kanten, Flächen und Zellen. Diese beschränken in der üblichen Weise den Zeitschritt expliziter Zeitschrittverfahren und erschweren die Implementierung der Randbedingungen. Aufgrund der approximativen Darstellung der Ränder sind außerdem Genauigkeitsverluste, z. B. im Vergleich zu strukturierten Gittern, zu erwarten, weshalb diese Technik bei reibungsbehafteten Flugkörpersimulationen bis jetzt nicht angewandt wird. Da die regelmäßige Aufteilung des Rechengebietes durch den Quad- bzw. Octree nicht mit den Notwendigkeiten der Diskretisierung korrespondieren muss, versuchen neuere Ansätze, das Rechengitter nachträglich zu vergröbern. In [16] wird von einer Halbierung der Anzahl der Gittermaschen berichtet, ohne dabei die Qualität der Lösung zu beeinträchtigen.

Die Regelmäßigkeit der Unterteilung erzeugt numerische Probleme, denen man zu entgehen versucht, indem das kartesische Gitter nicht mehr als Rechengitter sondern als Hintergrundgitter eingesetzt wird. Für die Darstellung des Rechengitters muss dann eine weitere, unstrukturierte Gitterrepräsentation bereitgestellt werden.

2.3.4 Unstrukturierte Gitter

Der Einsatz von Dreiecken bzw. Tetraedern als Gittermaschen wurde durch die Finite-Elemente-Diskretisierung motiviert, die zuerst unter Verwendung stetiger, stückweise differenzierbarer Funktionen formuliert wurde. Solche Funktionen lassen sich auf Dreiecken bzw. Tetraedern leicht darstellen, indem die Variablen in den Eckpunkten der Masche lokalisiert werden. Die Implementierung solcher Maschen erfordert die explizite Repräsentation der Nachbarschaftsbeziehungen zwischen den Maschen des Gitters, weil keine globale Sicht dieser Beziehungen existieren muss, wie sie für strukturierte Gitter typisch ist. Diese Tatsache ermöglicht die Repräsentation beliebig komplexer Geometrien.

Im Vergleich zu strukturierten Gittern benötigt der Zugriff auf benachbarte Maschen eine zusätzliche, indirekte Adressierung zur Ermittlung der Adresse des Nachbarn, bevor dessen Daten benutzt werden können. Da diese Nachbarn an "beliebiger" Stelle im Speicher stehen können, ist die Datenlokalität herabgesetzt, wodurch der Nutzen schneller Cache-Speicher geringer wird. Die stärkere Belastung des Speichersystems durch ein numerisches Verfahren auf unstrukturierten Gittern führt zu einer Verringerung der Floating-Point-Leistung im Vergleich zu einem Verfahren auf strukturierten Gittern.

Die Erzeugung eines unstrukturierten Gitters für ein Rechengebiet erfordert die Wahl einer Menge von Knoten (Eckpunkten) und eines Kriteriums für die Wahl der Maschen, da die

gegebenen Punkte auf viele verschiedene Arten zu Dreiecken bzw. Tetraedern verbunden werden können.

Das bekannteste Kriterium ist die Delaunay-Triangulierung, welche durch Verbindung derjenigen Knoten entsteht, die im Voronoi-Diagramm benachbart sind ("duales Gitter"), vgl. z. B. [6, 38]. Das Voronoi-Diagramm ordnet jedem Punkt P_0 der Punktmenge P die Punkte der Ebene bzw. des Raumes zu, die dem Punkt P_0 näher liegen als jedem anderen Punkt aus P. Die Dreiecke einer Delaunay-Triangulierung haben die Eigenschaft, dass der Umkreis der Masche keinen anderen Punkt der Punktmenge P enthält. Diese Eigenschaft wird benutzt, um die Punkte sequenziell in das Gitter einzufügen. Dazu werden adaptive (Bowyer-Watson-Algorithmus, Green-Sibson-Algorithmus) oder Advancing-Front-Techniken (Tanemura-Merriam-Algorithmus) benutzt. Die genannten Verfahren benötigen effiziente Suchstrukturen, z. B. zum Auffinden nächstliegender Punkte oder zum Erkennen von Frontüberlappungen.

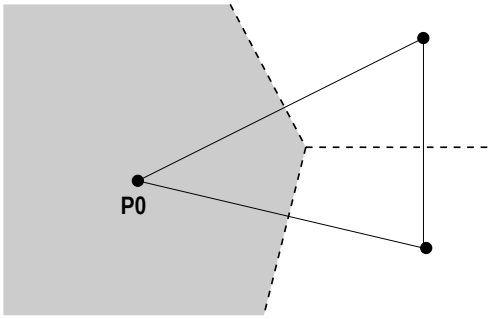


Abbildung 2.4: Dreieck mit Voronoi-Diagramm (gestrichelt)

Zur Erzeugung der notwendigen Punktmenge existieren ebenfalls verschiedene Möglichkeiten. Für eine Geometrie mit mehreren Komponenten, z. B. für ein mehrteiliges Tragflügelprofil, können die Knoten durch eine vorangehende strukturierte Gittergenerierung der Einzelkomponenten erzeugt werden. Ein anderer Ansatz ist die Erzeugung eines Quadtrees der Geometrie, dessen Knoten zur Gittergenerierung herangezogen werden können. Advancing-Front-Techniken erzeugen Knoten und Konnektivität gleichzeitig. Neue Knoten werden dabei aus lokalen geometrischen Betrachtungen ermittelt. In einer bestehenden Delaunay-Triangulierung können Dreiecke durch Einfügen neuer Knoten verfeinert werden, die nicht einer vorgegebenen Verteilung der Dreiecksgrößen entsprechen (Steiner-Triangulierung). Die genannten Methoden zur Punkterzeugung beruhen auf einer vorzugebenden Verteilung der Dreiecksgrößen, welche beispielsweise auf einem Quadtree (Hintergrundgitter) dargestellt werden kann. Die Verteilung selbst kann analytisch oder durch Lösung einer partiellen Differentialgleichung auf dem Hintergrundgitter bestimmt werden.

Die entstandene Triangulierung kann abschließend geglättet werden, um Unregelmäßigkeiten des Erzeugungsprozesses auszugleichen. Dazu werden Federanalogien eingesetzt. Die Kanten des Gitters als Federn aufgefasst, welche Kräfte auf die Punkte ausüben. Die Position der Punkte wird so bestimmt, dass das Federsystem im Gleichgewicht ist.

Eine weitere Aufgabe bei der Erzeugung des Gitters ist die (Wieder-) Herstellung der Geometrieoberflächen. Da der Triangulierungsprozess nur Punkte und Maschen kennt, muss in einem separaten Arbeitsgang dafür gesorgt werden, dass die Oberflächen der Geometrie in der Triangulierung dargestellt werden ("constrained-Delaunay-triangulation"). Dies kann

entweder am Anfang der Gittergenerierung geschehen oder nachträglich durch Verändern der erzeugten Triangulierung. Dazu werden die fehlenden Kanten durch "edge-swapping" in die Triangulierung eingefügt. Das analoge Vorgehen in 3D ("face-swapping") kann jedoch an der Punktverteilung scheitern; es existieren unveränderliche ("non-swappable") Konfigurationen von fünf Punkten. In einem solchen Fall muss die Diskretisierung der Geometrieoberfläche verändert werden.

Eine wesentliche Stärke unstrukturierter Gitter stellt die Möglichkeit dar, das Gitter lösungsangepasst zu verfeinern, zu vergröbern oder die Gitterknoten zu verschieben. Verfeinerungs- und Vergröberungstechniken können in Anlehnung an die Algorithmen zur Gittererzeugung implementiert werden. Die Re-Tetrahedrierung eines durch Entfernen eines Gitterknotens entstandenen Polyeders in 3D kann jedoch scheitern. Ferner sind der Vergröberung eines Gitters Grenzen durch die Auflösung der Geometrieoberfläche gesetzt, weil an jede Kante der Oberflächenbeschreibung ein Dreieck angrenzen muss. Dies kann insbesondere für Multilevel-Verfahren hinderlich sein. Die Erzeugung gestreckter Dreiecke für viskose Strömungssimulationen ist eine weitere Herausforderung.

Eine detaillierte Darstellung der Techniken und Schwierigkeiten der Generierung unstrukturierter Gitter findet sich in [38].

2.3.5 Unstrukturierte Gitter mit mehreren Maschentypen

Bei der Simulation reibungsbehafteter Strömungen ist besondere Sorgfalt bei der Diskretisierung der Grenzschichten in der Nähe von Wandrändern notwendig. Dieser Anforderung wird bei der Gittergenerierung durch stark gestreckte Maschen in Wandnähe Rechnung getragen. Die Schwierigkeiten bei der Simulation solcher Strömungen auf gestreckten Dreiecks-/Tetraedergittern haben die Erweiterung unstrukturierter Gitter um weitere Maschentypen motiviert, z. B. um Rechtecke, Prismen, Pyramiden und Hexaeder. Für die Erzeugung von Vierecken, Prismen und Hexaedern werden vorzugsweise Techniken der strukturierten Gittergenerierung oder Advancing-Front-Techniken eingesetzt [29]. Pyramiden werden als Pufferzone zwischen Tetraedern und Hexaedern eingesetzt, siehe Abschnitt 2.4.2.

Die Verwendung prismatischer Maschen in Wandnähe ermöglicht die Erzeugung gestreckter Maschen, ohne Degenerationen (z. B. große oder kleine Winkel) zu verursachen. Aufgrund der eingesetzten Maschentypen werden diese Gitter als Hybridgitter bezeichnet, obwohl sie von ihrer Struktur her die wesentlichen Eigenschaften eines unstrukturierten Gittertyps zeigen. Prismatische Maschen können einfach zwischen eine triangulierte Geometrieoberfläche und ein Raumgitter aus Tetraedern eingebracht werden, da die Oberflächen der Maschen zueinander kompatibel sind. Sind hängende Knoten bzw. Kanten unzulässig, so ist eine lokale Verfeinerung innerhalb der Prismenschicht nur durch Einschieben sehr flacher Tetraeder möglich [21]. Schwierigkeiten können auch an den Schnittkanten verschiedener Oberflächen auftreten. Dort können die auf die auf die verschiedenen Oberflächen aufgesetzten Prismen miteinander in Konflikt geraten.

Um eine einheitliche Repräsentation des Gitters trotz unterschiedlicher Maschentypen zu erreichen, werden kanten-basierte Datenstrukturen eingesetzt. Die Kanten sind dabei die Kanten des dualen Gitters, welches aus der Verbindung benachbarter Maschenschwerpunkte entsteht.

2.4 Gekoppelte Gitter

Werden mehrere Teilgitter aneinandergesetzt, so lassen sich komplexere Geometrien einfacher repräsentieren. Diese Ansätze werden zur Kopplung mehrerer strukturierter Gitter ("Multiblockgitter"), zur Kopplung strukturierter mit unstrukturierten Gittern ("Hybridgitter") oder zur Kopplung überlappender Gitter ("Chimera-Gitter") eingesetzt.

2.4.1 Multiblockgitter

Werden mehrere SG aneinandergesetzt, so entsteht dadurch für die Gittererzeugung nur dann ein Vorteil, wenn nicht die Kontinuität der Gitterlinien über die Blockgrenzen hinweg erzwungen wird. Andernfalls sind die Teilgitter durch strukturierte Nachbarschaftsbeziehungen miteinander verbunden und die Knoten an der Blockgrenze sind identisch. Dann können beide Blöcke durch einen einzigen Block dargestellt werden, d. h. dass die Aufteilung in Teilgitter eine rein organisatorische Maßnahme ist, etwa zur Aufteilung des Gitters auf den verteilten Speicher eines Parallelrechners.

Sind an den Blockgrenzen flexiblere Nachbarschaftsbeziehungen möglich, können komplexe Geometrien einfacher vernetzt werden. Die Implementierung flexibler Nachbarschaftsbeziehungen für die Blöcke beinhaltet genau die gleiche Komplexität wie bei unstrukturierten Ansätzen.

Der Preis für die zusätzliche Flexibilität besteht in Interpolationsfehlern, die an diskontinuierlichen Blockgrenzen entstehen und die Konservativität des Gesamtverfahrens beschädigen. Obwohl die Verletzung der Konservativität bei geeigneter Wahl der Blockgrenzen keine wesentliche Rolle spielt, macht das Fehlen zuverlässiger Qualitätskriterien diese Interpolationsfehler unkontrollierbar. Die Aufteilung der Geometrie in Teilblöcke ist deshalb nicht automatisierbar. Sie bleibt dem Anwender überlassen und erfordert einen guten Überblick über die Konsequenzen der Blockgrenzenwahl.

Die Erzeugung von Multiblockgittern ist aufwändig, weil viele Details spezifiziert werden müssen. Durch die große Zahl von Parametern, die eine solche Struktur beschreiben, ist auch die Wiederverwendbarkeit der Gitter oder der Blocktopologie eingeschränkt [17].

2.4.2 Hybridgitter

Die Verbindung strukturierter und unstrukturierter Gitter bei der Gittererzeugung erlaubt die spezifischen Vorteile beider Ansätze zu nutzen.

So werden Gebiete mit strukturierten Gittern vernetzt, die sich dafür besonders gut eignen, so dass die Gitter mit geringen Aufwand erzeugt werden können. Bei reibungsbehafteten Strömungen sind dies die Gitter in der Nähe von festen Wänden, da die strukturierten Gitter mit der Lösung dort gut harmonieren und durch die Geometrieoberfläche natürliche Koordinatenrichtungen gegeben sind. Die Bereiche zwischen diesen Teilgittern werden dann durch unstrukturierte Gitter aufgefüllt.

Shaw, Georgala and Childs beschreiben in [53] ein elf-stufiges Vorgehen zur Erzeugung hybrider Initialgitter. Dabei stellt jeder einzelne Schritt eine Investition in Forschung und

Entwicklung dar und erfordert Eingriffe oder Interaktionen mit dem Anwender. Strukturierte und unstukturierte Gitterblöcke werden dabei durch eine "Pufferzone" aus Pyramiden interpolationsfrei verbunden. Auf die Oberfläche des strukturierten Gitters wird eine Schicht Pyramiden aufgetragen und deren dreieckige Seiten mit Tetraedern aufgefüllt, so dass für die Erzeugung des unstrukturierten Bereichs keine Pyramidenkanten in der Randbeschreibung auftauchen.

Die Kopplung von Gitterblöcken durch geeignete Anpassung der Maschen an den Blockgrenzen erlaubt es, die gewählte Diskretisierung der Differentialgleichung auch an der Blockgrenze zu benutzen. Die Kopplung stellt sich also als Gittergenerierungsaufgabe dar.

2.4.3 Chimera-Technik

Sind die Oberflächen benachbarte Gitterblöcke zueinander nicht kompatibel, so können Interpolationstechniken eingesetzt werden, um die Randbedingungen an den Oberflächen der Gitterblöcke bereitzustellen. Sind an den Blockgrenzen auch Lücken oder Überlappungen zwischen den Gitterblöcken zugelassen, so spricht man von Chimera-Gittern, "Overset Grids", oder überlappenden Gittern ("overlapping grids").

Dieser Ansatz ist besonders für Geometrien mit mehreren Komponenten oder relativ zueinander bewegten Komponenten attraktiv, weil auf eine Anpassung der Gitter verzichtet werden kann. Dabei wird jeder Komponente mit einem eigenen Umgebungsgitter versehen. Dazu werden in der Regel strukturierte Gitter mit oberflächenangepassten Koordinaten eingesetzt. Nach der Identifikation der Überlappungsbereiche werden diese aus den Teilgittern ausgeblendet. An die dadurch entstehenden Blockgrenzen können mehr als zwei Blöcke angrenzen. Die Randwerte für die Teilblöcke werden durch Interpolation aus den benachbarten Blöcken ermittelt.

Die Verletzung der Konservativität und der Einfluss der Interpolation auf die Qualität der Lösung in den Überlappungsbereichen werden häufig kritisiert. Andererseits wurden Verfahren vorgeschlagen, die Schockfronten in den Überlappungsbereichen ohne sichtbare Probleme berechnen [49].

2.5 Diskussion der Gittereigenschaften

Soll der Vorgang der Gittergenerierung im gleichen Maße automatisiert werden, wie es für ein numerisches Verfahren selbstverständlich ist, muss die Struktur dieses Vorgangs den Möglichkeiten einer Software-Lösung angepasst werden. In einem Code können Intuition und Anwendungserfahrung nur schlecht repräsentiert werden, "Fleißaufgaben" stellen hingegen kein Hindernis dar.

In diesem Abschnitt sollen charakteristische Eigenschaften von Gittergenerierungstechniken herausgestellt werden, die wesentlich zur Komplexität der Gittergenerierung beitragen. Von herausragender Bedeutung für die Gittergenerierung ist die Beschränkung auf konvexe, einfach zusammenhängende Maschen mit fester Anzahl von Oberflächen sowie die mit dem Schlagwort "Gitterqualität" bezeichnete Strategie, Mängel in den Diskretisierungs- und Lösungstechniken auf die Gittergenerierung abzuwälzen. Unter diesen Randbedingungen wird die Erzeugung eines Rechengitters zu einem vielstufigen Transformationsprozess. Dabei müssen mehrere Gitter für die Geometrie (Hintergrundgitter) aufgebaut werden,

um Detailprobleme zu lösen, die erst durch die Struktur des Prozesses selbst hervorgerufen werden.

2.5.1 Maschentypen mit fester Anzahl von Oberflächen

Da die meisten Diskretisierungstechniken nur auf bestimmte Maschentypen anwendbar sind, werden die zulässigen Maschentypen in der Regel nach der favorisierten Diskretisierungstechnik ausgewählt. Diese Maschentypen (Dreiecke, Vierecke, Tetraeder, Prismen, Hexaeder) sind durch eine feste Anzahl von Oberflächen ausgezeichnet. Darüberhinaus wird die Struktur der diskreten Gleichungssysteme von der Regelmäßigkeit der Maschen stark beeinflusst. So lassen sich beispielsweise Abhängigkeiten von der Orthogonalität der Maschen bei strukturierten Gittern bzw. vom größten Winkel bei unstrukturierten Gittern oder von Größenänderungen der Maschen nachweisen.

Der Ansatz, die Maschen nach der Diskretisierungstechnik und den Eigenschaften der diskreten Gleichungssysteme zu wählen, ist eine wichtige Ursache für den Engpass, den die Gittergenerierung heute für den routinemäßigen Einsatz numerischer Simulationen bildet. Nach dem heutigen Stand der Technik erfordert die Gittererzeugung *unter diesen Einschränkungen* einen hohen Einsatz seitens des Anwenders, entweder in Form manueller Tätigkeiten oder in Form von Fachwissen über die Bedienung von Werkzeugen, die diese Tätigkeiten unterstützen. Dabei stellt die Verfügbarkeit geeigneter Werkzeuge ein weiteres, eigenes Problem dar, das Anlass zu großangelegten Entwicklungsbemühungen wie dem "National Grid Project" gibt [47].

Ein Problemkreis, der mit der festgelegten Anzahl von Oberflächen zusammenhängt, ist die algorithmische Kopplung zwischen der Erzeugung der Oberflächen- und der Raumdiskretisierung. Da die Anzahl der Oberflächen einer Masche festgelegt ist, müssen die Oberflächen- und Raumgitter aneinander angepasst werden, um überhaupt ein zulässiges Gitter zu erhalten. Überdies wiederholt sich in 3D die Einschränkung der Oberflächenzahl bei den Oberflächen der Maschen (Dreiecke, Vierecke). Die Flächenstücke der Oberflächendiskretisierung und die Oberflächen der zulässigen Maschen müssen kompatibel gewählt werden, damit das Oberflächengitter als Randbedingung für die Erzeugung des Raumgitters dienen kann. So hat ein unstrukturiertes Initialgitter wenigstens so viele Maschen wie Oberflächen, bevor die Geometrie in einer zulässigen Darstellung repräsentiert werden kann.

Die Wahl von Maschen mit fester Anzahl von Oberflächen bringt für Algorithmen zur lokalen Verfeinerung erhebliche Schwierigkeiten mit sich. Wird eine Masche mit ihren Oberflächen verfeinert, müssen nun auch die angrenzenden Maschen verfeinert werden (Folgeteilungen), weil sie sonst zu viele Oberflächen haben. Damit auf diese Weise keine uniforme Verfeinerung erzwungen wird, werden die angrenzenden Maschen so verfeinert, dass keine weiteren Oberflächen geteilt werden. Da diese Maschen dann für eine weitere Verfeinerung ungeeignet sind, müssen solche Folgeteilungen vor einer Verfeinerung wieder rückgängig gemacht werden.

Die Problematik einer oberflächenkompatiblen Verbindung zwischen Tetraedern und Hexaedern bei der Erzeugung hybrider Gitter und die Notwendigkeit von Interpolationstechniken bei Multiblockgittern wird durch die festgelegte Anzahl von Maschenoberflächen bewirkt.

In der vorliegenden Arbeit wird deshalb die umgekehrte Frage gestellt: Welche Diskretisierungstechniken sind auf einfach zu erzeugende Gitter anwendbar? Der beschriebene Zusammenhang motiviert die in dieser Arbeit entwickelte Repräsentation des Gebietes durch polygonale bzw. polyhedrale Maschen. Ein "beliebiges" Gebiet ist nach Diskretisierung seiner Oberfläche ein Polygon bzw. Polyeder und kann sofort als Gitter benutzt werden. Damit reduziert sich das Geometrie-Preprocessing auf die Bereitstellung einer diskretisierten Oberflächenbeschreibung der Geometrie.

2.5.2 Maschentypen

Die Komplexität einer Geometrie kann beispielsweise durch die Anzahl der Flächenstücke des Oberflächengitters, durch die Anzahl der Zusammenhangskomponenten der Geometrie oder durch die Größe einer Zerlegung des Rechengebietes in konvexe oder sternförmige Teilgebiete charakterisiert werden. Dabei bedeutet eine Zunahme einer dieser "Kenngrößen" erhöhte Anforderungen an die Algorithmen der Gittergenerierung. Dies rührt neben der bereits diskutierten Beschränkung auf Maschen mit fester Anzahl von Oberflächen daher, dass die benutzten Maschentypen selbst einfach zusammenhängend und konvex sind.

Aus diesem Grund muss das Rechengebiet im Verlauf der Gittergenerierung so lange in Teilgebiete geringerer Komplexität zerlegt werden, bis die Teilgebiete durch zulässige Maschen darstellbar sind. Damit diese Kenngrößen einer algorithmischen Kontrolle zugänglich sind, müssen sie explizit ermittelt werden. Die Aufgabe, das Rechengebiet in konvexe Teilgebiete zu zerlegen, ist jedoch äquivalent zu einer Triangulierung des Gebietes. Da eine Triangulierung nur unter Verwendung der Knoten auf der Oberfläche des Rechengebietes zu einer numerisch ungünstigen Zerlegung führt, müssen von vorneherein Mengen innerer Knoten gewählt werden. Die Algorithmen zur Triangulierung bringen selbst weitere Probleme mit sich, etwa die Forderung, sie in exakter Arithmetik auszuführen.

Werden zur Darstellung eines Gitters also auch nicht-konvexe Maschen mit mehrfach zusammenhängender Oberfläche zugelassen, reduziert sich die Komplexität der Gittergenerierung allein deshalb, weil verschiedene, algorithmisch schwer zu beherrschende Probleme der Geometrie nicht mehr gelöst werden müssen. Es zeigt sich, dass eine nicht-konvexe, mehrfach zusammenhängende Masche leichter in zwei Maschen dieser Klasse zerlegt werden kann als in eine notwendig große Zahl konvexer, einfach zusammenhängender Maschen. Eine solche Zerlegung in zwei Teile reduziert tendentiell auch die Komplexität der Maschen, so dass schließlich numerisch geeignete Maschen entstehen.

Ein anderer Aspekt der Maschenwahl wird deutlich, wenn aus numerischen Gründen besondere Anforderungen an den Maschentyp gestellt werden, etwa für viskose Ränder, an Unstetigkeiten der Lösung oder bei der Ausrichtung des Gitters an Stromlinien. Während ein Viereck/Hexaeder leicht zu strecken ist, führt die gleiche Transformation für ein Dreieck/Tetraeder zu ungünstigen Maschenformen. Dies hat bereits zu einer Erweiterung der zulässigen Maschentypen geführt (Tetraeder-Prismen-Gitter). Die Auflösung einer Randschicht mittels kartesischer Gitter kann nicht erreicht werden, ohne gleichzeitig in alle Raumrichtungen zu verfeinern; die Anpassung des Gitters an die Geometrieoberfläche oder an andere, schräg zum Gitter verlaufende Phänomene ist nicht möglich, ohne den Kontext des kartesischen Gitters zu verlassen.

2.5.3 Gitterqualität

In Veröffentlichungen, die sich mit der Erzeugung von Gittern für numerische Strömungssimulationen befassen, findet sich regelmäßig der Hinweis, dass eine hohe Gitterqualität notwendig sei, weil die Funktion numerischer Verfahren und die Genauigkeit der Lösungen andernfalls beeinträchtigt würde. Weniger häufig findet sich der Hinweis [28, 38], dass der Begriff der Gitterqualität nur vage umrissen ist und in der Regel auf geometrische Eigenschaften der Maschen, z. B. Glattheit oder Orthogonalität, abhebt, während andere Aspekte, wie etwa das Diskretisierungsschema, unbeachtet bleiben. Das Fehlen geeigneter Fehlerschätzer leistet dieser Sichtweise noch Vorschub.

Der Zusammenhang zwischen der Forderung nach Gitterqualität und den eingesetzten Diskretisierungstechniken soll anhand von Finite-Differenzen-Diskretisierungen diskutiert werden.

Untersucht man die Konsistenzordnung der diskreten ersten zentralen Ableitung (1D), so ermittelt man mit Hilfe einer Taylorreihenentwicklung, dass die erste Ableitung am Entwicklungspunkt für ein quadratisches Polynom exakt bestimmt wird, wenn die benachbarten Punkte gleich weit entfernt sind, d. h. dass die Approximation der ersten Ableitung am Auswertungspunkt von zweiter Ordnung ist. Weichen die Abstände der benachbarten Punkte voneinander ab, so geht eine Ordnung verloren.

Diese Überlegung legt den Schluss nahe, dass die Verwendung unregelmäßiger Punktverteilungen einen wesentlichen Verlust an Lösungsqualität zur Folge hat, die durch Verbesserung der Gitterqualität aufgefangen werden muss. Eine genauere Analyse der Fehlerterme erster Ordnung zeigt, dass sie von der Änderung der Punktabstände abhängen. Dies rechtfertigt die gängige Forderung nach glatten Netzen, in denen die Variation der Punktabstände möglichst gering gehalten wird. Dadurch bleibt der verursachte Fehler klein, sofern die zweite Ableitung beschränkt ist.

Kann die Variation der Gitterweite so gewählt werden, dass sie selbst von zweiter Ordnung gegen Null geht [27], so erhält man wieder eine Approximation zweiter Ordnung. Auch dieser Ansatz wälzt die Schwierigkeiten auf die Gittergenerierung ab.

Ersetzt man die zentrale erste Ableitung durch eine geeignete Linearkombination aus vorwärts- und rückwärtsgerichteten Differenzenquotienten, so erhält man die zweite Ordnung unabhängig von der Wahl des Gitters zurück. Dieser Ansatz führt zu dem Schluss, dass der zentrale erste Differenzenquotient nicht die optimale Diskretisierung der ersten Ableitung auf unregelmäßigen Gittern darstellt und rückt den Einfluss der Diskretisierungstechnik auf die Qualität der Lösung in den Mittelpunkt der Betrachtung.

Diese Überlegung lässt sich auf einem strukturierten Gitter in mehreren Raumdimensionen nicht genauso wiederholen, weil ein quadratisches Polynom dann mehr Koeffizienten hat, als Nachbarwerte im Gitter bereitstehen. Trotzdem hat die Wahl der Linearkombinationen, die den Gradienten bilden, Einfluss auf die Auslöschung von Fehlertermen höherer Ordnung. Die Forderung nach Gittern hoher Qualität steht also in Zusammenhang mit Mängeln in den verwendeten Diskretisierungstechniken. Insofern liefert die Entwicklung robusterer Diskretisierungstechniken einen wichtigen Beitrag zur Gittergenerierung: werden weniger hohe Anforderungen an das Gitter gestellt, können diese leichter automatisch erzeugt werden.

2.5.4 Polyhedrale Maschen

Die oben dargestellten Schwierigkeiten bei der Erzeugung von Gittern aus konvexen, einfach zusammenhängenden Maschen mit fester Anzahl von Oberflächen motivieren den in dieser Arbeit verfolgten Ansatz, Gitter aus polyhedralen Maschen aufzubauen.

Ausgehend von einer konsistenten Diskretisierung der Geometrieoberfläche wird das Ziel einer automatischen Gittergenerierung tatsächlich erreicht. Gitter und Lösung entstehen in einem integrierten Lösungsprozess, ohne den Anwender mit der Erzeugung des Gitters in irgendeiner Weise zu konfrontieren. Der entwickelte Ansatz bietet also eine Perspektive für die Entwicklung von CFD-Werkzeugen, die ohne die heute unumgängliche, detaillierte Kenntnis numerischer Methoden und ihrer Probleme eingesetzt werden können.

Im folgenden Kapitel 3 wird die gewählte Gitterstruktur spezifiziert und eine Gitterverfeinerungsstrategie entwickelt. Der beschriebene Ansatz kommt dabei mit nur einer einzigen Darstellung des Gitters aus, wodurch die Implementierung stark vereinfacht wird. Die Gittergenerierung ist im Gegensatz zu den Techniken der strukturierten [53] oder unstrukturierten Gittererzeugung [38] ein homogener, einstufiger Prozess.

2.6 Adaptive Verfahren

Die numerische Berechnung genauer Lösungen für Strömungsprobleme erfordert stellenweise eine äußerst feine Auflösung des Gitters, um starken Änderungen der Lösung folgen zu können, wie sie z. B. an Schocks oder in der Grenzschicht viskoser Strömungen auftreten. Existieren in der Lösung Phänomene, deren Längenskalen um mehrere Größenordnungen voneinander abweichen, so wird die gleichmäßig feine Diskretisierung des gesamten Rechengebietes nach dem kleinsten Lösungsphänomen unökonomisch oder unmöglich, wenn die benötigten Speicher- oder Rechenzeitresourcen nicht vorhanden sind.

Deshalb haben sich adaptive Verfahren, deren Gitter an die verschiedenen Längenskalen der Lösung angepasst werden können, für die numerische Simulation von Strömungsproblemen durchgesetzt. So stellt Löhner fest [35], dass die Fähigkeit zur Adaption darüber entscheidet, ob derartige Fragestellungen mit aktzeptabler Genauigkeit in vernünftiger Zeit gelöst werden können.

Ein adaptives Verfahren ist aus drei Komponenten aufgebaut: einer Adaptionsstrategie, einem Fehlerindikator und einem Kriterium, welches durch die Adaption erreicht werden soll. Häufig besteht dieses Kriterium in der Gleichverteilung des Fehlers auf dem Rechengebiet oder in absoluten Fehlerschranken.

2.6.1 Adaptionstechniken

Zur Adaption wurden verschiedene Techniken vorgeschlagen [35].

Die lokale Verfeinerung und Vergröberung des Gitters wird als “h-Methode” bezeichnet. Bei der Verfeinerung werden vorhandene Maschen des Gitters durch mehrere neue, kleinere Maschen ersetzt, bei der Vergröberung werden mehrere vorhandene Maschen durch eine neue, größere Masche ersetzt. Dabei werden häufig solche Maschen vergröbert, die zuvor geteilt wurden. Dies motiviert die Organisation der Maschen in hierarchischen Datenstrukturen [45, 7].

Bei der "r-Methode" ("repositioning") werden nicht die Nachbarschaftsbeziehungen sondern die Koordinaten der Knoten verändert. Deshalb ist diese Technik die Grundlage für die Adaption strukturierter Gitter.

Eine adaptive Steuerung der Diskretisierungsordnung ("p-Methode") wurde in dieser Arbeit nicht in Betracht gezogen, da eine Erhöhung nicht einfach zu erreichen ist. Eine Verringerung der Ordnung verspricht ebenfalls keinen Gewinn, da die Berechnung der linearen Rekonstruktion keinen großen Beitrag zur Rechenzeit leistet.

2.6.2 Adaptionsindikator

Der zuverlässigste Ansatz für die Wahl eines Adaptionsindikator ist die Auswertung eines Fehlerschätzers, der den Diskretisierungsfehler misst. A posteriori Fehlerschätzer, die den Diskretisierungsfehler durch das Residuum abschätzen sind für Finite-Volumen-Verfahren für die Euler-Gleichungen jedoch nicht einfach zu erhalten, vgl. z. B. [56, 59].

Daher wird häufig auf einfachere Techniken ("ad hoc"-Kriterien) zurückgegriffen, deren Zuverlässigkeit aber angezweifelt wird [35, 38, 59], weil kein gesicherter Zusammenhang zum Diskretisierungsfehler besteht. Dies wird offensichtlich, wenn man adaptierte Gitter vergleicht, die mit unterschiedlichen Indikatoren erzeugt wurden. Folgende Ansätze werden häufig betrachtet:

- Betrachtung von Gradienten oder Differenzen: in der Lösung werden bestimmte Phänomene, wie Schocks, identifiziert, indem Ableitungen oder Differenzen von Kenngrößen (z. B. Dichte, Druck, Machzahl, Entropie) ermittelt werden. Dabei wird vorausgesetzt, dass der Verlauf der Lösung in den übrigen Bereichen auf dem vorhandenen Gitter bereits ausreichend genau ermittelt wurde. Die Eigenschaften verschiedener Indikatoren sind z. B. in [69] zusammengestellt.

- Untersuchung der Approximationseigenschaften der Lösung: dabei wird eine glatte Lösung vorausgesetzt, die durch durch das numerische Verfahren bis zur Ordnung p approximiert wird. Man nimmt an, dass der Diskretisierungsfehler durch Ableitungen der Ordnung $p+1$ dargestellt werden kann und bestimmt diese Ableitungen näherungsweise.

- Alternative Diskretisierung: die vorhandene Lösung kann in eine andere Diskretisierung eingesetzt werden, deren Residuum mit dem vorhandenen Residuum verglichen werden kann. Beispielsweise kann mit den in Abschnitt 4.3 beschriebenen Rekonstruktionstechniken $\operatorname{div} F(\varphi)$ aus dem Gradient der Flussfunktion F ermittelt werden.

- Alternatives Gitter: durch Vergleich der Lösung auf verschieden feinen Gittern (Richardson Extrapolation) oder durch Vergleich der Residuen mit den Residuen auf dem dualen Gitter.

Kapitel 3

Unstrukturierte Gitter aus polyhedralen Maschen

In diesem Kapitel wird eine formale Spezifikation einer Gitterrepräsentation entwickelt, die als Grundlage eines Finite-Volumen-Verfahrens (FVM) dienen kann und eine einfache Schnittstelle zu Geometrien bietet, die mit CAD-Werkzeugen erstellt wurden (Abschnitt 3.1). Darauf aufbauend werden Techniken der Gitteradaption, Verfeinerung und Vergröberung behandelt und ihre Implementierung unter Berücksichtigung der Anforderungen einer Finite-Volumen-Diskretisierung höherer Ordnung beschrieben. Abschnitt 3.2 stellt ein geeignetes Verfeinerungskonzept vor, das in Abschnitt 3.3 im Hinblick auf eine effiziente Implementierung weiterentwickelt und in Abschnitt 3.4 präzisiert wird. Die Abschnitte 3.5 und 3.6 behandeln die Vergröberung von Gitterobjekten sowie die geometrische Deformation von Gitterobjekten ("r-refinement"). Schließlich werden die charakteristischen Eigenschaften des gewählten Ansatzes zusammengefasst und denen anderer Strukturen gegenübergestellt (Abschnitt 3.7).

3.1 Beschreibung der Gitterstruktur

3.1.1 Das Gitter als Graph

Ein Gitter wird hier im Wesentlichen als Multi-Graph auf Knoten, Kanten, Facetten und Zellen (*Gitterobjekten*) aufgefasst. Die Kanten des Graphen repräsentieren dabei die Nachbarschafts- oder Konnektivitätsrelationen. Die im Folgenden vorgestellte Gitterrepräsentation ist zur Darstellung von ein-, zwei- und dreidimensionalen Gebieten geeignet. Die Flexibilität bei der Darstellung von Maschen stellt die wesentliche Neuerung und Stärke der vorgeschlagenen Gitterstruktur dar.

Aufgrund ihrer Flexibilität enthält die Gitterrepräsentation nur wenig direkte Information über die Gestalt der Gitterobjekte. Geometrisch motivierte Begriffsbildungen, die auf impliziten Annahmen über die Gestalt der Maschen beruhen, bedürfen einer grundlegenden Neuformulierung. Im Vergleich dazu erlauben herkömmliche strukturierte und unstrukturierte Gitter *nach ihrer Erzeugung* eine gute Kontrolle über die Gestalt der Maschen. Die Erzeugung dieser Information aber ist es, die zu einem großen Teil für die Komplexität

der Gittergenerierung verantwortlich ist und deren Bereitstellung die Grundvoraussetzung dafür ist, eine gegebenes Gebiet überhaupt repräsentieren zu können.

Der Verzicht auf die explizite Repräsentation dieser Information vereinfacht den Prozess der Gittergenerierung wesentlich. Die Erzeugung von Initialgittern entfällt, weil die Problemgeometrie direkt als Masche dargestellt werden kann, nachdem ihre Oberfläche diskretisiert ist. Die Erzeugung des Oberflächengitters ist unabhängig von der Erzeugung des Raumgitters. Die vorgeschlagene Gitterrepräsentation ist einer automatischen, d. h. programm-gesteuerten Gittergenerierung zugänglich, weil sie sich aus Sicht einer Implementierung homogen darstellt. So können lösungsabhängige Gitter zusammen mit der Lösung durch selbst-adaptive Verfahren erzeugt werden.

3.1.2 Hierarchische, rekursive Beschreibung des Gitters

Die zentrale Idee des vorgestellten Ansatzes besteht darin, ein Gebiet des R^d mit stückweise glattem Rand rekursiv durch seine Oberflächen zu beschreiben. So wird ein dreidimensionales Gebiet (*Zelle*) von seinen berandenden Oberflächen begrenzt, die Oberflächen (*Facetten*) werden durch ihre Randkurven (*Kanten*) und diese wiederum durch ihre Anfangs- und Endpunkte (*Knoten*) begrenzt. Die Begriffe Knoten, Kante, Facette und Zelle beschreiben *geometrische Objekte* des Gitters in der Reihenfolge aufsteigender Dimension. Dabei drückt sich die Tatsache aus, dass Kanten durch Knoten, Facetten durch Kanten und Zellen durch Facetten rekursiv beschrieben werden können.

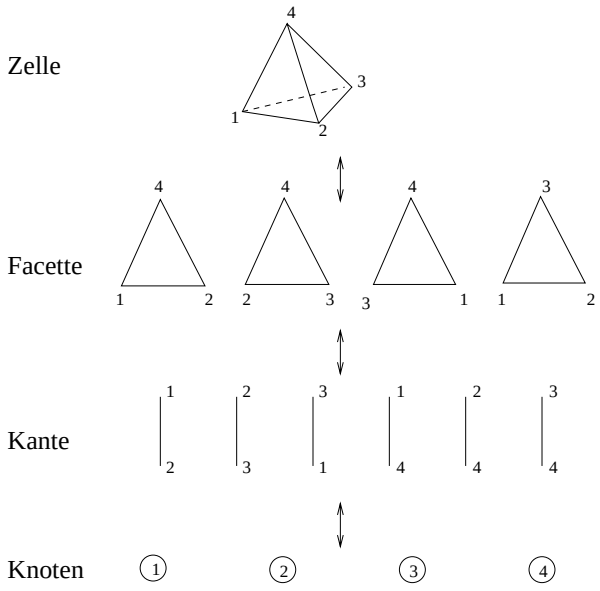


Abbildung 3.1: Objekthierarchie

Alternativ kann die Objekthierarchie aus dem Blickwinkel des Finite-Volumen-Verfahrens in der Reihenfolge absteigender Dimension betrachtet werden, dessen Begriffsbildungen das *Kontrollvolumen* und seine Oberflächen, die *Seiten*, sind. Dabei meint ein Kontrollvolumen ein geometrisches Objekt der Dimension des Problemraumes. So wird ein Kontrollvolumen und seine Seiten im eindimensionalen Fall durch eine Kante und ihren Anfangs- und Endpunkt repräsentiert, in 2D durch eine Facette und deren Kanten und in 3D durch eine Zelle und deren Facetten.

Die geometrischen Objekte o des Gitters werden durch ihre Oberflächenobjekte p charakterisiert, d. h. es gibt eine "besteht-aus"-Relation zwischen o und p. Diese Relation wird im Folgenden als (Abwärts-) Konnektivität bezeichnet.

Darüber hinaus benötigt jedes Objekt Informationen, die es erlauben, das Innere vom Äußeren des Objektes zu unterscheiden. Zu diesem Zweck wird die Konnektivität um Orientierungen erweitert. Diese vermitteln Kanten eine Richtung, Facetten einen Umlaufsinn und Zellen eine nach außen gerichtete Oberflächennormale. Damit lassen sich die im Kontext eines Finite-Volumen-Verfahrens notwendigen Oberflächenintegrationen durchführen (siehe Anhang) und Informationen über die Lage des Innen- bzw. Außenbereichs eines Objektes ermitteln, wie sie im Verlauf der Gitterteilung benötigt werden.

3.1.3 Beschreibung der Gitterobjekte

Die im Folgenden vorgestellte Gitterstruktur dient der Diskretisierung von Teilmengen G des R^d ($d = 1, 2, 3$). Wir setzen dabei G als offene und beschränkte Punktmenge (Gebiet) voraus, deren Rand aus einer endlichen Zahl zusammenhängender, glatter Teilstücke besteht.

Geometrische Beschreibung

Als Gitterobjekte werden Punkte des R^d sowie s-dimensionale ($1 \leq s \leq d$), stetig differenzierbare Mannigfaltigkeiten S des R^d betrachtet,

- die zusammenhängend sind,
- die in der induzierten Topologie des R^s offen sind und
- deren Oberflächen aus endlich vielen, $(s-1)$-dimensionalen Gitterobjekten bestehen.

Zur Mannigfaltigkeit S existiert eine definierende *Kartenabbildung*

$$\kappa_S : S \to T, \quad S \subset R^d\,,\; T \subset R^s \text{offen}\,.$$

Die Abbildung κ_S ist eine Diffeomorphismus, d. h. sie ist zusammen mit ihrer Umkehrabbildung stetig differenzierbar. Mit Hilfe dieser Abbildung können geometrische Fragestellungen auf S, wie sie im Verlauf der Gitterteilung auftreten, äquivalent in der Parametermenge T untersucht werden.

Darstellung eines Gebietes

Bei der Darstellung eines Gebietes durch ein Gitter werden die Zerlegungen bzw. die geometrischen Objekte derart gewählt, dass je zwei Objekte des Gitters entweder gleich oder disjunkt sind. Da die Gitterobjekte als offene Mengen definiert sind, sind auch die Schnitte mit benachbarten oder berandenden Objekten leer. Mit $\mathcal{O}_s$ bezeichnen wir die Menge aller geometrischen Objekte der Dimension s ($s = 0, ..., 3$), mit $\mathcal{O} := \cup_s \mathcal{O}_s$ die Menge aller Objekte des Gitters.

War das Ausgangsgebiet G von endlich vielen zusammenhängenden, glatten Mengen berandet, so kann auch das Gitter aus endlich vielen Objekten aufgebaut werden.

3.1.4 Beschreibung der Konnektivität

Orientierung. Für eine Kante wird durch die Markierung des Anfangs- (-1) und des Endknotens ($+1$) eine *Kantenrichtung* eindeutig festgelegt. Für eine Facette werden alle Kanten so markiert (± 1), dass ihre Richtungen kompatibel sind (*Umlaufsinn*). In diesem Fall wird jeder Knoten der Facette genau zweimal referenziert, einmal als Anfangs- und einmal als Endpunkt. Der Umlaufsinn einer Facette definiert gleichzeitig eine gerichtete Oberflächennormale, z. B. durch Anwendung der "Rechten-Hand-Regel" für den Zusammenhang zwischen der Richtung des elektrischen Stromes in einem Leiter (entsprechend der Normalenrichtung der Facette) und der Richtung des ihn umgebenden Magnetfeldes (entsprechend dem Umlaufsinn). Für eine Zelle werden die begrenzenden Facetten so markiert (± 1), dass die Oberflächennormalen aus der Zelle hinaus zeigen.

Die Orientierung ist keine Eigenschaft eines Objektes, sondern eine Eigenschaft der Relation zwischen zwei Objekten. Beispielsweise wird eine Facette von zwei gegenüberliegenden Zellen mit entgegengesetzter Orientierung referenziert.

Definition 3.1 (Konnektivität und Orientierung) *Unter der* Abwärtskonnektivität *soll die Relation*

$$\mathcal{S}_s \subset \mathcal{O}_s \times \mathcal{O}_{s-1} \quad (1 \leq s \leq d)$$

verstanden werden, welche die "besteht-aus"-Relation zwischen Objekten der Dimension s und $s-1$ beschreibt. Für jedes Paar in $\mathcal{S} := \bigcup_s \mathcal{S}_s$ ist genau eine Orientierung

$$\mathcal{R} : \mathcal{S} \to \{\pm 1\}$$

festgelegt.

Definition 3.2 (Objekt-Konnektivität und Oberflächenobjekte) *Als* Abwärtskonnektivität eines Objektes $o \in \mathcal{O}_s$ *wird die Menge seiner Oberflächenobjekte zusammen mit ihren Orientierungen*

$$\mathcal{S}(o) := \{s = (p, r) \mid (o, p) \in \mathcal{S}, r = \mathcal{R}(o, p)\}$$

bezeichnet, die Menge der Oberflächenobjekte des Objektes o *mit*

$$S(o) = S^1(o) := \{p \mid (o, p) \in \mathcal{S}\} \subset \mathcal{O}_{s-1}$$

und deren Oberflächenobjekte mit

$$S^2(o) = S(S^1(o)) := \bigcup_{p \in S^1(o)} S(p) \subset \mathcal{O}_{s-2} \quad \textit{und} \tag{3.1}$$

$$S^3(o) = S(S^2(o)) := \bigcup_{q \in S^2(o)} S(q) \subset \mathcal{O}_{s-3} \tag{3.2}$$

Die Oberflächenobjekte der Oberflächenobjekte werden abkürzend als "Ober-Oberflächenobjekte" bezeichnet.

Definition 3.3 (Nachbarschaft) *Zwei Objekte o_1 und o_2 heißen* benachbart *bzw. genauer* benachbart bezüglich p, *wenn p in der Abwärtskonnektivität beider Objekte enthalten ist:*

$$p \in \mathcal{S}(o_1) \cap \mathcal{S}(o_2) \neq \emptyset$$

Ein Objekt o der Dimension s wird eindeutig beschrieben durch seine Kartenabbildung κ_o und seine Abwärtskonnektivität $\mathcal{S}(o)$:

$$o = (\kappa_o, \mathcal{S}(o)) \in \mathcal{O}_s$$

Die Definition 3.1 der Konnektivität impliziert, dass Oberflächenobjekte nicht zweimal in der Abwärtskonnektivität eines Objektes vorkommen können, auch nicht mit entgegengesetzten Orientierungen. Dadurch sind *innere Oberflächenobjekte* ausgeschlossen, d. h. Oberflächenobjekte, die das Innere des beschriebenen Objektes auf beiden Seiten haben. Ferner kann ein Objekt nicht nur an seinem Rand zusammenhängen, da es sonst nicht offen und zusammenhängend ist.

3.1.5 Beispiele für Maschen

Neben "konventionellen" Maschenformen wie Dreiecken, Vierecken, Tetraedern, Prismen und Hexaedern erlaubt die vorgestellte Gitterrepräsentation komplette Rechengebiete als eine einzige Masche darzustellen. So zeigt Abbildung 3.2 ein NLR-7301 Tragflügelprofil mit Klappe (2D) und eine Flugzeug-Konfiguration mit Tragflügel, Triebwerksaufhängung und Triebwerksgondel (3D). In der linken unteren Ecke ist jeweils die Ansicht des gesamten Rechengebietes dargestellt. Das Gitter besteht in beiden Fällen aus einem Kontrollvolumen mit vielen Oberflächen. In beiden Fällen ist das Kontrollvolumen nicht konvex und nicht einfach zusammenhängend.

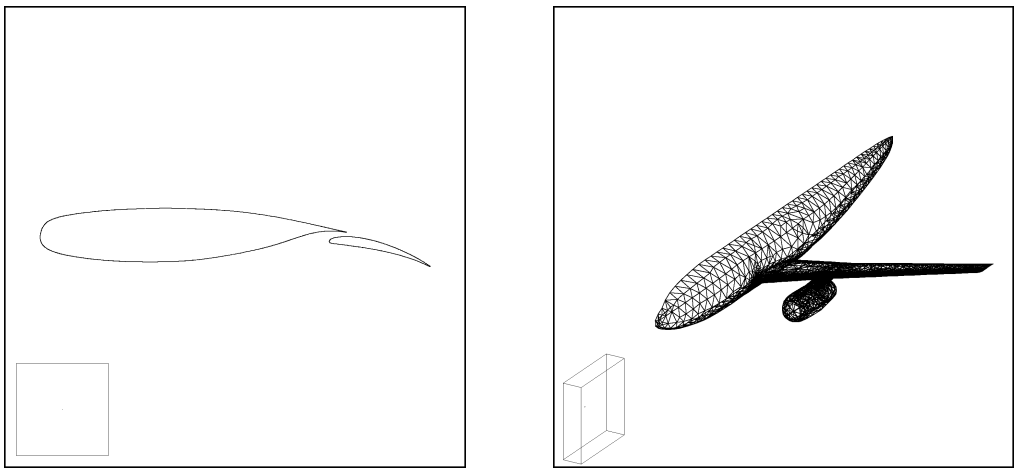


Abbildung 3.2: Beispiele für komplexe Maschenformen

3.2 Gitterverfeinerung durch Schnittflächen

3.2.1 Das Konzept des Teilungsprozesses

Verfeinerungsalgorithmen, die bei unstrukturierten Tetraeder- oder Hexaedergittern Anwendung finden, teilen die Kanten des Ausgangsobjektes und verbinden die neu entstandenen Knoten in geeigneter Weise, z. B. mit dem Schwerpunkt der Ausgangsmasche, so

dass wieder Maschen des verlangten Typs entstehen, siehe z. B. [8]. Nach Möglichkeit werden dabei alle Kanten des Ausgangsobjektes halbiert, um die Form des Ausgangsobjektes möglichst gut zu erhalten. Bei großer Anzahl von Kanten führt dieses Vorgehen aber zu unerwünschten Maschengeometrien, den von Vankeirsbilck [61] beschriebenen "Spinnen". Darüber hinaus ist die Konvexität der Ausgangsmasche eine notwendige Bedingung dafür, dass keine unkontrollierten Überschneidungen mit der Oberfläche der Masche entstehen.

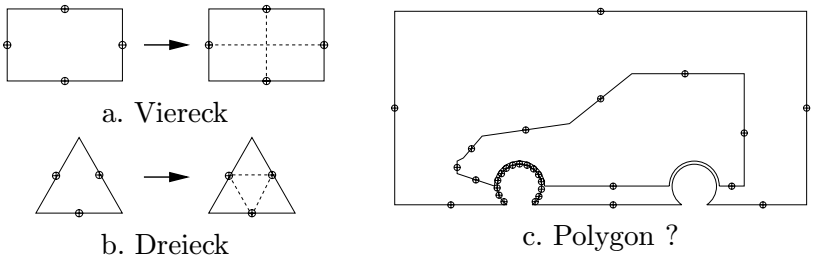


Abbildung 3.3: Verfeinerungsstrategien

Da für die hier verwendeten Gitterobjekte weder die Konvexität garantiert ist noch von einer geringen Anzahl von Kanten ausgegangen werden soll, muss ein anderer Weg zur Teilung von Gitterobjekten beschritten werden.

Statt alle Kanten des Ausgangsobjektes zu halbieren, wird die Teilung entlang einer Schnittfläche durchgeführt, so dass das Ausgangsobjekt in der Regel in zwei Teile zerfällt. Dazu werden alle Gitterobjekte zerlegt, die die Schnittfläche schneiden, und eine Darstellung der Schnittfläche durch Oberflächenobjekte erzeugt. Die Oberflächenobjekte auf jeder Seite der Schnittfläche bilden dann zusammen mit den Schnittflächenobjekten die neuen Top-Level-Objekte (siehe Abbildung 3.4).

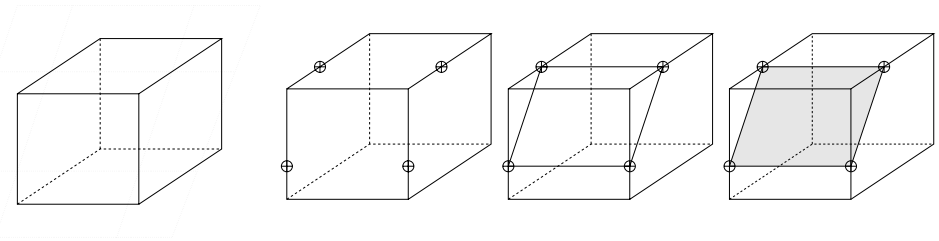


Abbildung 3.4: Konzept des Teilungsprozesses

3.2.2 Schnittfläche und Halbräume

Definition 3.4 (Schnittfläche) *Als Schnittflächen betrachten wir $(d-1)$-dimensionale Mannigfaltigkeiten des R^d, welche eine Bijektion $\bar{\kappa}_H$ des R^d auf sich selbst induzieren:*

$$\bar{\kappa}_H : R^d \to R^d,\ \vec{x} \mapsto \bar{\kappa}_H(\vec{x}) = \begin{pmatrix} \kappa_H \\ \delta_H \end{pmatrix}(\vec{x})$$

Dabei bezeichnet κ_H die Kartenabbildung der Schnittfläche in den R^{d-1} und δ_H die Komponente in Normalenrichtung. Dies entspricht einer Koordinatentransformation, die die Mannigfaltigkeit in die $d-1$-dimensionale Grundebene projeziert, während δ_H die Koordinate senkrecht dazu beschreibt. Gleichzeitig erlaubt die Abbildung κ_H die Projektion

geometrischer Objekte auf die Schnittfläche, sodass die Lage der Objekte zueinander anhand ihrer Projektionen bestimmt werden kann, während die Abbildung δ_H den Abstand eines Punktes von der Schnittfläche misst und die Zuordnung eines geometrischen Objektes zu den induzierten Halbräumen erlaubt.

Definition 3.5 (Schnittfläche und Halbräume) *Die Schnittfläche H unterteilt den Raum R^d in drei Mengen, die nach dem Wert der Funktion δ_H unterschieden werden können: den positiven Halbraum, den negativen Halbraum und die Schnittfläche selbst.*

$$\begin{array}{lrcl} \text{Schnittfläche} & \bar{H} & := & \{\vec{x} \in R^d \mid \delta_H(\vec{x}) = 0\} \\ \text{positiver Halbraum} & H^+ & := & \{\vec{x} \in R^d \mid \delta_H(\vec{x}) > 0\} \\ \text{negativer Halbraum} & H^- & := & \{\vec{x} \in R^d \mid \delta_H(\vec{x}) < 0\} \end{array} \tag{3.3}$$

Als die positive Seite $\bar{H}^+$ *soll der positive Halbraum H^+ zusammen mit der Schnittfläche $\bar{H}$, d. h. $\bar{H}^+ := \bar{H} \cup H^+$, bezeichnet werden. Analog wird die Menge $\bar{H}^- := \bar{H} \cup H^-$ als die* negative Seite *bezeichnet.*

3.2.3 Abstrakte Beschreibung des Vorgehens

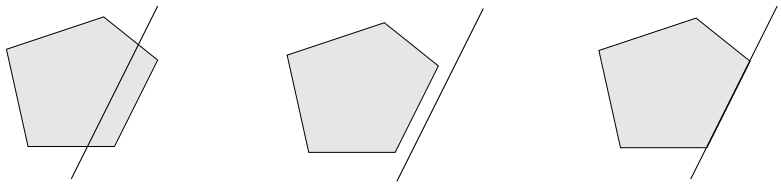
Die Teilung eines geometrischen Objektes o entlang einer Schnittfläche $\bar{H}$ wird in fünf Schritten durchgeführt. Sie verfolgt im Wesentlichen das Ziel, neue Gitterobjekte einzuführen, die die Menge $\bar{o} \cap \bar{H}$ beschreiben, und die daraus folgenden Verletzungen der Schnittfreiheit aller Gitterobjekte aufzulösen.

Schritt 1: Bestimmen der Lage des Objektes

Das betrachtete Objekt muss nur dann einer Teilung unterworfen werden, wenn es die Schnittfläche kreuzt, d. h.:

$$o \cap H^+ \neq \emptyset \quad \text{und} \quad o \cap H^- \neq \emptyset \tag{3.4}$$

Die folgenden Schritte 2–6 werden nur für Objekte unternommen, die die Schnittfläche kreuzen.



a. Objekt kreuzt b. $\partial o \cap \bar{H} = \emptyset$ c. $\partial o \cap \bar{H} \neq \emptyset$

Abbildung 3.5: Lage eines Objektes

Schritt 2: Behandlung der Oberflächenobjekte

Oberflächenobjekte, die sich vollständig im positiven (bzw. negativen) Halbraum befinden, beranden die noch zu bildenden Teilobjekte der positiven (bzw. negativen) Seite. Oberflächenobjekte, die sich in der Schnittfläche befinden, werden der Seite zugeschlagen, auf der sich das Innere des betrachteten Objektes befindet, vgl. Abbildung 3.5.c.

Schritt 3: Teilung der Oberflächenobjekte

Alle anderen Oberflächenobjekte $p \in S(o)$ werden geteilt. Dadurch werden diejenigen Oberflächenobjekte im betrachteten Objekt und seinen Nachbarn ersetzt, die von der Schnittfläche echt geschnitten werden. Nach diesem Schritt befinden sich also alle Oberflächenobjekte entweder im positiven Halbraum, im negativen Halbraum oder in der Schnittfläche.

$$\forall p \in S(o) : \begin{cases} \text{entweder:} & p \subset H^+ \\ \text{oder:} & p \subset \bar{H} \\ \text{oder:} & p \subset H^- \end{cases} \tag{3.5}$$

Schritt 4: Erzeugung der Trennobjekte

Für den Schnitt zwischen dem betrachteten Objekt und der Schnittfläche

$$\delta_H(\kappa_o) = 0 \tag{3.6}$$

werden neue Oberflächenobjekte ("Trennobjekte ") erzeugt. Diese Trennobjekte werden von den (Ober-Oberflächen-) Objekten berandet, die in der Schnittfläche liegen und denen, die bei der Teilung der Oberflächenobjekte als Trennobjekte erzeugt wurden, vgl. Abbildung 3.4. Das heißt, dass das Schnittproblem auf der Parametermenge gelöst werden muss. Zerfällt der Schnitt in mehrere, unzusammenhängende Teile, so wird für jede Zusammenhangskomponente ein Trennobjekt erzeugt.

Schritt 5: Erzeugung der Teilobjekte

Die Oberflächenobjekte der positiven Seite und die Trennobjekte bilden eine geschlossene, zweiseitige Oberfläche, die ein Gebiet auf der positiven Seite der Schnittfläche beranden. In gleicher Weise beschreiben die Oberflächenobjekte der negativen Seite zusammen mit den Trennobjekten ein Gebiet auf der negativen Seite, jedoch gehen die Trennobjekte mit umgekehrter Orientierung ein. Für jede Zusammenhangskomponente dieser beiden Gebiete wird ein Teilobjekt erzeugt.

Schritt 6: Ersetzen des Originalobjektes

Der letzte Schritt besteht darin, das Originalobjekt in der Konnektivitätsrelation zu Superobjekten durch die neuen Teilobjekte zu ersetzen und das Originalobjekt aus dem Gitter zu entfernen. Dann befindet sich das Gitter wieder in einem konsistenten Zustand.

3.3 Implementierungsaspekte

Die obige Darstellung weist im Hinblick auf eine Implementierung einige ungünstige Eigenschaften auf:

- für die Kartenabbildungen der geometrischen Objekte ist eine explizite Darstellung erforderlich, damit die Schnittgleichungen gelöst werden können (Glg. 3.6).

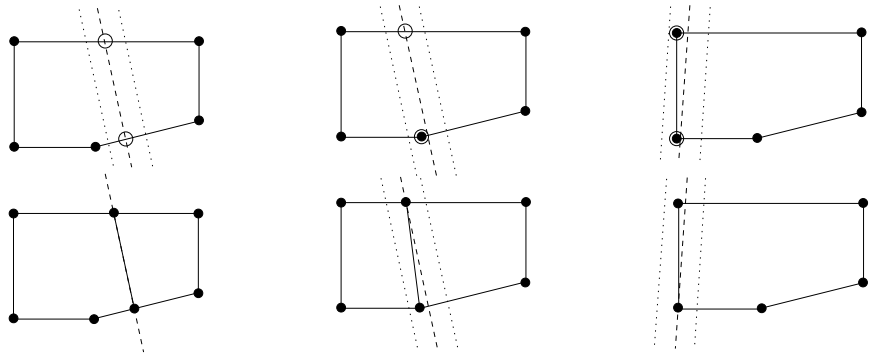
- es sind keinerlei Vorkehrungen getroffen, die die unerwünschte Erzeugung sehr kleiner Objekte verhindern.
- das Konzept der "breiten Schnittfläche" (vgl. Abschnitt 3.3.1), welches die Erzeugung kleiner Kanten unterdrückt, bringt eine für die Parallelisierung hinderliche Reihenfolgeabhängigkeit in den Algorithmus ein.

3.3.1 Breite Schnittflächen

Teilt man ein gegebenes Objekt entlang einer *beliebigen* Schnittfläche, erhebt sich die Frage, wie die unerwünschte Erzeugung sehr kleiner Objekte vermieden werden kann. Verliefe eine gegebene Schnittfläche in der Nähe der Oberflächenobjekte, so würden neue Objekte entstehen, wovon eines von verschwindener Größe ist, aber den vollen Speicherplatz und den vollen numerischen Aufwand benötigt.

Da Informationen über die Gestalt der Objekte nur in Form von Abbildungen vorhanden sind, läßt sich die Wirkung einer Schnittflächenwahl auf das Ergebnis der Teilung nicht vorhersehen. Durch die rekursive Struktur des Teilungsprozesses stehen solche Informationen erst zur Verfügung, wenn die Teilung der Oberflächenobjekte bereits abgeschlossen ist. Auch dann enthält der Algorithmus keine explizite "Vorstellung" von der Gestalt des Ausgangsobjektes, lediglich eine Klassifikation kann ermittelt werden, weil alle Teilungsinformationen auf die Lage im Bezug auf die Halbräume beschränkt sind.

Um die Wahl einer geeigneten Schnittfläche zu ermöglichen, wäre es notwendig, "Testschnitte" durchzuführen, die im Falle ungünstiger Ergebnisse zurückgenommen werden müssten. Eine Modifikation der Schnittfläche wird allerdings durch die mögliche Komplexität der Geometrie erschwert. So kann eine Verbesserung an einer Stelle eine Verschlechterung an anderen Stellen bewirken. Da die Anzahl der Freiheitsgrade zur Festlegung einer Schnittfläche (sie entspricht der *eines* Gitterobjektes der Dimension $d-1$) systematisch niedriger ist als die Anzahl der Freiheitsgrade in der Objektgeometrie, lässt sich nicht sicherstellen, dass auf diesem Wege eine "geeignete" Schnittfläche ermittelt werden kann.



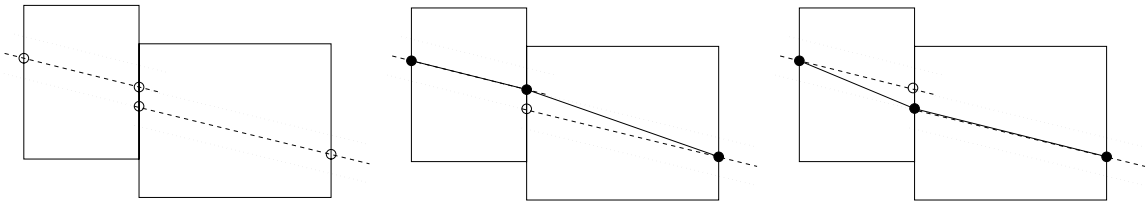
a. normaler Schnitt b. Schnitt fängt Knoten c. Schnitt herausgebogen

Abbildung 3.6: Breite Schnittfläche

Statt die Schnittfläche an das Objekt anzupassen, wird deshalb der Ansatz verfolgt, nur diejenigen Objekte zu teilen, die die Schnittfläche "wesentlich" schneiden. Objekte, die nahezu auf der Schnittfläche liegen, werden behandelt, als ob sie exakt darauf lägen. Diese Strategie vermittelt den Eindruck, eine breite oder gekrümmte Schnittfläche zu verwenden, vgl. Abbildung 3.6.

3.3.2 Reihenfolgeabhängigkeit der Gitterverfeinerung

Werden breite Schnittflächen benutzt, führt die Teilung benachbarter Objekte mit verschiedenen Schnittflächen unter Umständen auf andere Ergebnisse, falls die Reihenfolge der Teilungen verändert wird, vgl. Abbildung 3.7. Wird nämlich bei einer ersten Teilung das gemeinsame Oberflächenobjekt geteilt, findet die zweite Teilung eventuell (Ober-Oberflächen-) Objekte vor, an die sich deren Schnittfläche entsprechend anpasst. Wird die Reihenfolge umgekehrt, trifft dieses Argument die Schnittfläche des anderen Objektes.



a. Konfiguration b. zuerst linkes Volumen c. zuerst rechtes Volumen

Abbildung 3.7: Reihenfolgeabhängigkeit

Die Auswirkungen der Reihenfolgeabhängigkeit auf den Verlauf einer selbst-adaptiven Simulation sind erheblich, weil die zur Steuerung des Verfahrens eingesetzten Parameter, z. B. der Verfeinerungsindikator, starke Gitterabhängigkeiten aufweisen. Darüberhinaus wird die verwendete Finite-Volumen-Diskretisierung von der lokalen Geometrie beeinflusst, so dass Simulationen zu deutlich verschiedenen Ergebnissen führen können. Offensichtlich wird dieses Verhalten bei Rechnungen auf symmetrischen Problemen, da hier unsymmetrische Verfeinerungen entstehen können, die zu unsymmetrischen Ergebnissen führen.

Bei der Implementierung der Teilung auf Parallelrechnern entstehen ebenfalls unerwünschte Nebenwirkungen. So muss die Teilung von Objekten an Prozessorgrenzen explizit serialisiert werden und die Ergebnisse einer Simulation hängen von der Anzahl der eingesetzten Prozessoren ab. Insbesondere gibt es keine Möglichkeit zu garantieren, dass eine auf vielen Prozessoren durchgeführte Rechnung exakt die gleichen Ergebnisse liefert wie dieselbe Rechnung auf einer anderen Zahl von Prozessoren.

Durch die Festlegung einer Teilungsreihenfolge für die Kontrollvolumen, die von der Partitionierung unabhängig ist, können diese Schwierigkeiten vermieden werden. Eindeutige Ergebnisse werden erreicht, wenn die an eine Seite angrenzenden Kontrollvolumen immer in der gleichen Reihenfolge geteilt werden. Diese Forderung impliziert eine Teilordnung auf den Kontrollvolumen. Eine solche Ordnung ist beispielsweise durch eine Sortierung nach den Koordinaten der Kontrollvolumen gegeben. Unabhängig von der Partitionierung erhält diese Sortierung die notwendige Teilordnung über die Seiten. Gleichzeitig zeigen numerische Experimente, dass durch diese Sortierung auf symmetrischen Problemen symmetrische Gitterverfeinerungen entstehen.

3.3.3 Kartenabbildungen

Der zuvor beschriebene Teilungsalgorithmus wälzt die mögliche geometrische Komplexität der Gitterobjekte und der Schnittfläche auf die Beherrschung von Kartenabbildungen ab. So wird beispielsweise die Bestimmung von Schnittmengen auf die Lösung von Gleichungssystemen abgebildet.

Für Objekte der Dimension des Problemraumes ($s = d$) ist die Abbildung trivial ($\kappa = \mathrm{id}$), für Knoten ($s = 0$) ist keine Abbildung notwendig. Für Facetten ($3D$), Kanten ($2D$, $3D$) und die Schnittfläche muss jedoch eine explizite Repräsentation für diese Abbildungen gewählt werden, d. h. es werden endlich-dimensionale Teilräume V_s differenzierbarer Abbildungen gesucht, die wenigstens folgende Kriterien erfüllen:

1. Randwertproblem: zu jeder zulässigen Oberfläche existiert eine Abbildung, die ein Objekt mit dieser Oberfläche beschreibt (vgl. Kap. 3.2.3, Schritt 4).
2. Abschluss unter Schnitten: $V_s \cap V_s \subset V_{s-1}$.
3. Die Lösung von Gleichungssystemen ist beherrschbar.
4. Einfache Repräsentation der Parametermengen der Abbildungen, z. B. keine Notwendigkeit, Triangulierungen erzeugen zu müssen.
5. Die Basisfunktionen des Raumes sind zugänglich: eine ökonomische Repräsentation ist wünschenswert, da ihre Wahl das numerische Verfahren nur über geometrische Invarianten beeinflusst.
6. Die für das numerische Verfahren notwendigen Quadraturformeln und geometrischen Invarianten sind ermittelbar. Dazu zählen Kurvenlängen, Flächen- und Rauminhalte und Integrationspunkte.

Eine Lösung, die die meisten Fragen einfach und effizient behandelt, ist die Wahl ebener Polygonzüge als Facetten, gerader Strecken als Kanten und affiner Hyperebenen als Schnittflächen. Die zu diesen Facetten bzw. Kanten gehörigen Abbildungen sind bekannt, das Schnittproblem ist einfach lösbar, die Räume sind von kleiner Dimension, ihre Basen sind bekannt und Quadraturformeln ermittelbar.

Bei der Verwendung breiter Schnittflächen ergibt sich allerdings die Notwendigkeit auch Facetten darstellen zu können, die von nicht-ebenen Polygonzügen berandet sind. Andernfalls müssen diese, z. B. durch Triangulierung, in ihre ebenen Teile zerlegt werden. Dies ist unerwünscht, weil dabei eine hohe Zahl von Oberflächenobjekten und damit zusätzlicher numerischer Aufwand entstehen kann (Flussberechnungen). Darüberhinaus ist die Implementierung entsprechender Algorithmen aufwändig. Alternativ lassen sich gekrümmte Flächen beispielsweise mit Polynomen (NURBS, Splines, Bernstein-Polynome) darstellen. Die Lösung des Randwertproblems (Fläche zu einem gegebenen Polygonzug), die Repräsentation der Parameterräume (Triangulierungen, Quader) und die Ermittlung und Darstellung der Schnitte bereiten aber Schwierigkeiten. Auch ist ihre Implementierung mit hohem algorithmischen und Speicheraufwand verbunden.

Eine andere Option ist die implizite Darstellung der Kartenabbildungen. Dies ist insbesondere deshalb attraktiv, weil das numerische Verfahren von diesen Abbildungen keinen direkten Gebrauch macht.

3.3.4 Implizite Darstellung der Kartenabbildungen

Wird auf eine explizite Darstellung der Abbildung verzichtet, müssen andere Werkzeuge gefunden werden, um geometrische Fragestellungen zu beantworten. Als eine geeignete Strategie hat es sich dabei erwiesen, die gestellten Aufgaben rekursiv auf den Oberflächen zu lösen. Voraussetzung dafür ist allerdings, dass die Oberflächen genügend Information

über das Objekt tragen. Dies wird durch die Wahl geeigneter Einschränkungen erreicht, die die geometrische Komplexität reduzieren.

Da eine explizite Repräsentation des Objektinneren nicht vorliegt, erhält man allerdings auch keine explizite Repräsentation des Schnittes zwischen einer Schnittfläche und dem Objekt, lediglich die Schnittpunkte mit den begrenzenden Polygonzügen stehen zur Verfügung. Verbindet man diese Schnittpunkte durch Strecken, so liegen die neu entstandenen Kanten eventuell nicht in der ursprünglichen Fläche. Damit sind auch die neu entstandenen Teilflächen nicht Teilmengen der Ausgangsfläche. Für das Innere des Rechengebietes stellt diese Tatsache keine Schwierigkeit dar, dagegen sind für die Adaption an der Oberfläche besondere Vorkehrungen notwendig, um die Darstellung gekrümmter Oberflächen zu gewährleisten.

Eine weitere Schwierigkeit ist die Aufrechterhaltung der Schnittfreiheit nahe beieinander liegender gekrümmter Facetten. Da explizite Formeln fehlen, besteht keine Möglichkeit, den Schnitt zu bestimmen bzw. Schnittfreiheit festzustellen.

Für die Kartenabbildungen werden deshalb folgende Einschränkungen festgelegt:

1. die Projektion einer Facette f entlang des mittleren Normalenvektors

$$\vec{s}_o = \int_f 1 \, \mathbf{d}\sigma(\vec{x}) \tag{3.7}$$

   ist injektiv.

2. die Kartenabbildung genügt einem Minimums- und einem Maximumsprinzip:

$$\forall \vec{n} \in R^d, ||\vec{n}|| = 1 : \{\langle \vec{x}, \vec{n} \rangle \mid \vec{x} \in o\} \subset \{\langle \vec{x}, \vec{n} \rangle \mid \vec{x} \in \partial o\} \tag{3.8}$$

Forderung 3.7 bewirkt, dass eine gekrümmte Facette im Wesentlichen wie eine ebene Facette behandelt werden kann. Insbesondere können Anomalien beim Teilungsvorgang durch geeignete Wahl der Schnittdicke verhindert werden. Forderung 3.8 schränkt die möglichen Kartenabbildungen für Kanten auf Strecken ein. Für Facetten wird die Ausdehnung der Fläche in Richtung der Flächennormalen eingeschränkt. Darüberhinaus ist die Verwendung gekrümmter Schnittflächen durch diese Annahmen im Wesentlichen ausgeschlossen.

Die im folgenden Abschnitt dargestellten Überlegungen zeigen, dass mit Hilfe dieser Einschränkungen die Abstandsfunktion δ_H ausgewertet werden kann, ohne dabei auf die Kartenabbildungen zurückgreifen zu müssen. Stattdessen wird die Auswertung unter Zuhilfenahme topologischer Argumente (Färbung) auf die Ränder abgewälzt.

3.4 Gitterverfeinerung unter vereinfachten Bedingungen

Unter Verwendung der in Abschnitt 3.3.4 diskutierten Einschränkungen wird der in Abschnitt 3.2.3 vorgestellte Verfeinerungsalgorithmus neu formuliert. Dabei werden alle Fragestellungen der Teilung eines Objektes, die auf die Kartenabbildungen Bezug nehmen, durch Operationen auf den Oberflächen der Objekte ersetzt, so dass eine explizite Darstellung für die Kartenabbildungen nicht mehr notwendig ist. Die Teilung eines Objektes wird auf die Teilung seiner Oberflächen zurückgeführt, der Algorithmus durchläuft rekursiv die Hierarchie der geometrischen Objekte bis hinunter zu den Knoten, die ihre Lage im Bezug auf die Schnittfläche einfach bestimmen können.

3.4.1 Breite Schnittfläche und Lage von Punktmengen

Als Schnittflächen H betrachten wir affine Hyperebenen des R^d. Die beschreibende Kartenabbildung $\bar{\kappa}_H$ ist durch einen Punkt $\vec{p}_H$, eine Normalenrichtung $\vec{n}_H$ und eine Basis $\{\vec{d}_{H,1}, \vec{d}_{H,2}\}$ für $\vec{n}_H^{\perp}$ festgelegt:

$$\bar{\kappa}_H : R^d \to R^d, \vec{x} \mapsto \bar{\kappa}_H(\vec{x}) = \begin{pmatrix} \kappa_H \\ \delta_H \end{pmatrix}(\vec{x}) = \begin{pmatrix} \vec{d}_{H,1}^{\,t} \\ \vec{d}_{H,2}^{\,t} \\ \vec{n}_H^{\,t} \end{pmatrix} \cdot (\vec{x} - \vec{p}_H)$$

Definition 3.6 (Breite Schnittfläche und Halbräume) *Eine* Schnittfläche H_δ der Breite δ *unterteilt den Raum R^d in drei Mengen, den positiven Halbraum, den negativen Halbraum und die Schnittfläche selbst.*

$$\begin{array}{lrcl} \text{breite Schnittfläche} & \bar{H}_\delta & := & \{\vec{x} \in R^d \,|\, |\delta_H(\vec{x})| \leq \delta\} \\ \text{positiver Halbraum} & H_\delta^+ & := & \{\vec{x} \in R^d \,|\, \delta_H(\vec{x}) > \delta\} \\ \text{negativer Halbraum} & H_\delta^- & := & \{\vec{x} \in R^d \,|\, \delta_H(\vec{x}) < \delta\} \end{array} \tag{3.9}$$

Wie in Definition 3.4 soll unter der *positiven Seite* die Schnittfläche zusammen mit dem positiven Halbraum, $\bar{H}_\delta^+ := \bar{H}_\delta \cup H_\delta^+$, und unter der *negativen Seite* die Schnittfläche zusammen mit dem negativen Halbraum, $\bar{H}_\delta^- := \bar{H}_\delta \cup H_\delta^-$, verstanden werden.

Definition 3.7 (Lagefunktion) *Die Lagefunktion ist eine Abbildung h_δ, die jedem Ortsvektor $\vec{x} \in R^d$ eine Lage im Bezug auf die Schnittfläche zuweist:*

$$h_\delta(\vec{x}) = \begin{cases} +1 & \textit{iff} \quad \delta_H(\vec{x}) & > +\delta \\ 0 & \textit{iff} \quad |\delta_H(\vec{x})| & \leq \delta \\ -1 & \textit{iff} \quad \delta_H(\vec{x}) & < -\delta \end{cases} \tag{3.10}$$

Diese Abbildung lässt sich in natürlicher Weise auf Teilmengen Ω des R^d ausdehnen:

$$h_\delta(\Omega) = \{h_\delta(\vec{x}) \,|\, \vec{x} \in \Omega \subset R^d\}$$

Im Folgenden soll geklärt werden, inwieweit sich aus der Lage des Randes $\partial\Omega$ einer offenen Teilmenge $\Omega \subset R^s$, $(1 \leq s \leq d)$ Aussagen über die Lage von Ω bzgl. der Schnittfläche ableiten lassen. Aufgrund des Minimal- und Maximalprinzips (3.8) lassen sich zu jedem Punkt $\vec{x} \in \Omega$ der offenen Menge zwei Punkte $\vec{x}_m, \vec{x}_M \in \partial\Omega$ aus dem Rand finden, für die gilt:

$$\delta_H(\vec{x}_m) \leq \delta_H(\vec{x}) \leq \delta_H(\vec{x}_M)\,.$$

Enthält also $h_\delta(\Omega)$ höchstens zwei Werte, so werden diese auch auf dem Rand angenommen und es gilt:

$$h_\delta(\Omega) \subset h_\delta(\partial\Omega). \tag{3.11}$$

Enthält $h_\delta(\Omega)$ dagegen drei Werte, d. h. $h_\delta(\Omega) = \{+1, 0, -1\}$, so liegen Punkte der Menge in der breiten Schnittfläche, ohne dass ihr Rand einen nicht-leeren Schnitt mit der breiten Schnittfläche haben müsste, z. B. im Falle nicht zusammenhängender Ränder, und es gilt:

$$h_\delta(\Omega) \subset h_\delta(\partial\Omega) \cup \{0\}. \tag{3.12}$$

Umgekehrt enthält jede offene Kugel um einen Randpunkt $\vec{x} \in \partial\Omega$ nach Definition des Randes einen nicht-leeren Schnitt mit der Menge Ω. Liegt der Randpunkt nicht auf dem Rand der breiten Schnittfläche, gilt also $\delta_H(\vec{x}) \neq \delta$, so liegt die Kugel mit dem Radius $||\delta_H(\vec{x})| - \delta| > 0$ im selben Halbraum wie $\vec{x}$ selbst und es gilt:

$$h_\delta(\partial\Omega) \subset h_\delta(\Omega). \tag{3.13}$$

Besteht der Schnitt des Randes mit der breiten Schnittfläche ausschließlich aus Punkten mit dem Abstand δ, so lässt sich diese Argumentation nicht aufrechterhalten und es gilt:

$$h_\delta(\partial\Omega) \subset h_\delta(\Omega) \cup \{0\}. \tag{3.14}$$

Damit erhält man für den allgemeinen Fall die Beziehung

$$h_\delta(\Omega) \cup \{0\} = h_\delta(\partial\Omega) \cup \{0\}. \tag{3.15}$$

3.4.2 Schritt 1: Bestimmen der Lage des Objektes

Ausgehend von der Definition der Schnittfläche wird nun eine Lagefunktion h_δ konstruiert, welche Auskunft über die Lage eines Objektes im Bezug auf die Schnittfläche $\bar{H}_\delta$ gibt. Die Lage eines Objektes, welches durch Oberflächenobjekte beschrieben ist, bestimmt sich rekursiv aus der Lage dieser Oberflächen. Wie Glg. 3.15 zeigt, trägt der Fall $\{0\} \subset h_\delta(p)$ für die Lage eines Oberflächenobjektes p nur wenig verwertbare Information zur Lage des Objektes bei. Diese Information wird in der Lagefunktion deshalb durch eine Information über Oberflächenobjekte ersetzt, die ganz in der Schnittfläche enthalten sind. Knoten besitzen keine Oberflächenobjekte und ihre Lage bezüglich der Schnittfläche wird durch die Lage ihres Ortsvektors bestimmt.

Definition 3.8 (Lage eines Gitterobjektes) *Unter der* Lage eines Gitterobjektes o *bezüglich der Schnittfläche H_δ soll der Wert der Funktion*

$$\begin{array}{rccl} h_\delta: & \mathcal{O} & \to & \mathcal{P}(\{+1,0,-1\}), \\ & o & \mapsto & \left\{ \begin{array}{ll} \bigcup_{p\in S(o)} \left\{ \begin{array}{ll} h_\delta(p) \cap \{+1,-1\}, & h_\delta(p) \neq \{0\} \\ \{0\}, & h_\delta(p) = \{0\} \end{array} \right\}, & o \notin \mathcal{O}_0 \\ h_\delta(\vec{x}_o), & o \in \mathcal{O}_0 \end{array} \right\} \end{array} \tag{3.16}$$

verstanden werden.

Die folgende Überlegung zeigt, dass ein Objekt o die Schnittfläche H_δ genau dann kreuzt, wenn $\{+1,-1\} \subset h_\delta(o)$ gilt. Da Knoten eine eindeutige Lage in Bezug auf die Schnittfläche haben, diese also nicht kreuzen können, betrachten wir dazu ein Objekt $o \in \mathcal{O}_s$ der Dimension $1 \leq s \leq d$. Dieses Objekt hat eine nicht-leere Menge von Oberflächenobjekten $S(o)$. Glg. 3.15 zeigt, dass außerhalb der Schnittfläche die Lage der Menge $\{\vec{x} \in o\}$ mit der Lage der Menge $\{\vec{x} \in \partial o\}$ übereinstimmt . Dieses Argument lässt sich rekursiv bis auf die Knoten fortsetzen, d. h. es gibt in der Oberfläche des Objektes sowohl Knoten, die im positiven Halbraum liegen, als auch solche, die im negativen Halbraum liegen. Aufgrund der Definition der Objektlage gilt dann $\{+1,-1\} \subset h_\delta(o)$. Umgekehrt lässt sich aufgrund der rekursiven Definition der Objektlage aus $\{+1,-1\} \subset h_\delta(o)$ auf die Existenz von Knoten in beiden Halbräumen schließen. Damit kreuzt auch die Menge $\{\vec{x} \in \partial o\}$ die Schnittfläche und die Behauptung folgt aus Glg. 3.15.

Objekte werden nur dann einer Teilung unterworfen, wenn sie die Schnittfläche kreuzen. Die folgenden Schritte 2–6 werden also nur für kreuzende Objekte durchgeführt.

3.4.3 Schritt 2: Behandlung der Oberflächenobjekte

Die vorhandenen Oberflächenobjekte $p \in S(o)$ müssen so aufbereitet werden, dass sie den neu zu bildenden Teilobjekten auf der positiven bzw. negativen Seite zugeordnet werden können. Im Vergleich zu der Darstellung in Abschnitt 3.2.3 treten durch die positive Breite der Schnittfläche zusätzliche Fragestellungen auf.

Für Oberflächenobjekte p, die zusammen mit ihrem Rand im positiven bzw. negativen Halbraum liegen, stimmt die Lage der Punktmenge mit der Lage des Objektes ($h_\delta(p) = \{+1\}, \{-1\}$) überein. Sie können deshalb der zugehörigen Seite einfach zugeordnet werden. Ihre Ober-Oberflächenobjekte spielen für den Aufbau der Trennobjekte keine Rolle.

Kreuzende Oberflächenobjekte können äquivalent anhand ihrer Lagefunktion identifiziert werden ($\{+1, -1\} \subset h_\delta(p)$) und werden im folgenden Schritt (3) geteilt, vgl. Abschnitt 3.4.4. Die entstehenden Oberflächenobjekte werden dabei den entsprechenden Seiten zugeordnet. Neue Ober-Oberflächenobjekte, die ganz in der breiten Schnittfläche enthalten sind, beranden die noch zu bildenden Trennobjekte.

Trennobjekte sind solche Gitterobjekte, die zur Repräsentation der Menge $o \cap \bar{H}_\delta$ in Schritt (4) neu erzeugt werden, vgl. Abschnitt 3.4.5, und trennen die Teilobjekte der positiven und negativen Seite voneinander. Sie beranden sowohl die Teilobjekte der positiven als auch der negativen Seite und werden deshalb beiden Seiten zugeordnet. Trennobjekte werden von Ober-Oberflächenobjekten (und deren Oberflächen) berandet, die ganz in der Schnittfläche enthalten sind.

Dies führt zu einer Interpretation der Lage $h_\delta(p) = \{+1, 0\}, \{-1, 0\}$ eines Objektes, die sich wesentlich von der Lage der zugehörigen Punktmenge unterscheidet. Ein Oberflächenobjekt berührt die Schnittfläche in diesem Sinne, wenn eines seiner Ober-Oberflächenobjekte ($S^2(o)$, $S^3(o)$) ganz in der Schnittfläche liegt und damit in die Berandung eines Trennobjektes eingeht. Bei der Zellteilung (3D) kann die Erzeugung geschlossener Kantenzüge zur Berandung von Trennfacetten zur Teilung einer Facette der Lage $\{+1, 0\}$ oder $\{-1, 0\}$ führen, vgl. Abschnitt 3.4.5. In allen anderen Fällen können solche Oberflächenobjekte der entsprechenden Seite zugeordnet werden.

Für Oberflächenobjekte p, die vollständig in der breiten Schnittfläche enthalten sind, stimmt der Wert der Lagefunktion $h_\delta(p)$ mit der Lage der Punktmenge $h_\delta(\{\vec{x} \in p\})$ überein, weil die Lagefunktion nur einen Wert hat und die breite Schnittfläche eine abgeschlossene Menge ist (vgl. auch Abschnitt 3.4.1). Diese Oberflächenobjekte trennen *nicht* die neu zu bildenden Teilobjekte voneinander, sondern das Innere vom Äußeren des Ausgangsobjektes. Sie sind also keine Trennobjekte und müssen deshalb auf einem anderen Weg einem der beiden Teilobjekte zugeordnet werden. Dazu kommt das in Abschnitt 3.4.8 beschriebene Vorgehen zum Einsatz, das auf der Betrachtung von Zusammenhangskomponenten dieser Objekte beruht. Dabei werden auch diejenigen Ober-Oberflächenobjekte identifiziert, die in die Berandung der Trennobjekte eingehen.

3.4.4 Schritt 3: Teilung der Oberflächenobjekte

Alle Oberflächenobjekte, die die Schnittfläche kreuzen, werden geteilt und dabei durch neue Oberflächenobjekte ersetzt, die der positiven bzw. negativen Seite zugeordnet werden können. Neu entstehende Ober-Oberflächenobjekte $q \in S^2(o)$, die vollständig in der

Schnittfläche enthalten sind ($h_\delta(q) = \{0\}$), werden in die Berandung der Trennobjekte aufgenommen.

Nach diesem Schritt existieren keine Oberflächenobjekte ($S^1(o)$, $S^2(o)$, $S^3(o)$) mehr, welche die Schnittfläche kreuzen. Alle Oberflächenobjekte sind entweder in der positiven Seite, in der negativen Seite oder in der breiten Schnittfläche *vollständig* enthalten (vgl. Glg. 3.5). Außerdem entstehen dabei auch die Ober-Oberflächenobjekte, die den Schnitt des Objektinneren mit der Schnittfläche $o \cap \bar{H}_\delta$ beranden.

Dieser Schritt entfällt für Kanten, weil deren Oberflächenobjekte, die Knoten, eine eindeutige Lage in Bezug auf die Schnittfläche haben und diese folglich nicht kreuzen können.

3.4.5 Schritt 4: Erzeugen der Trennobjekte

Trennobjekte sind diejenigen Oberflächenobjekte, die die neu zu bildenden Teilobjekte voneinander trennen und den Schnitt des Objektinneren mit der Schnittfläche $o \cap \bar{H}_\delta$ repräsentieren.

Da Knoten eine eindeutige Lage in Bezug auf die Schnittfläche haben, also die Schnittfläche nicht kreuzen, wird dieser Schritt nur für Kanten, Facetten und Zellen durchgeführt. Aufgrund des Extremalprinzips (Unglg. 3.8) werden Kanten durch Strecken repräsentiert, d. h. es ist eine explizite Darstellung der Kartenabbildung verfügbar. Deshalb können die gesuchten Trennknoten durch Lösung der Schnittgleichung 3.6 bestimmt werden. Dabei ist die Länge der entstehenden Teilkanten durch die Schnittdicke gegen Null beschränkt.

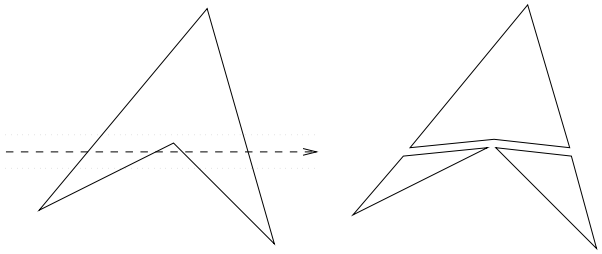
Der Schnitt einer kreuzenden Facette oder Zelle mit der Schnittfläche wird durch neue Trennobjekte (Kanten bzw. Facetten) repräsentiert. Nach Teilung kreuzender Oberflächenobjekte (Schritt 3) wird das Objekt entlang dieser Trennobjekte in eine Teilmenge auf der positiven und eine Teilmenge auf der negativen Seite zerlegt. Da für das Innere von Facetten nur eine implizite Beschreibung in Form von Einschränkungen vorliegt, kann der Schnitt einer Facette oder Zelle mit der Schnittfläche nicht explizit bestimmt werden. Lediglich die Schnittpunkte mit den begrenzenden Kanten stehen zur Verfügung. Neue Trennobjekte beschreiben deshalb den Schnitt nicht exakt. Insbesondere wird bei ihrer Erzeugung der Freiheitsgrad ausgenutzt, der durch die Breite der Schnittfläche gegeben ist.

Trennkanten (bzw. -facetten) werden nach der Teilung kreuzender Kanten (bzw. Facetten) aus Knoten (bzw. Kanten und Knoten) der Lage $h_\delta(q) = \{0\}$ aufgebaut. Für die Berandung von Trennkanten bei der Facettenteilung (bzw. Trennfacetten bei der Zellteilung) werden nur solche Knoten (bzw. Kanten) der Lage $\{0\}$ berücksichtigt, die an Kanten (bzw. Facetten) der positiven oder negativen Seite angrenzen. Unberücksichtigt bleiben also solche Knoten (bzw. Kanten), die bei der Zuordnung von Oberflächen der Lage $\{0\}$ in Schritt (2) bereits einer Seite zugeschlagen wurden, vgl. dazu Abschnitt 3.4.8.

Aus den so bestimmten Schnittknoten (bzw. Schnittkanten) werden Trennkanten (bzw. Trennfacetten) aufgebaut. Dabei muss garantiert werden, dass neu erzeugte Trennobjekte nicht mit bereits vorhandenen Objekten kollidieren. Dies kann mit Hilfe der Projektion κ_H der Objekte auf die Schnittfläche sichergestellt werden.

Entstehen bei diesem Vorgang zwei oder mehr Trennobjekte, so kann das zu teilende Objekt in mehrere, unzusammenhängende Teilmengen zerfallen (vgl. Abbildung 3.8). In diesem Fall wird für jede Zusammenhangskomponente ein eigenes Teilobjekt erzeugt. Dabei

erfordert die korrekte Zuordnung innerer Ränder zu den Zusammenhangskomponenten besondere Beachtung.



a. Konfiguration b. nach Teilung

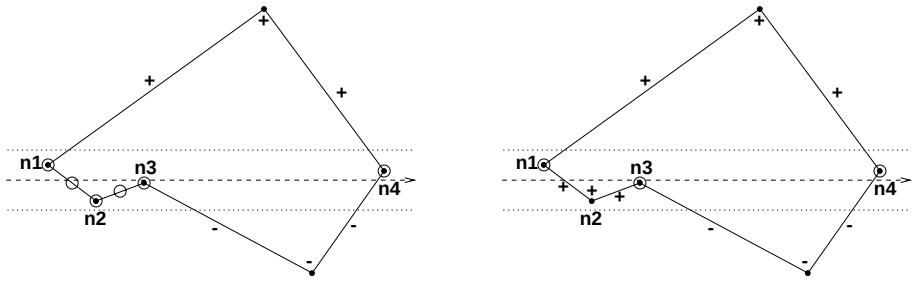
Abbildung 3.8: Unzusammenhängende Teilmengen

Trennkanten für Facetten

Der Schnitt zwischen dem Inneren der Facette und der breiten Schnittfläche (*"Schnittkurve"*) wird durch neue Kanten (*"Trennkanten"*) repräsentiert, die aus Schnittknoten aufgebaut werden. Schnittknoten sind solche Knoten der Lage $\{0\}$, die an Kanten der positiven oder negativen Seite, d. h. an Kanten der Lage $\{+1, 0\}$ oder $\{-1, 0\}$, angrenzen. Diese Knoten repräsentieren Schnittpunkte des Randes der Facette mit der Schnittkurve, d. h. in diesen Knoten kreuzt oder berührt die Schnittkurve den Rand der Facette.

Nun werden die Schnittknoten entsprechend ihrer Projektion auf die Schnittkurve sortiert. In jedem Schnittknoten kann anhand der Anzahl angrenzender Kanten, die der positiven bzw. negativen Seite zugeordnet sind ermittelt werden, ob die Trennkurve den Rand der Facette kreuzt oder berührt. Diese Überlegung beruht auf der Tatsache, dass ein Knoten eine gerade Anzahl von Kanten (einer Facette) berandet, weil es keine inneren Kanten gibt (vgl. Definition 3.1). Begrenzt ein Knoten also eine gerade Anzahl von Kanten der positiven Seite, so grenzt auch eine gerade Anzahl von Kanten der negativen Seite an, und die Schnittkurve berührt den Rand. Verlief die Schnittkurve zuvor innerhalb der Facette, so verläuft sie auch nach diesem Knoten innerhalb und eine weitere, neue Trennkante muss erzeugt werden. Verlief die Schnittkurve dagegen zuvor außerhalb der Facette, so verläuft sie auch nach diesem Knoten außerhalb, so dass keine Trennkante erzeugt werden darf. Begrenzt ein Knoten dagegen eine ungerade Anzahl von Kanten der positiven als auch der negativen Seite, so kreuzt die Schnittkurve den Rand. In diesem Fall kreuzt die Trennkurve den Rand der Facette, d. h. sie wechselt vom Inneren ins Äußere der Facette oder umgekehrt.

An Hand der in Abbildung 3.9 dargestellten Facette soll das oben beschriebene Vorgehen verdeutlicht werden. Abbildung 3.9.a zeigt eine Facette und eine breite Schnittebene nach Auswertung der Lagefunktion (Schritt 1). Dabei sind Objekte des Typs $\{+1\}$ und $\{+1, 0\}$ mit "+", Objekte des Typs $\{-1\}$ und $\{-1, 0\}$ mit "−" und Objekte des Typs $\{0\}$ mit einem Kreis gekennzeichnet. Abbildung 3.9.b zeigt das Ergebnis der Behandlung der Kanten der Lage $\{0\}$ in Schritt (2), vgl. Abschnitt 3.4.8. Beide Kanten und der Knoten n_2 werden der positiven Seite zugeschlagen. Da die dargestellte Facette keine kreuzenden Kanten hat, wird sie in Schritt (3) nicht mehr verändert. Tabelle 3.9 verdeutlicht die Auswahl und Sortierung der Schnittknoten entsprechend ihrer Projektion auf die Schnittfläche. Die Spalten eins bis drei beschreiben den betrachteten Ausschnitt der Schnittkurve und ihren



a. nach Schritt (1) b. nach Schritt (2)

von	Verlauf	bis	$\tilde{H}_\delta^+$	$\tilde{H}_\delta^-$	danach
$-\infty$	außerhalb	n_1	#2	#0	außerhalb
n_1	außerhalb	n_3	#1	#1	innerhalb
n_3	innerhalb	n_4	#1	#1	außerhalb
n_4	außerhalb	∞			

c. Verlauf der Trennkurve

Abbildung 3.9: Erzeugung der Trennkanten

Verlauf innerhalb bzw. ausserhalb der betrachteten Facette. Die vierte und fünfte Spalte zeigt die Anzahl der angrenzenden Kanten der positiven bzw. negativen Seite und Spalte sechs die Folgerung für den weiteren Verlauf, der in die zweite Spalte der nächsten Zeile übertragen wird.

Trennfacetten für Zellen

Der Schnitt zwischen dem Inneren einer Zelle und der breiten Schnittfläche (*"Schnittfläche"*) wird durch neue Facetten (*"Trennfacetten"*) repräsentiert, die aus Schnittkanten aufgebaut werden.

Durch die Anpassung an vorhandene Schnittknoten und -kanten können gekrümmte Facetten entstehen, deren Ausdehnung in Normalenrichtung kleiner gleich der Schnittbreite ist. Bei der Teilung gekrümmter Facetten muss die Schnittbreite größer als die Ausdehnung der Facette gewählt werden. Vor der Teilung einer Zelle wird die Schnittbreite deshalb an die Ausdehnung der berandenen Facetten angepasst.

Die Schnittkanten, die bei der Teilung der verschiedenen, kreuzenden Facetten der Zelle erzeugt werden, beschreiben (eventuell mehrere) geschlossene Kantenzüge, weil die Lage $\{0\}$ der begrenzenden Knoten unabhängig von der Facette ist, in dessen Oberfläche der Knoten liegt. Deshalb berühren sich die Schnittkanten benachbarter Facetten in diesen Schnittknoten, und die Schnittkantenzüge sind geschlossen. Facetten, die die Schnittfläche nicht kreuzen, können ebenfalls isolierte Schnittknoten oder nicht-geschlossene Schnittkanten(-züge) enthalten, die in die Berandung der Trennfacette aufgenommen werden müssen. Diese Schnittknoten bzw. -kanten müssen durch Einfügen zusätzlicher Schnittkanten zu geschlossenen Kantenzügen vervollständigt werden. Das Einfügen zusätzlicher Schnittkanten kann nachträglich eine Teilung von Facetten der Lage $\{+1, 0\}$ oder $\{-1, 0\}$ bewirken, wenn dabei Kanten der gleichen Facette verbunden werden, vgl. Abschnitt 3.4.3.

Um zulässige Trennfacetten zu erhalten, muss die von den Schnittkanten umschriebene Fläche in ihre Zusammenhangskomponenten zerlegt werden. Dabei muss wiederum die korrekte Zuordnung innerer Ränder zu den enthaltenden Zusammenhangskomponenten gewährleistet werden.

3.4.6 Schritt 5: Erzeugung der Teilobjekte

Nachdem alle Oberflächen- und Trennobjekte den zugehörigen Seiten zugeordnet sind, kann das zu teilende Objekt in zwei Teilmengen zerlegt werden, welche sich auf der positiven bzw. negativen Seite befinden. Diese Teilmengen werden von den Oberflächenobjekten der positiven (bzw. negativen) Seite und den Trennobjekten berandet und müssen noch in ihre Zusammenhangskomponenten zerlegt werden, um als Gitterobjekte repräsentiert werden zu können.

Die bestehenden Oberflächenobjekte gehen in die Teilobjekte mit den gleichen Orientierungen ein, die sie bereits zu dem zu teilenden Objekt hatten. Um die neu erzeugten Trennobjekte in die Oberflächenbeschreibung der Teilobjekte aufnehmen zu können, müssen diese noch mit einer Orientierung versehen werden.

Der Trennknoten p^0 einer kreuzenden Kante

$$o = \{(p^-, -r), (p^+, r)\}$$

($r = \mathcal{R}(o, p^+) = -\mathcal{R}(o, p^-)$) wird bzgl. der Teilkanten o^- der negativen Seite und o^+ der positiven Kante so orientiert, dass die Knoten der Ausgangskante in den Teilkanten ihre Orientierungen behalten können. Damit ist die Orientierung des Trennknotens in der folgenden Weise eindeutig festgelegt:

$$o^- = \left\{(p^-, -r), (p^0, r)\right\} \tag{3.17}$$

$$o^+ = \left\{(p^0, -r), (p^+, r)\right\} \tag{3.18}$$

Behalten die bestehenden Kanten (bzw. Facetten) einer kreuzenden Facette (bzw. kreuzenden Zelle) ihre Orientierungen auch in den Teilobjekten bei, so können die Trennkanten (bzw. Trennfacetten) so erzeugt werden, dass sie den Teilobjekten der positiven Seite mit positiver Orientierung beigefügt werden können.

3.4.7 Schritt 6: Ersetzen des Originalobjektes

Der letzte Schritt der Teilung besteht darin, das Originalobjekt in der Konnektivität zu allen Objekten ("Superobjekten") zu ersetzen, in deren Oberfläche das Originalobjekt enthalten war, und das Originalobjekt aus dem Gitter zu entfernen. Das Originalobjekt wird in allen Relationen zu Superobjekten durch die neu erzeugten Teilobjekte ersetzt. Aufgrund der vorherigen Festlegung der Orientierungen erben o^+ und o^- ihre Orientierung bzgl. der Superobjekte.

3.4.8 Zuordnung der Oberflächen der Lage $\{0\}$

Im Vergleich zu dem in Kapitel 3.2.3, Schritt (2) beschriebenen Vorgehen treten hier durch die Wahl einer positiven Schnittdicke komplexe Fragestellungen auf. Aufgrund der positiven Breite der Schnittfläche können mehrere Oberflächenobjekte an der gleichen "Position" (Projektion auf die Schnittfläche) liegen.

Bei einem Objekt, das die Schnittfläche kreuzt, können Oberflächenobjekte der Lage $\{0\}$ nur dann auftreten, wenn das betrachtete Objekt mindestens drei Oberflächenobjekte hat, je eines auf der positiven und negativen Seite (damit das Objekt die Schnittfläche kreuzt) und ein drittes in der Schnittfläche. Die Zuordnung tritt deshalb nur bei der Zell- und Facettenteilung auf, nicht aber bei der Kantenteilung.

Die Zuordnung eines Oberflächenobjektes zu einem Teilobjekt geschieht durch die Veränderung des Wertes der Lagefunktion dieser Oberflächenobjekte, das heißt durch die Zuordnung zur positiven oder negativen Seite. Nach diesem Schritt gibt es keine Oberflächenobjekte des Typs $\{0\}$ mehr. Die Lageveränderung wird nicht in die nachfolgende Teilung kreuzender Oberflächenobjekte (Schritt 3) einbezogen, sondern dient nur der Auswahl von Schnittobjekten bei der Erzeugung der Trennobjekte (Schritt 4).

Zusammenhangskomponenten. Für die Behandlung der Oberflächenobjekte des Typs $\{0\}$ spielen deren Zusammenhangskomponenten eine wesentliche Rolle. Werden nämlich zwei aneinandergrenzende Oberflächenobjekte verschiedenen Seiten zugeordnet, so muss das gemeinsame Ober-Oberflächenobjekt in die Berandung der Trennkurve (bzw. Trennfläche) aufgenommen werden. Das heißt, dass die Schnittfläche den Rand des Objektes dort schneidet. Wenn die Position solcher Übergänge nicht eindeutig bestimmt ist, treten unerwünschte Abhängigkeiten der Trennobjekterzeugung von der Speicherorganisation der Implementierung (z. B. Reihenfolgeabhängigkeiten) auf. Diese Mehrdeutigkeiten können durch die Betrachtung von Zusammenhangskomponenten dieser Oberflächenobjekte weitgehend vermieden werden.

Schnittfreiheit. Da über den Verlauf der neu zu erzeugenden Trennobjekte keine Informationen zur Verfügung stehen, müssen deren Oberflächenobjekte so gewählt werden, dass keine Schnitte mit vorhandenen Objekten entstehen können. Da alle als Oberflächenobjekte in Frage kommenden Objekte die Lage $\{0\}$ haben, verlaufen die Trennobjekte aufgrund des Extremalprinzips 3.8 in der breiten Schnittfläche. Somit können keine Schnitte mit Objekten der Typen $\{+1\}$ und $\{-1\}$ auftreten, kreuzende Objekte (Typ $\{+1, 0, -1\}$) existieren zum Zeitpunkt der Trennobjekterzeugung (Schritt 4) nicht mehr.

Werden die Trennobjekte so erzeugt, dass ihre Projektionen auf die Schnittfläche die Projektionen der Oberflächenobjekte des Typs $\{0\}$ nicht schneiden, so sind auch Kollisionen mit diesen Objekten ausgeschlossen. Werden alle anderen Objekte des Typs $\{0\}$ ($\mathcal{S}^2$, $\mathcal{S}^3$) in die Berandung der Trennfläche aufgenommen, so werden auch Schnitte mit Objekten der Typen $\{+1, 0\}$ und $\{-1, 0\}$ verhindert.

Können nun Ober-Oberflächenobjekte gefunden werden, die den Rand der Projektion repräsentieren ("Extremalobjekte") und eine zulässige Oberfläche bilden, können die Zusammenhangskomponenten entlang dieser Extremalobjekte aufgespalten und der entsprechenden Seite zugeordnet werden. Die ausgewählten Extremalobjekte werden in die Berandung der Trennobjekte aufgenommen.

Überlappende Projektionen

Überlappen sich die Projektionen verschiedener Komponenten, so müssen diese Überlappungen zuvor aufgelöst und zusätzliche Verbindungsobjekte erzeugt werden, die ebenfalls in die Berandung der Trennobjekte aufgenommen werden. Ist die Projektion einer Komponente in der Projektion einer anderen enthalten, so wird die enthaltende Komponente aufgeteilt und die enthaltene Komponente einer Seite so zugeordnet, dass die Zuordnung mit der Aufteilung der enthaltenden Komponente konsistent ist, vgl. Abbildung 3.10. Sind die Projektionen zweier Komponenten gleich, so wird eine Komponente als enthaltende Komponente ausgewählt.

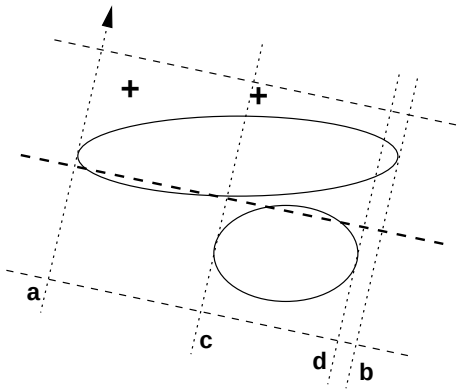


Abbildung 3.10: Überlappende Projektionen: breite Schnittfläche (gestrichelt), Komponentengrenzen (gepunktet), Überlappung zwischen (c) und (d)

Haben die Projektionen verschiedener Komponenten echte Schnitte, so liegen die Extremalobjekte der Projektion auf verschiedenen Komponenten. Dies bedeutet, dass alle Komponenten sowohl Oberflächenobjekte (Facettenteilung: Kanten, Zellteilung: Facetten) enthalten, die dem positiven Halbraum als auch Oberflächenobjekte, die dem negativen Halbraum zugeordnet werden müssen, vgl. Abbildung 3.11. Dadurch ist es notwendig, auf den Komponenten zusätzliche Ober-Oberflächenobjekte auszuzeichnen, in denen die Zuordnung zu einem Halbraum wechselt. Aus diesen Ober-Oberflächenobjekten werden zusätzliche Trennobjekte aufgebaut, die verschiedene Komponenten miteinander verbinden.

Die Wahl dieser Objekte ist nicht eindeutig. Aus Sicht der Gitterstruktur ist sie nur durch die Forderungen beschränkt, dass alle ausgewählten Ober-Oberflächenobjekte (insbesondere die Extremalobjekte) in die Berandung aufgenommen werden und dass die Verbindungsobjekte keine vorhandenen Objekte des Gitters schneiden. Da diese Fragestellung insbesondere für grobe Gitter mit mehrfach zusammenhängenden Rändern in einer frühen Phase der Gitterverfeinerung eine Rolle spielt, hat die Wahl der Schnittobjekte Einfluss auf die Gestalt der im weiteren Verlauf der Gitteradaption entstehenden Objekte.

3.5 Gittervergröberung

Der Gittervergröberung kommt insbesondere bei instationären Simulationen eine hohe Bedeutung zu. Durchwandert beispielsweise eine Schockfront das Rechengebiet, so wird

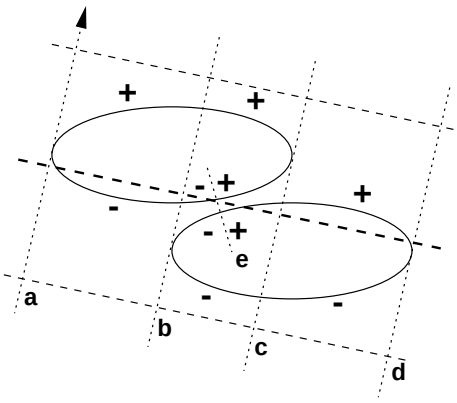


Abbildung 3.11: Überlappende Projektionen: breite Schnittfläche (gestrichelt), Komponentengrenzen (gepunktet), Überlappung zwischen (b) und (c)

das Rechengebiet an allen Stellen, die mit der Schockfront in Berührung kommen, sehr stark verfeinert. Dies führt schließlich zu einer Verfeinerung des gesamten Rechengebietes mit der für die Auflösung der Unstetigkeit notwendigen hohen Anzahl von Gittermaschen. Die lokale Vergröberung von Bereichen des Gitters, die zu fein aufgelöst sind, bietet in diesem Falle eine effiziente Möglichkeit, den numerischen Aufwand zu reduzieren.

Die Vergröberung von polyhedral bzw. polygonal berandeten Gittermaschen im Kontext der hier vorgestellten Gitterstruktur lässt sich durch Zusammenlegen benachbarter Gittermaschen leicht erreichen, weil keine Einschränkungen an die Anzahl der Oberflächen zu beachten sind. Werden alle gemeinsamen Oberflächenobjekte zweier benachbarter Maschen aus dem Gitter entfernt, so kann die neu entstehende Masche durch die verbleibenden Oberflächen beschrieben werden. Dabei bleiben alle Orientierungen erhalten. Abbildung 3.12 illustriert diesen Vorgang.

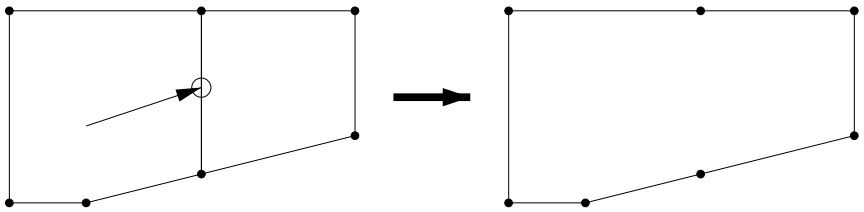


Abbildung 3.12: Zusammenlegen benachbarter Maschen

Eine lösungsorientierte Gittervergröberung, die sich z. B. an kompatiblen Rekonstruktionen in benachbarten Kontrollvolumen orientiert, kann zu äußerst unkonventionell geformten Gitterobjekten (z. B. Ringen, Schlangen) führen. Obwohl Rechnungen auf solchen Maschen möglich sind, ist bei der Verwendung solcher Maschen gegenwärtig auch kein besonderer Vorteil erkennbar. Da nur wenig Information über die Form einer Masche abgeleitet werden kann und fundierte Maßstäbe zur Bewertung einer Maschengeometrie fehlen, wird die Generierung solcher Maschen unterdrückt. Deshalb werden zur Vergröberung immer nur diejenigen Oberflächen einer Masche zugelassen, welche als letzte in die Masche eingefügt wurden. So besitzen die durch Vergröberung erzeugten Maschen die gleiche Form wie die Maschen, aus denen sie zuvor durch Gitterverfeinerung entstanden waren, vgl. Abbildung 3.13.

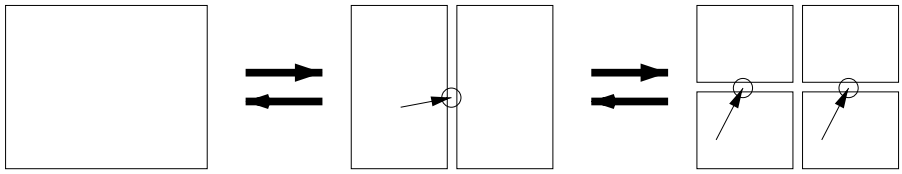


Abbildung 3.13: Vergröberung durch Rücknahme einer Verfeinerung

3.6 Gittertransformation

Die Verschiebung von Gitterknoten unter Beibehaltung der Gittertopologie ("m-refinement") erlaubt eine sehr gezielte Anpassung des Gitters an Unstetigkeiten der Lösung, [38]. Eine weitere Motivation für den Einsatz dieser Technik ist die Möglichkeit, die Gestalt der Kontrollvolumen oder die Lage der benachbarten Schwerpunkte zu verändern, [14].

Die Translation der Knoten des Gitters mittels beliebiger, knoten-spezifischer Vektoren kann jedoch zu Gitterobjekten mit negativen Längen, Flächen bzw. Volumina führen oder Schnitte zwischen Objekten hervorrufen, die nicht durch Gitterobjekte repräsentiert sind und so die Konsistenz des Gitters beschädigen. Ferner dürfen Randknoten, die die Geometrie repräsentieren, nur entlang der Oberfläche verschoben werden. Ein Ansatz zur Kontrolle der Verschiebungsvektoren bzw. der Maschengeometrie in 2D beruht auf der Betrachtung der Dreiecke, die von einer Kante und dem Schwerpunkt der Facette gebildet werden. Ist die Facette sternförmig bezüglich ihres Schwerpunktes, so haben alle diese Dreiecke einen positiven Flächeninhalt. Die Verschiebungsvektoren können so begrenzt werden, dass diese Eigenschaft erhalten bleibt. Damit kann keine Kante die Verbindungslinien der Endpunkte mit dem Schwerpunkt der Facette überqueren. Auf diese Weise kann garantiert werden, dass der Flächeninhalt der Facette positiv bleibt und keine unkontrollierten Schnitte zwischen existierenden Kanten entstehen.

In 3D können analog die Tetraeder betrachtet werden, die von einer Kante, dem Schwerpunkt einer angrenzenden Facette und dem Schwerpunkt der Zelle gebildet werden. Die Verschiebung von Gitterknoten in 3D wirft jedoch noch ein anderes Problem auf. Dazu betrachten wir zwei Facetten mit vier oder mehr Kanten, die nicht in einer Ebene liegen und eine gemeinsame Kante besitzen. Werden die Endknoten der gemeinsamen Kante in einer Richtung senkrecht zur Kante verschoben, so wird wenigstens eine der beiden Facetten dadurch gekrümmt. Dadurch können zusätzliche Diskretisierungsfehler hervorgerufen werden, etwa wenn die Berechnung der Flächen- bzw. Rauminhalte von Facetten bzw. Zellen nur ebene Flächen und eben berandete Zellen korrekt behandelt. Ein verwandtes Problem ist die Erhaltung der Konsistenzordnung der Flussintegration bei Verwendung gekrümmter Flächen. Dazu sind über korrekte Flächen- und Rauminhalte hinaus zusätzliche Integrationspunkte auf jeder Fläche notwendig. Werden die Zellen ausschließlich von dreieckige Facetten berandet, tritt dieses Problem nicht auf, weil Dreiecke unter beliebiger Translation ihrer Eckpunkte eben bleiben.

3.7 Zusammenfassung

Die in Abschnitt 3.1 spezifizierten Gitterstruktur unterstützt die Repräsentation äußerst komplexer Maschen, z. B. Abbildung 3.2. Diese Maschen können von beliebig vielen Oberflächen berandet sein, die Oberflächen können selbst wieder eine komplexe Struktur haben.

Nicht-konvexe Facetten und Zellen können ebenso dargestellt werden wie nicht einfach zusammenhängende Maschen oder Maschen mit mehrfach zusammenhängender Oberfläche.

3.7.1 Struktur von Gitteradaptionstechniken

Ein Algorithmus zur Gitteradaption besteht im Grundsatz aus zwei unterschiedlichen Teilen. Der erste Teil verändert eine vorhandene Gitterstruktur aufgrund geometrischer Argumente, z. B. werden Kanten halbiert oder Schnitte durch den Schwerpunkt der Maschen gelegt. Der zweite Teil besteht darin, die Konnektivitätsrelationen so anzupassen, dass die Einschränkungen der Gitterrepräsentation erfüllt werden können. Bevor der numerische Lösungsprozess auf dem adaptierten Gitter fortgesetzt werden kann, müssen die dafür notwendigen Geometrieparameter neu bestimmt werden.

Diese Zweiteilung führt zu sehr unterschiedlichen Adaptionstechniken. Das eine Extrem stellen dabei strukturierte Gitter dar, deren Struktur durch globale Bedingungen eingeschränkt ist. Da die Struktur der Konnektivitätsrelation global festgelegt ist, konzentrieren sich die Adaptionstechniken auf die Manipulation der durch die Knoten gegebenen geometrischen Aspekte des Gitters.

Das andere Extrem tritt bei "gitterlosen" Diskretisierungstechniken auf. Diese beruhen auf Knoten im Rechengebiet, die über beliebige Nachbarschaften (Punktwolken) miteinander verbunden sind, [41, 42, 55]. Die Bezeichnung "gitterlos" bedeutet dabei, dass die Knotennachbarschaften nicht durch Kanten beschrieben sind. Diese Nachbarschaftsbeziehungen bilden zusammen mit den Knoten einen Graphen, der die räumliche Diskretisierung des Rechengebietes beschreibt. Dieser Graph stellt das "Gitter" der gitterlosen Diskretisierung dar.

Für die Anpassung der Nachbarschaftsrelationen nach einer Adaption der Knotenmenge stehen hier keine, durch eine Gitterstruktur motivierten, geometrischen Heuristiken mehr zur Verfügung, so dass auf konventionelle Gittergeneratoren zurückgegriffen wird.

Der hier gewählte Ansatz polyhedraler Maschen ist dem gitterlosen Ansatz sehr verwandt. Aus der Maschendefinition ergeben sich nur noch wenige Einschränkungen für die Wahl der Nachbarschaftsbeziehungen. Im Gegensatz zu "gitterlosen" Gittern existiert aber noch eine geometrische Interpretation einer Gittermasche, die Anhaltspunkte für die Gitteradaption bietet.

3.7.2 Vorteile des gewählten Ansatzes

Gegenüber anderen Gittertechniken bietet der hier verfolgte Ansatz eine natürliche Möglichkeit zur Darstellung komplexer Rechengebiete, deren Repräsentation beispielsweise durch strukturierte Multiblockgitter einige Anstrengung erfordert. Gleichzeitig sind dazu nur wenige Maschen nötig, weil eine Masche beliebig viele Oberflächen haben kann. So muss ein unstrukturiertes Initialgitter wenigstens ungefähr so viele Maschen wie Oberflächen haben, um eine Geometrie überhaupt repräsentieren zu können, vgl. z. B. Abbildung 3.25 in [6]. Das Minimalgitter ist allein durch die Beschränkungen der Maschendefinition festgelegt. Im Falle eines einzelnen 2D Tragflügelprofils besteht es aus extrem gestreckten Dreiecken, die jeweils eine Kante des Profils mit einem Knoten des umschließenden Randes des Rechengebietes verbinden. Auf einem solchen Gitter Lösungen zu ermitteln bereitet Schwierigkeiten. In unserem Ansatz lassen sich auf einem Gitter für ein solches 2D-Profil,

welches ungefähr so viele Randkanten wie Kontrollvolumen hat, die Maschen frei wählen und die wesentlichen Eigenschaften der Lösung ermitteln. Die für eine Simulation notwendige Anzahl von Maschen, um eine adaptive Verfeinerungsstrategie verfolgen zu können, wird allein von numerischen Gesichtspunkten kontrolliert.

Die Flexibilität der Gitterstruktur erlaubt es, die ansonsten eng an die Erzeugung des Raumgitters gekoppelte Oberflächengittererzeugung völlig unabhängig auszuführen. Insbesondere bei unstrukturierten Tetraedergittern wird die Gittererzeugung durch Rückkopplungen zwischen diesen beiden Schritten erschwert, vgl. auch Abschnitt 2.2.3. Dabei spielt die Tatsache eine wesentliche Rolle, dass das Rechengebiete nicht von vorneherein als Gitter repräsentiert werden kann. Der hier verfolgte Ansatz erlaubt es, das Raumgitter ausschließlich durch adaptive Verfeinerung zu erzeugen. Gleichzeitig wird dadurch erreicht, dass die Gittergenerierung vollständig in den Simulationsprozess integriert werden kann und so als separater Arbeitsschritt entfällt.

Polygonale bzw. polyhedrale Kontrollvolumen werden auch in anderen Ansätzen seit Längerem eingesetzt, z. B. [61]. So werden beispielsweise in knoten-zentrierten Ansätzen polygonale Kontrollvolumen implizit dargestellt. Das numerische Verfahren benutzt dazu Kontrollvolumen, die dual zu einer explizit repräsentierten Triangulierung sind, vgl. [5]. Eine weiterer, typischer Fall, in dem polygonale Kontrollvolumen eingesetzt werden, sind Mehrgitterverfahren, deren grobe Gitter durch Agglomeration hergestellt werden, vgl. [33, 39]. Die explizite Repräsentation der Kontrollvolumen ist zwar komplizierter, bietet aber bessere Handhabe bei der Analyse und Visualisierung wesentlicher Eigenschaften des numerischen Verfahrens. Darüberhinaus wurde eine der Gitterstruktur angepasste Verfeinerungsstrategie entwickelt und implementiert, die die Flexibilität der Gitterstruktur für die numerische Simulation nutzbar macht. Weder werden allzu ungewöhnliche Maschen erzeugt, noch ist man auf einen konventionellen Gittergenerator angewiesen, der lediglich drei oder vier verschiedene Polyedertypen unterstützt.

Durch die Verwendung komplexer Maschen entfällt ein Teil des Preprocessing durch Integration in das numerische Simulationsverfahren. Alle die Gittererzeugung betreffenden Operationen sind in das Lösungsverfahren integriert: Verfeinerung, Vergröberung, Gitter für Multilevel-Verfahren, verteilte Gitter für Rechner mit verteiltem Speicher. Diese Integration stellt die herausragende Eigenschaft dieses Ansatzes dar, weil sie die Möglichkeit bietet, Gitter und Lösungswerte als Teile einer Gesamtlösung zu behandeln.

Den genannten Vorteilen stehen Nachteile gegenüber, die häufig die Stärken anderer Ansätze darstellen. Trotz intensiver Arbeit auf dem Gebiet der Gittergenerierung hat sich bis heute keine Gittertechnik in allen Anwendungsbereichen durchsetzen können. Stattdessen hat sich eine wachsende Palette von Techniken entwickelt, die jeweils spezifische Aspekte in den Vordergrund stellen. Diese Entwicklung verdeutlicht das Spannungsfeld widersprüchlicher Anforderungen, denen die numerische Simulation ausgesetzt ist, z. B. die Notwendigkeit hohe Rechenleistung zu erzielen, die Notwendigkeit adaptive Techniken einzusetzen oder die steigende Komplexität der betrachteten Untersuchungsgebiete.

3.7.3 Speicherbedarf

Die Repräsentation des Gitters durch Knoten, Kanten, Facetten und Zellen sowie aller Relationen zwischen diesen Gittern bedeutet einen zusätzlichen Speicheraufwand. Im Vergleich dazu werden in einem Tetraedergitter nur Knoten, Tetraeder und die Konnektivität von Tetraedern zu Knoten repräsentiert.

Die Speicherung aller Gitterobjekte und Relationen wurde aber trotzdem vorgesehen, weil diese Entscheidung am Anfang der Entwicklung getroffen werden musste, ohne dass die konkreten Anforderungen der Gitterteilung oder des numerischen Verfahrens bereits klar gewesen wären. Sie bot eine zuverlässige Grundlage, auf der die Implementierung durchgeführt werden konnte. Darüberhinaus bestand die Notwendigkeit, einen klaren Entwicklungspfad von 2D nach 3D vorzusehen. Die dafür gefundene Lösung beruht auf einer dimensionsunabhängigen Formulierung des numerischen Verfahrens, verknüpft mit einer selbstähnlichen Struktur der Geometrie. Diese Selbstähnlichkeit beruht auf der Speicherung aller Gitterobjekte und wird in Abschnitt 5.2.1 genauer erläutert.

Da zwischen den Anzahlen der verschiedenen Gitterobjekte keine festen Beziehungen bestehen, sind alle Angaben auf die Kontrollvolumen bezogen. In der vorliegenden Implementierung werden für einen typischen 2D-Fall (NLR-7301-Profil mit Klappe, 10576 Kontrollvolumen) pro Facette ca. 2.4 KB Speicherplatz benötigt, davon 1.3 KB in der Facette selbst und 0.2 KB für die Konnektivität. Das Gitter enthält 1.3 Knoten und 2.2 Kanten pro Facette. In einem 3D-Testfall (Kanal mit zwei Rampen, 16326 Kontrollvolumen) wurden pro Zelle insgesamt ca. 3.1 KB Speicherplatz benötigt, davon 1.3 KB in der Zelle selbst und 0.5 KB für die Konnektivität. Das Gitter enthält ca. 2.6 Facetten, 2.8 Kanten und 1.0 Knoten pro Zelle. Realistische Testfälle müssen zeigen, ob diese Zahlen für 3D typisch sind.

3.7.4 Rekursivität des Teilungsalgorithmus

Wie bereits beschrieben, folgt der Teilungsvorgang dem Baum der Konnektivitätsrelation vom aktuellen Kontrollvolumen bis zu den Knoten. Während der Baum der Relationen zwischen dem Wurzelobjekt und den Knoten abgearbeitet wird, werden Daten, die von der Wahl der Schnittfläche und den Relationen abhängig sind, gesammelt.

Jedes Objekt wertet die von seinen Relationen gesammelten Informationen aus und gibt eine Zusammenfassung der Information den Baum hinauf zurück. Diese Daten sind in natürlicher Weise mit den Relationen des aktuellen Baumes verknüpft und werden nicht länger benötigt, wenn ein Zweig des Baumes abgearbeitet ist. Diese temporären Daten können auf dem Aufrufstapel zusammen mit der Funktionsaufruffolge des Algorithmus gespeichert werden, d. h. dass die Teilung eines Objektes komplett abgeschlossen sein muss, bevor ein anderes Objekt des gleichen Levels bearbeitet werden kann.

Dies bedeutet, dass der Verfeinerungsalgorithmus auf Vektorarchitekturen (SIMD) nicht einfach zu vektorisieren ist. Für die Vektorisierung müssten die kurzen Schleifen den Baum hinab mit den Schleifen über alle Objekte vertauscht werden. Neben anderen Schwierigkeiten müssten dazu alle temporären Daten zwischengespeichert werden bis der nächst höhere Level fertig bearbeitet ist. Die Größe der anfallende Datenmenge ist proportional zur Anzahl aller Pfade durch den Konnektivitätsbaum und würde die Größe des Speicherplatzes, der für das Gitter notwendig ist, deutlich übersteigen. Dadurch würde die maximal mögliche Größe eines Gitters auf einen Bruchteil des verfügbaren Speichers beschränkt.

Diese Überlegung führt zu dem Schluss, dass eine Vektorisierung des Verfeinerungsalgorithmus auf diese Weise nicht erreicht werden kann, weil der vorgestellte Algorithmus große Mengen temporärer Daten produziert, die in rekursiver Weise berechnet, gespeichert und genutzt werden müssen.

Kapitel 4

Eine Finite-Volumen-Methode auf allgemeinen Maschen

Dieses Kapitel beschreibt die gewählte, zell-zentrierte Finite-Volumen-Methode und ihre Komponenten: das grundlegende Finite-Volumen-Schema, die Berechnung und Integration der Flüsse (4.2), die Techniken der linearen Rekonstruktionen (4.3), die Implementierung der Randbedingungen (4.4), das Zeitschrittverfahren (4.5) und die Limitierung der Rekonstruktion (4.6). Dieses Verfahren wird in der vorliegenden Arbeit exemplarisch auf das mathematische Modell reibungsfreier, kompressibler Strömungen angewendet (vgl. Kapitel 6), ist darauf aber nicht beschränkt. In Abschnitt 4.1 werden deshalb das beschreibende System der Euler-Gleichungen vorgestellt.

Für das Verständnis und die Beschreibung physikalischer Vorgänge haben sich Erhaltungs- und Extremalprinzipien als Grundbausteine erwiesen. So beschreibt ein Erhaltungsgesetz das Prinzip, dass sich die betrachteten Zustandsgrößen nur durch Zu- oder Abfluss über die Systemgrenzen ändern, während einem Extremalgesetz die Tatsache zu Grunde liegt, dass ein System sich im Gleichgewichtszustand befindet, wenn die enthaltene "Energie" minimal ist.

In mathematischer Form stellen sich Erhaltungsprinzipien als partielle Differentialgleichungen in Divergenzform und Extremalprinzipien als Minimierungsaufgabe über Integralformen (Energiefunktionale) dar. Unter erhöhten Anforderungen an die Differenzierbarkeitseigenschaften der Lösung können sie äquivalent als partielle Differentialgleichungen formuliert werden. Obwohl der stationären Erhaltungsgleichung, Glg. 4.7, keine Extremalaufgabe entspricht, existieren für eine Reihe wichtiger Fälle äquivalente Formulierungen als Extremalaufgabe, z. B. für die reibungs- und rotationsfreie, isentrope Strömung.

4.1 Erhaltungsgleichungen

Als der grundlegende Mechanismus für die Beschreibung von Strömungsvorgängen wird die Erhaltung von Masse, Impuls und Energie angesehen, [26]. Deshalb soll auf die allgemeine Erhaltungsgleichung in integraler Form und die zugehörige Darstellung als partielle Differentialgleichung eingegangen werden. Darüberhinaus sollen die Erhaltungsgleichungen für die reibungsfreie Strömung (Euler-Gleichungen) eines idealen Gases sowie Modellgleichungen vorgestellt werden, die Teilaspekte der Euler-Gleichungen modellieren und zur

Verifikation der numerischen Verfahren eingesetzt wurden. Eine ausführliche Darstellung verschiedener, mathematischer Modelle zur Approximation der Strömung von Fluiden findet sich z. B. in [26].

Die allgemeine Erhaltungsgleichung

Die in Zeit und Raum integrierte Form der instationären Erhaltungsgleichung,

$$\int_\Omega \varphi(t_1,\vec{x})\,\mathbf{d}\lambda(\vec{x}) + \int_\Omega \varphi(t_0,\vec{x})\,\mathbf{d}\lambda(\vec{x}) + \int_{\partial\Omega}\int_{t_0}^{t_1} F(\varphi(t,\vec{x}))\mathbf{d}t\,\mathbf{d}\sigma(\vec{x}) = 0 \tag{4.1}$$

verknüpft die Änderung der Zustandsgrößen

$$\int_\Omega \varphi(t,\vec{x})\,\mathbf{d}\lambda(\vec{x}) \in R^m$$

im Kontrollvolumen $\Omega \subset R^d$ im Zeitinterval $]t_0,t_1[$ mit dem Fluss über die Oberfläche $\partial\Omega$ des Kontrollvolumens, beschrieben durch den Flusstensor $F \in R^{m\times d}$, der von der Dichte φ der Zustandsgrößen und der nach außen gerichteten Oberflächennormale σ abhängt. Die Abhängigkeit des Flusstensors von der Oberflächennormale ist linear.

Ist die Lösung differenzierbar, so erhält man durch Anwendung des Gaußschen Integralsatzes auf Glg. 4.1 die dazu äquivalente Formulierung als partielle Differentialgleichung,

$$\partial_t\varphi(t,\vec{x}) + \operatorname{div} F(\varphi(t,\vec{x})) = 0 \tag{4.2}$$

und durch Anwendung der Kettenregel die nicht-konservative Darstellung

$$\partial_t\varphi(t,\vec{x}) + \sum_{i=1\ldots d} \frac{\partial F_i}{\partial\varphi}(\varphi(t,\vec{x}))\frac{\partial\varphi}{\partial\vec{x}_i}(t,\vec{x}) = 0 \tag{4.3}$$

Trotz glatter Anfangs- und Randbedingungen können die Lösungen von Erhaltungsgleichungen Unstetigkeiten aufweisen, deren numerische Behandlung eine besondere Herausforderung für die Diskretisierung darstellt.

Das System der Euler-Gleichungen

Das folgende Gleichungssystem beschreibt den reibungsfreien Transport (3D) eines Fluids, definiert durch die Erhaltungsgrößen Masse, Impuls und Totalenergie

$$\partial_t \underbrace{\begin{pmatrix} \rho \\ \rho v_1 \\ \rho v_2 \\ \rho v_3 \\ \rho E \end{pmatrix}}_{=:\varphi} + \operatorname{div} \underbrace{\begin{pmatrix} \rho v_1 & \rho v_2 & \rho v_3 \\ \rho v_1^2 + p & \rho v_1 v_2 & \rho v_1 v_3 \\ \rho v_2 v_1 & \rho v_2^2 + p & \rho v_2 v_3 \\ \rho v_3 v_1 & \rho v_3 v_2 & \rho v_3^2 + p \\ (\rho E + p)v_1 & (\rho E + p)v_2 & (\rho E + p)v_3 \end{pmatrix}}_{=:F(\varphi)} = 0 \tag{4.4}$$

in den Variablen (Massen-)Dichte ρ, Druck p, Geschwindigkeit $\vec{v} = (v_1, v_2, v_3)^t$ und Totalenergie E. Das Gleichungssystem stellt sich in vektorieller Notation mit dem Vektor der Zustandsdichten φ wie folgt dar

$$\partial_t\varphi + \operatorname{div} F(\varphi) = 0 \quad \text{mit } F(\varphi) = \varphi\cdot\vec{v}^t + p\begin{pmatrix} 0^t \\ \mathrm{Id} \\ \vec{v}^t \end{pmatrix} \tag{4.5}$$

und wird durch eine algebraische Zustandsgleichung für das Fluid geschlossen. Für ideale Gase nimmt diese Zustandsgleichung die Form

$$p = (\gamma - 1)(\rho E - \frac{1}{2}\rho||\vec{v}||^2) \tag{4.6}$$

an. Dabei bezeichnet c_p bzw. c_v die spezifischen Wärmekoeffizienten bei konstantem Druck bzw. konstantem Volumen und $\gamma = c_p/c_v$ ihr Verhältnis (für Luft gilt: $\gamma \approx 1.4$).

4.2 Räumliche Diskretisierung

Ausgehend von den drei Darstellungsformen (Erhaltungsgleichung, Extremalaufgabe und partielle Differentialgleichung) wurden verschiedene Diskretisierungstechniken entwickelt, die jeweils den Charakter der zu Grunde liegenden mathematischen Formulierung betonen.

So spiegelt der *Finite-Differenzen*-Ansatz die Bildung partieller Ableitungen wider, während der *Finite-Elemente*-Ansatz die Approximation schwacher Ableitungen zum Ausgangspunkt nimmt und durch diskrete Extremalprinzipien motiviert ist. Ebenso wie der Finite-Elemente-Ansatz beruht der *Finite-Volumen*-Ansatz auf schwachen Ableitungen, führt aber auf diskrete Erhaltungsprinzipien. Die entstehenden diskreten Formulierungen sind in Spezialfällen zueinander äquivalent. Über die bereits erwähnten Ansätze und deren Varianten hinaus finden noch verschiedene andere Diskretisierungstechniken Anwendung, z. B. Spektralmethoden [37], Partikel-Verfahren [19, 55, 66, 67]) und kinetische Verfahren [44].

Für die Anwendung auf Probleme der Strömungsmechanik, die durch Erhaltungsprinzipien beschrieben sind, erscheint die Wahl eines Finite-Volumen-Ansatzes vorteilhaft, der dieses Prinzip auch diskret widerspiegelt. Diese Eigenschaft ist umso wichtiger, je geringer die zu erwartende "Qualität" der Gitter ist, da der Schwerpunkt der vorliegenden Arbeit auf der Untersuchung der numerischen Eigenschaften einer höchst flexiblen Gitterstruktur (vgl. Kapitel 3) liegt. Darüberhinaus ist die Formulierung in natürlicher Weise auf beliebigen Maschen einsetzbar, erlegt also keine Einschränkungen bei der Wahl der Maschenformen auf. Demgegenüber werden Finite-Differenzen- und Finite-Elemente-Ansätze auf festen Maschenformen, z. B. Dreiecken, Vierecken und ihren 3D-Äquivalenten angewendet.

In diesem Abschnitt sollen die Bausteine der räumlichen Diskretisierung eines zell-zentrierten Finite-Volumen-Verfahrens beschrieben sowie die Grundlagen numerischer Flussfunktionen und ihre numerischen Integration dargestellt werden. Die übrigen Aspekte der Diskretisierung werden in den folgenden Abschnitten diskutiert: Rekonstruktionstechniken in Abschnitt 4.3 und 4.6, die Behandlung der Randbedingungen in Abschnitt 4.4 und das Zeitschrittverfahren in Abschnitt 4.5.

4.2.1 Finite-Volumen(FV)-Diskretisierung

Diskretisierungstechnik

Ausgehend von der zu Grunde liegenden stationären Differentialgleichung

$$\operatorname{div} F(\varphi) = 0 \tag{4.7}$$

erhält man das diskrete Gleichungssystem durch Integration über ein Bilanzvolumen Ω und die Anwendung des Gauß'schen Integralsatzes (partielle Integration). Dabei wird das Raumintegral über die Terme der räumlichen Ableitungen in ein Oberflächenintegral über den Flusstensor F mit dem nach außen gerichteten Normalenvektor σ umgewandelt:

$$\int_{\partial\Omega} F(\varphi)\,\mathbf{d}\sigma = 0\,.$$

Das Oberflächenintegral wird durch numerische Integration ausgewertet. Die dazu benötigten Werte von φ an den Integrationspunkten werden mit Hilfe der Lösung eines Approximationsproblems (Rekonstruktion) aus den Mittelwerten der Zustandsdichte ermittelt, die als Unbekannte dienen.

Varianten des Finite-Volumen-Verfahrens

Es haben sich verschiedene Varianten dieser Methode entwickelt, die nach der Lokalisierung der Unbekannten im Gitter und der Wahl der Kontrollvolumina unterschieden werden.

Der *zell-zentrierte* ("cell-centered") Ansatz benutzt die Maschen des gegebenen Gitters als Kontrollvolumen und lokalisiert die Unbekannten in den Schwerpunkten der Maschen. Die Werte an den Integrationspunkten der Maschenoberflächen werden durch Ermittlung einer Rekonstruktion extrapoliert. So stehen an jedem Integrationspunkt zwei verschiedene Werte aus den angrenzenden Bilanzvolumen zur Verfügung, die zur näherungsweisen Lösung eines eindimensionalen lokalen Riemann-Problems herangezogen werden.

Der *knoten-zentrierte* ("node-centered") Ansatz lokalisiert die Unbekannten in den Knoten des Gitters, als Kontrollvolumen werden duale Maschen benutzt. Diese können beispielsweise aus der Verbindung der Kantenschwerpunkte ("centroid dual") oder zusätzlich der Facetten- und Zellschwerpunkte ("median dual") entstehen. Dieser Ansatz bietet eine Reihe praktischer Vorteile. So können die Werte an den Integrationspunkten durch lineare Interpolation aus den angrenzenden Knoten ermittelt werden (centroid dual/unstrukturiertes Gitter, oder bei strukturierten Gittern). Der Ansatz führt auf einem unstrukturierten Dreiecksgitter auf vielseitige Kontrollvolumen und wird besonders in 2D bevorzugt eingesetzt. Bei reibungsbehafteten Strömungen können zur Diskretisierung der Reibungsterme Finite-Elemente-Techniken eingesetzt werden.

Beim Zelleckpunktsansatz ("cell-vertex") werden knoten-zentrierte Variable und die durch das Gitter gegebenen Maschen als Kontrollvolumen benutzt. Das maschen-zentrierte Residuum wird mit Hilfe von Gewichtungsoperatoren auf die Knoten verteilt. Eine Variante dieser Verfahren, das "multi-dimensional upwinding", wählt die Gewichte entsprechend einer lösungsorientierten Analyse der Transportrichtungen [50].

Die hier vorgestellten Varianten der Finite-Volumen-Methode weisen im Hinblick auf den Einsatz polyhedraler Maschen verschiedene Eigenschaften auf. Der zell-zentrierte Ansatz bietet die klare Analogie zwischen Gitter und Kontrollvolumen und erleichtert dadurch die Analyse und Visualisierung der Verfahrensparameter, die nicht durch ihre Zugehörigkeit zu dualen, aber unsichtbaren Maschen verwischt werden. Darüberhinaus läßt sich der Ansatz in einfacher Weise von 2D auf 3D erweitern. Der knoten-zentrierte Ansatz bietet den Vorteil, auf einem unstrukturierten Gitter aus Dreiecken bzw. Tetraedern Kontrollvolumen mit vielen Oberflächen bereitzustellen. Stellt man die so erhaltenen Kontrollvolumen (beispielsweise des centroid dual) aber selbst als Maschen dar und bildet erneut das duale

Gitter, so erhält man im Wesentlichen das Ausgangsgitter zurück, d. h. die Verwendung polyhedraler Maschen bringt keinen Vorteil. Die bei einem Zelleckpunktsansatz notwendigen Gewichtungsoperatoren beruhen auf elementargeometrischen Überlegungen, deren Anwendung auf polygonale Maschen problematisch sind. Besonders deutlich wird dies, wenn beim "multi-dimensional upwinding" die Kante identifiziert werden muss, die in Konvektionsrichtung einem Knoten der Masche gegenüberliegt.

So stellt die ausgewählte zell-zentrierte Finite-Volumen-Diskretisierung eine dem zu lösenden Problem und der gewünschten Gitterrepräsentation angepasste, erfolgversprechende Standardtechnik dar.

4.2.2 Diskretisierungsansatz

Auf einem in offene, nicht-überlappende Kontrollvolumen $\Omega_j \subset R^d$, $(j \in J)$, aufgeteilten Rechengebiet $\bar{\Omega} = \overline{\sum_j \Omega_j}$ und dem Zeitintervall $]t_0, t_1[\subset R$ führt das zell-zentrierte Finite-Volumen-Verfahren auf den folgenden Diskretisierungsansatz für Glg. 4.1:

$$\int_{\Omega_j} \varphi(t_1, \vec{x})\, \mathbf{d}\lambda = \int_{\Omega_j} \varphi(t_0, \vec{x})\, \mathbf{d}\lambda - \sum_{k \in N_j} \int_{t_0}^{t_1} \int_{\partial\Omega_{jk}} F\left(\varphi(t, \vec{x})\right)\, \mathbf{d}\sigma \mathbf{d}t\,,$$

wobei N_j die Indizes der über Seiten benachbarten Kontrollvolumen von Ω_j enthält und $\partial\Omega_{jk} := \bar{\Omega}_j \cap \bar{\Omega}_k$ den an Ω_k grenzenden Teil der Oberfläche von Ω_j bezeichnet.

Wird der Inhalt des Kontrollvolumens Ω_j mit

$$\lambda_j := \lambda(\Omega_j) := \int_{\Omega_j} 1\, \mathbf{d}\lambda$$

bezeichnet und als Unbekannte der mittlere Zustand im Kontrollvolumen

$$\bar{\varphi}_j(t) := \frac{1}{\lambda_j} \int_{\Omega_j} \varphi(t, \vec{x})\, \mathbf{d}\lambda \tag{4.8}$$

eingesetzt, so erhält man

$$\bar{\varphi}_j(t_1) = \bar{\varphi}_j(t_0) - \frac{1}{\lambda_j} \sum_{k \in N_j} \int_{t_0}^{t_1} \int_{\partial\Omega_{jk}} F\left(\varphi(t, \vec{x})\right)\, \mathbf{d}\sigma \mathbf{d}t\,.$$

Die Auswertung des Oberflächenintegrals über $\partial\Omega_{jk}$ erfolgt über eine Quadraturformel mit $\tilde{n}$ Integrationspunkten $\vec{x}_{jk,n}$ und Seitennormalen $\vec{s}_{jk,n}$, $(n = 1 \ldots \tilde{n})$. Bezeichnet man das Produkt des Flusstensors $F(\varphi)$ mit der Seitennormalen $\vec{s}$ als Flussvektor $f(\varphi, \vec{s})$, so erhält man die folgende Approximation des Oberflächenintegrals

$$\int_{\partial\Omega_{jk}} F\left(\varphi(t, \vec{x})\right)\, \mathbf{d}\sigma \approx \sum_n f\left(\varphi(t, \vec{x}_{jk,n}), \vec{s}_{jk,n}\right)\,.$$

Um den Ansatz zu schließen, wird eine Beziehung zwischen $\bar{\varphi}_j(t)$ und $\varphi(t, \vec{x})$ bzw. $F(\varphi(t, \vec{x}))$ benötigt. Wir wählen hier den Weg, aus den Mittelwerten $\bar{\varphi}_k$ der Umgebung $\bar{N}_j$ interpolierende Funktionen $\varphi_j(\vec{x})$ zu rekonstruieren (MUSCL-Ansatz, vgl. Abschnitt 4.3.1) und mit den aus den Rekonstruktionen gewonnenen Werten an den Integrationspunkten den

Flussvektor durch einen Godunov-Ansatz zu approximieren: die zum Zeitpunkt t_0 gegebene Lösung soll entlang der Charakteristiken transportiert werden, vgl. Abschnitt 4.2.3.

So stehen zur Auswertung des numerischen Flussvektors f^* an jedem Integrationspunkt zwei Werte zur Verfügung:

$$f\left(\varphi(t,\vec{x}_{jk,n}),\vec{s}_{jk,n}\right) \approx f^*\left(\varphi_j(t,\vec{x}_{jk,n}),\varphi_k(t,\vec{x}_{jk,n}),\vec{s}_{jk,n}\right) .$$

Somit erhält man das folgende Gleichungssystem in der Zeit:

$$\bar{\varphi}_j(t_1) = \bar{\varphi}_j(t_0) - \frac{1}{\lambda_j}\int_{t_0}^{t_1}\sum_{k\in N_j}\sum_n f^*\left(\varphi_j(t,\vec{x}_{jk,n}),\varphi_k(t,\vec{x}_{jk,n}),\vec{s}_{jk,n}\right) \tag{4.9}$$

Dieses Gleichungssystem muss durch die Wahl einer Rekonstruktionstechnik (Abschnitt 4.3, 4.6), einer numerischen Flussfunktion (Abschnitt 4.2.3) und einer Quadraturformel für das Oberflächenintegral (Abschnitt 4.2.4) vervollständigt werden. Die Implementierung der Randbedingungen wird in Abschnitt 4.4 und die Approximation des Zeitintegrals durch ein Zeitschrittverfahren in Abschnitt 4.5 diskutiert.

4.2.3 Das Riemann-Problem und numerische Flüsse

Numerische Flussfunktionen können aus der Analyse des *Riemann-Problems* abgeleitet werden, welches für ein System eindimensionaler, hyperbolischer Erhaltungsgleichungen

$$\partial_t\varphi + \partial_x F(\varphi) = 0$$

mit stückweise konstanten Anfangsbedingungen

$$\varphi(0,x) = \begin{cases} \varphi_L & , \quad \forall x < 0 \\ \varphi_R & , \quad \forall x > 0 \end{cases}$$

gegeben ist, vgl. [52]. Dessen Lösung ist entlang der Strahlen $x/t = \xi$ konstant, d. h. die Lösung des Riemann-Problems hängt nur von φ_L, φ_R und ξ, dem sogenannten *Riemann-Löser*

$$\varphi(t,x) = R(\varphi_L,\varphi_R,x/t)$$

ab. Da die exakte Lösung des Riemann-Problems zu aufwändig ist, werden Näherungen R^A ("approximate Riemann solver") eingesetzt, die mit dem Fluss konsistent sind, und mit dem Riemann-Löser im Mittel übereinstimmen

$$R^A(\varphi,\varphi,\xi) = \varphi$$
$$\int_{-\infty}^{+\infty}\left(R^A(\varphi_L,\varphi_R,\xi) - R(\varphi_L,\varphi_R,\xi)\right)\mathbf{d}\xi = 0$$

Die numerische Flussfunktion erhält man dann aus

$$F^*(\varphi_L,\varphi_R) = F\left(R^A(\varphi_L,\varphi_R,0)\right) .$$

Der von Godunov vorgeschlagene Fluss entsteht durch Einsetzen des exakten Lösers R. Im Rahmen dieser Arbeit wurde die Approximation des Riemann-Lösers von Roe [48]

$$F^*(\varphi_L,\varphi_R) = \frac{1}{2}\left(F(\varphi_L) + F(\varphi_R) - \left|\frac{\mathbf{d}F}{\mathbf{d}\varphi}(\varphi_M)\right|(\varphi_R - \varphi_L)\right) \tag{4.10}$$

eingesetzt, welche auf einem Zustand φ_M beruht, der durch

$$\frac{\mathbf{d}F}{\mathbf{d}\varphi}(\varphi_M) = F(\varphi_R) - F(\varphi_L) \tag{4.11}$$

gegeben ist. Da diese Flussfunktion Expansionsschocks zulässt, wurde eine Entropiekorrektur benutzt [25]. Dieser Ansatz liefert genaue Lösungen, erfordert aber hohen Rechenaufwand. Es sind aber auch Mängel bekannt ("carbuncle"-Phänomen, [43]).

4.2.4 Quadraturformeln

Der Finite-Volumen-Diskretisierung liegt die Approximation des Raumintegrals durch das Oberflächenintegral

$$\int_\Omega \operatorname{div}\left(F(\vec{x})\right)\,\mathbf{d}\lambda = \int_{\partial\Omega} F(\vec{x})\,\mathbf{d}\sigma \tag{4.12}$$

(Satz von Gauß) zu Grunde. Hier sollen die Approximationseigenschaften dieses Ansatzes untersucht werden. Dazu setzen wir den Flusstensor $\vec{q}(\vec{x}) = F(\varphi(\vec{x}))$ einer skalaren Erhaltungsgleichung

$$\vec{q}(\vec{x}) = (q_i(\vec{x}))_{i=1\ldots d} \in R^d$$

als ein vektorwertiges, quadratisches Polynom

$$q_i(\vec{x}) = q_i^{(0)} + \langle q_i^{(1)}, \vec{x}\rangle + \langle q_i^{(2)}, \vec{x}\,\vec{x}^t\rangle$$

mit den Koeffizienten des i-ten Polynoms

$$q_i^{(0)} \in R,\ q_i^{(1)} = \left(q_{i,n}^{(1)}\right)_{n=1\ldots d} \in R^d,\ q_i^{(2)} = \left(q_{i,nm}^{(2)}\right)_{n,m=1\ldots d} \in R^{d\times d},$$

an. Dabei kann $q_i^{(2)}$ o. E. als symmetrische Matrix angenommen werden. Dabei bezeichnet $\vec{e}_i$ den i-ten Einheitsvektor und die Schreibweise $\langle A, B\rangle$ für zwei Matrizen A und B das Skalarprodukt $\sum_{ij=1\ldots d} a_{ij}b_{ij}$, mit $\langle A, a\,b^t\rangle = b^t A\,a$ $(a, b \in R^d)$.

Für das Raumintegral erhält man

$$\begin{aligned}
\int_\Omega \operatorname{div}\left(\vec{q}(\vec{x})\right)\,\mathbf{d}\lambda &= \int_\Omega \sum_i \partial_i q_i(\vec{x})\,\mathbf{d}\lambda \\
&= \int_\Omega \sum_i \left(\langle q_i^{(1)}, \vec{e}_i\rangle + \langle q_i^{(2)}, \vec{e}_i\,\vec{x}^t + \vec{x}\,\vec{e}_i^t\rangle\right)\,\mathbf{d}\lambda \\
&= \sum_i \langle q_i^{(1)}, \vec{e}_i\,(\int_\Omega 1\,\mathbf{d}\lambda)\rangle \\
&\quad + \sum_i \langle q_i^{(2)}, \vec{e}_i\,(\int_\Omega \vec{x}\,\mathbf{d}\lambda)^t + (\int_\Omega \vec{x}\,\mathbf{d}\lambda)\,\vec{e}_i^t\rangle
\end{aligned} \tag{4.13}$$

Dies zeigt, dass die Koeffizienten $q_i^{(0)}$ $(i = 1\ldots d)$, $q_{i,n}^{(1)} (n \neq i)$ und $q_{i,nm}^{(2)} (n \neq i, m = 1\ldots d)$ in das Integral nicht eingehen.

Unter Verwendung einer Ein-Punkt-Quadraturformel mit Schwerpunkt $\vec{x}_k = (x_{k,i})_{i=1\ldots d}$ und der nach außen gerichteten Oberflächennormalen $\vec{s}_k = (s_{k,i})_{i=1\ldots d}$ ermittelt man für

das Oberflächenintegral

$$\begin{aligned}\int_{\partial\Omega} \vec{q}(\vec{x})\,\mathbf{d}\sigma &\approx \sum_{k\in N}\sum_i s_{k,i} q_i(\vec{x}_k) \\ &= \sum_{k\in N}\sum_i s_{k,i} q_i^{(0)} + \sum_{k\in N}\sum_i s_{k,i}\langle q_i^{(1)}, \vec{x}_k\rangle + \sum_{k\in N}\sum_i s_{k,i}\langle q_i^{(2)}, \vec{x}_k\vec{x}_k^t\rangle \\ &= \sum_i q_i^{(0)}(\sum_{k\in N} s_{k,i}) + \sum_i \langle q_i^{(1)}, \sum_{k\in N} s_{k,i}\vec{x}_k\rangle + \sum_i \langle q_i^{(2)}, \sum_{k\in N} s_{k,i}\vec{x}_k\vec{x}_k^t\rangle \end{aligned} \tag{4.14}$$

Durch Koeffizientenvergleich verifiziert man, dass die numerische Auswertung der linken Seite von Glg. 4.12 durch die rechte Seite dieser Gleichung für quadratische Flussfunktionen exakt ist, wenn die folgenden Beziehungen gelten:

$$\sum_{k\in N} \vec{s}_{k,i} = 0 \tag{4.15}$$

$$\sum_{k\in N} \vec{s}_{k,i}\vec{x}_k = (\int_\Omega 1\,\mathbf{d}\lambda)\cdot\vec{e}_i \tag{4.16}$$

$$\sum_{k\in N} \vec{s}_{k,i}\vec{x}_k\vec{x}_k^t = (\int_\Omega \vec{x}\,\mathbf{d}\lambda)\,\vec{e}_i^t + \vec{e}_i\,(\int_\Omega \vec{x}\,\mathbf{d}\lambda)^t \tag{4.17}$$

Diese Beziehungen werden im Anhang, Abschnitt A.5, für eben berandete Kontrollvolumen untersucht. Die erste Bedingung, Glg. 4.15, ist dabei erfüllt, weil das Kontrollvolumen eine geschlossene Oberfläche hat und sich deshalb die Oberflächennormalenvektoren $\vec{s}_k$ zu Null addieren (vgl. Glg. A.11). Die Untersuchung zeigt für die beiden anderen Bedingungen, dass die auf der linken Seite eingesetzten Momente von einer Ordnung höher sein müssen, als die Momente der rechten Seite. So kann Glg. 4.16 verifiziert werden (vgl. Glg. A.12). Dagegen zeigt Glg. A.13, dass die rechte Seite von Glg. 4.17 durch die zweiten Momente $m_k^{(2)}$ der Seiten $\partial\Omega_k$

$$\sum_{k\in N} \vec{s}_{k,i} m_k^{(2)}$$

dargestellt wird, die nicht mit dem Quadrat $\vec{x}_k\vec{x}_k^t$ der ersten Momente übereinstimmen. Das heißt, das Glg. 4.17 nicht erfüllt ist. Dies bedeutet, dass auf beliebigen, eben berandeten Kontrollvolumen die Flussdivergenz *nicht* von zweiter Ordnung genau integriert wird, wenn eine Quadraturformel erster Ordnung für die Auswertung des Oberflächenintegrals herangezogen wird. Diese Überlegung ist deshalb von Bedeutung, weil eine lineare Rekonstruktion zu einer quadratischen Abhängigkeit der Flussfunktion der Euler-Gleichungen vom Ort führt.

4.3 Rekonstruktion

Die in Glg. 4.9 vorgestellte Finite-Volumen-Methode sieht die Auswertung der Flüsse auf den Oberflächen der Kontrollvolumen vor. Da die Unbekannten $\bar{\varphi}_j$ als Zellmittelwerte der konservativen Variablen definiert sind, stehen diese Flüsse an Integrationspunkten nicht von vorneherein zur Verfügung und müssen aus den Zellmittelwerten approximiert werden. Diese Approximation ist unabhängig von der Bestimmung des numerischen Flusses durch einen angenäherten Riemann-Löser und bestimmt zusammen mit der Flussintegration die Qualität der räumlichen Diskretisierung im Inneren des Rechengebietes.

4.3.1 Die Technik der Variablen-Extrapolation

Wir wählen hier den Ansatz, nicht die Flüsse direkt zu extrapolieren, sondern die Flüsse mittels extrapolierter Werte, den Rekonstruktionen, zu ermitteln. Dieser Ansatz wird in Anlehnung an das erste, von Van Leer [34] vorgestellte Verfahren diesen Typs als MUSCL-Ansatz bezeichnet ("monotone upwind-centered schemes for conservation laws").

Definition 4.1 (Rekonstruktion) *Unter einer* Rekonstruktion *wird die Lösung eines Interpolationsproblems auf einem Gebiet $\Omega \subset R^d$ verstanden, welches in Kontrollvolumen $\{\Omega_j\}_{j\in J}$ diskretisiert ist. Auf dem Gebiet sei eine unbekannte Funktion $\varphi : R^d \to R$ durch ihre Mittelwerte $\{\frac{1}{\lambda_j}\int_{\Omega_j}\varphi\,\mathbf{d}\lambda\}_{j\in J}$ gegeben. Diese Funktion φ soll in einem Unterraum von Ansatzfunktionen optimal approximiert werden, das heißt, dass Funktionen aus dem Ansatzraum exakt rekonstruiert werden.*

Enthält der Raum der Ansatzfunktionen alle Polynome bis zur Ordnung n, so spricht man man von einer polynomialen Rekonstruktion n-ter Ordnung. Ein etwas anderer Ansatz wird mit der ENO-Rekonstruktion verfolgt, die z. B. in [2] dargestellt wird. Dabei werden andere Forderungen an die Approximationseigenschaften der Rekonstruktion gestellt, so dass sie nicht mehr limitiert werden muss.

4.3.2 Ansätze zur Berechnung einer Rekonstruktion

Eine lineare Rekonstruktion kann durch Approximation der ersten Ableitung der unbekannten Funktion aus den Zellmittelwerten ermittelt werden. Die Interaktion der Gradientenberechnung mit den numerischen Randbedingungen stellt bei diesem Ansatz eine besondere Schwierigkeit dar, vgl. Abschnitt 4.4.1.

Lineare Interpolation

Den einfachsten Ansatz für die Berechnung eines Gradienten beinhaltet die knoten-zentrierte FV-Methode als Konstruktionsprinzip. Da die Rekonstruktion nicht überall im Kontrollvolumen benötigt wird, sondern nur an den Gaußpunkten der Flussquadratur, liefert die lineare Interpolation zwischen den angrenzenden Werten bereits genaue Resultate, siehe Abbildung 4.1.a . Da in diesem Fall in die numerische Flussfunktion nur ein Wert eingeht, wird die zur Stabilisierung des Verfahrens benötigte numerische Viskosität durch "Verschieben" der Interpolationskoeffizienten in die Aufwind-Richtung bewirkt.

Gewichtetes Knotenmittel

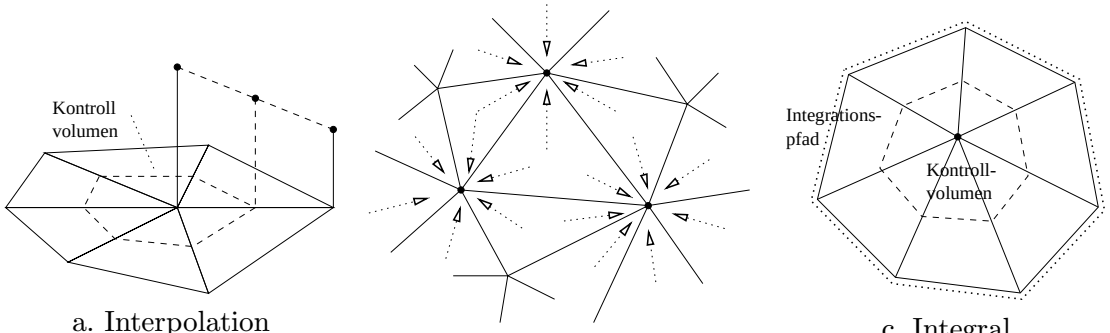
Für ein zell-zentriertes FV-Verfahren auf Tetraedergittern läßt sich ein Gradient einfach bestimmen, wenn Werte auf den Knoten bereitgestellt werden können. Ein solcher Knotenwert kann durch gewichtete Mittlung über alle an den Knoten angrenzenden Kontrollvolumen bestimmt werden, siehe Abbildung 4.1.b. Damit auf diesem Wege die notwendige Approximation zweiter Ordnung erreicht wird, also eine lineare Funktion exakt rekonstruiert werden kann, müssen die Gewichte geeignet gewählt werden, [64].

Mittlerer Gradient

Ein bei Finite-Volumen-Verfahren häufig angewendetes Prinzip der Gradientenberechnung ist die numerische Auswertung der Beziehung

$$\int_{\Omega} \mathrm{grad}(\varphi)\, \mathbf{d}\lambda = \int_{\partial\Omega} \varphi\, \mathbf{d}\sigma \tag{4.18}$$

Damit dieser Ansatz den Gradienten linearer Funktionen exakt ermittelt, muss Ω so gewählt werden, dass die Nachbarwerte genau an den Gaußpunkten der Oberflächenintegrale liegen und die benötigten Geometrieparameter (Gaußpunkte, Größe der Oberflächen, Größe der Masche) bekannt sind. Ist das zu den Kontrollvolumen des FV-Verfahrens duale Gitter zugänglich, kann Ω aus den Maschen des dualen Gitter aufgebaut werden, die den betrachteten Punkt und seine Nachbarn als Eckpunkte beinhalten, siehe Abbildung 4.1.c . Besonders einfach ist dies für kartesische Gitter [69] aufgrund der regelmäßigen Struktur des Gitters und für unstrukturierte Dreiecks- bzw. Tetraedergitter [61], deren duale Gitter leicht zugänglich sind. Für die hier eingesetzten polygonalen Maschen ist die Ermittlung des dualen Gitters so aufwändig, dass diese Technik in dieser Weise nicht einsetzbar ist.



a. Interpolation b. Knotenmittel c. Integral

Abbildung 4.1: Rekonstruktionsmethoden

Direkte Lösung des Interpolationsproblems

Um das gesuchte Polynom zu bestimmen, kann ein lineares Gleichungssystem für die gesuchten Koeffizienten des Polynoms aufgestellt werden, indem die Werte der Umgebung und ihre Orte eingesetzt werden [1, 7]. Stehen mehr Nachbarn zur Verfügung als Koeffizienten bestimmt werden sollen, können entweder Gleichungen weggelassen (etwa solche, die zu steilen Gradienten führen würden) oder ein Least-Squares Ansatz zur Lösung eingesetzt werden. Reicht die Anzahl der direkten Nachbarn nicht aus, werden weiter entfernte Kontrollvolumen mitberücksichtigt. Dieser Ansatz erlaubt auch Polynome höheren Grades zu bestimmen.

Gradienten auf beliebigen Nachbarschaften

Im Rahmen dieser Arbeit wurde der bereits in [24] vorgestellte Ansatz zur Gradientenberechnung im Kontrollvolumen Ω_j aus seiner Umgebung N_j verwendet:

$$M_j \quad = \quad \sum_{k \in N_j} \vec{w}_{jk} \cdot (\vec{x}_k - \vec{x}_j)^t \tag{4.19}$$

$$\vec{gr}_j = M_j^{-1} \cdot (\sum_{k \in N_j} \vec{w}_{jk} \cdot (\bar{\varphi}_k - \bar{\varphi}_j)) \tag{4.20}$$

$$\varphi_j(\vec{x}) = \bar{\varphi}_j + \langle \vec{x} - \vec{x}_j, \vec{gr}_j \rangle \tag{4.21}$$

Dieser Ansatz sowie die Wahl der Gewichtsvektoren $\vec{w}_{jk}$ wird im folgenden Abschnitt 4.3.3 diskutiert. Alle in dieser Arbeit vorgestellten Lösungen wurden unter Verwendung dieses Ansatzes berechnet.

4.3.3 Konsistenz der linearen Rekonstruktion

Für den in den Glgen. 4.19, 4.20 vorgestellten Rekonstruktionsansatz soll die konsistente Approximation einer linearen Funktion

$$\varphi(\vec{x}) = \varphi_0 + \langle \vec{x} - \vec{x}_0, \mathbf{d}\varphi \rangle$$

mit dem frei wählbaren Referenzpunkt $\vec{x}_0$ nachgewiesen werden. In der Umgebung $\bar{N}_j = N_j \cup \{j\}$ des Kontrollvolumens Ω_j ergeben sich die Schwerpunktswerte nach Glg. 4.8 als

$$\bar{\varphi}_k = \varphi_0 + \langle \vec{x}_k - \vec{x}_0, \mathbf{d}\varphi \rangle \, .$$

Zur Vereinfachung der Darstellung soll der Index j des betrachteten Kontrollvolumens im Folgenden weggelassen werden und wir erhalten mit Gewichtsvektoren $\vec{w}_k$ und der Schreibweise $\triangle\vec{x}_k = \vec{x}_k - \vec{x}_j$ für die Gewichtsmatrix M aus Glg. 4.19

$$M = \sum_{k \in N} \vec{w}_k \triangle \vec{x}_k^t \, .$$

Ist die Gewichtsmatrix invertierbar, so gilt für den approximierten Gradient

$$\begin{aligned} \vec{gr} &= M^{-1} \sum_{k \in N} \vec{w}_k (\bar{\varphi}_k - \bar{\varphi}_j) = M^{-1} \sum_{k \in N} \vec{w}_k \langle \triangle\vec{x}_k, \mathbf{d}\varphi \rangle \\ &= M^{-1} \left(\sum_{k \in N} \vec{w}_k \triangle \vec{x}_k^t \right) \cdot \mathbf{d}\varphi = \mathbf{d}\varphi \, . \end{aligned}$$

Eine lineare Lösung kann also aus den Mittelwerten exakt rekonstruiert werden, d. h. dass die Funktionswerte von zweiter Ordnung genau approximiert werden.

Die Invertierbarkeit der Gewichtsmatrix hängt von der Lage der Nachbarschaft $\triangle\vec{x}_k$ und der Wahl der Gewichtsvektoren $\vec{w}_k$ ab. Für die Wahl $\vec{w}_k = \triangle\vec{x}_k$ lässt sich zeigen, dass die Matrix M auf dem von der Nachbarschaft aufgespannten Teilraum $\mathrm{lin}\{\triangle\vec{x}_k\}_{k \in N}$ invertierbar ist. Die Behauptung leitet sich aus der Tatsache her, dass der Urbildraum einer Matrix in ihren Kern $\mathrm{Ker}(M)$ und das Bild $\mathrm{Im}(M^t)$ der dualen Abbildung zerfällt. Für die genannte Wahl der Gewichtsvektoren ist die Gewichtsmatrix symmetrisch, d. h. Bild- und Urbildraum zerfallen in Bild und Kern der Matrix. Liegt ein Vektor $\vec{x} \in \mathrm{lin}\{\triangle\vec{x}_k\}_k$ im Kern, so folgt

$$0 = \vec{x}^t M \vec{x} = \vec{x}^t (\sum_k \triangle\vec{x}_k \triangle\vec{x}_k^t)\vec{x} = \sum_k \langle \triangle\vec{x}_k, \vec{x} \rangle^2 \, ,$$

d. h. dieser Vektor muss auf allen Vektoren $\vec{x}_k$ senkrecht stehen und liegt somit gerade nicht in der linearen Hülle der $\vec{x}_k$. Damit ist auch die Summe in Glg. 4.20 im Bild enthalten und das Urbild kann bis auf einen Vektor aus dem Kern bestimmt werden. Da der Kern gerade die Richtungen enthält, in denen keine Nachbarn vorhanden sind, ist für die Bestimmung des Urbildes der Nullvektor geeignet.

4.3.4 Wahl der Gewichtsvektoren

Für die Wahl der Gewichtsvektoren wurden im Wesentlichen zwei verschiedene Optionen untersucht, nämlich die Wahl des zugehörigen Oberflächennormalenvektors $\vec{s}_{jk}$ und die Wahl des Abstandsvektors $\triangle\vec{x}_{jk} = \vec{x}_k - \vec{x}_j$.

Oberflächennormalenvektor

Die Wahl des Oberflächennormalenvektos führt zu einer Gradientenformel, die der Berechnung des Gradienten durch das Oberflächenintegral Glg. 4.18 ähnlich ist, ohne jedoch die Kenntnis einer Integrationsfläche vorauszusetzen. Dies macht den Einsatz auf Gittern mit beliebigen Nachbarschaften überhaupt erst möglich. Dafür geht der Vorteil der Berechnung des Gradienten durch eine Flussformulierung, vgl. Glg. 4.20, durch Multiplikation mit der Matrix M^{-1} aber verloren.

Aufgrund der in dieser Arbeit benutzten, vielseitigen Kontrollvolumen können benachbarte Kontrollvolumen auch mehrere gemeinsame Oberflächenstücke haben. Deshalb ist es wünschenswert, dass der Gradient unabhängig von der Aufteilung der Oberfläche in Teilstücke ist. Da der mittlere Oberflächennormalenvektor der gemeinsamen Oberfläche nur von der Randkurve dieser Fläche abhängt, vgl. Glg. A.4, ist die Gradientenformel invariant unter Aufteilungen dieser Fläche, wenn die Oberflächennormalenvektoren $\vec{s}_{jk}$ als Gewichtsvektoren benutzt werden.

Obwohl in diesem Fall die Invertierbarkeit der Matrix nicht gesichert ist, wurden Oberflächennormalen als Gewichtsvektoren mit Erfolg eingesetzt und haben sich als robust erwiesen.

Abstandsvektor

Eine andere, vorteilhafte Wahl ist die Bestimmung des Gewichtsvektors aus den Abstandsvektoren $\triangle\vec{x}_{jk}$, weil dadurch keine Richtungsinformation im Gradienten verloren geht. Grenzen mehrere Nachbarn an Flächen mit gleicher Oberflächennormalenrichtung, so wird der Versatz der Nachbarn in Tangentialrichtung durch Gewichtung mit den Abstandsvektoren korrekt bewertet.

Neben der Invertierbarkeit der Gewichtsmatrix führt diese Wahl der Gewichtsvektoren auf eine äquivalente Minimierungsaufgabe. Der resultierende Gradient minimiert das Funktional

$$F(\vec{g}) = \sum_{k \in N_j} \left(\bar{\varphi}_j + \langle \triangle\vec{x}_k, \vec{g}\rangle - \bar{\varphi}_k\right)^2 .$$

Dies ergibt sich aus der Untersuchung des Gradienten des Funktionals

$$\frac{\mathbf{d}F}{\mathbf{d}\vec{g}}(\vec{g}) = 2 \sum_{k \in N_j} \left(\bar{\varphi}_j + \langle \triangle\vec{x}_k, \vec{g}\rangle - \bar{\varphi}_k\right) \triangle\vec{x}_k ,$$

dessen Nullstelle gerade mit dem Gradienten übereinstimmt. Es besteht also ein Zusammenhang mit den von Barth [7] und den für gitterlose Diskretisierungen [42] vorgeschlagenen Techniken.

Für eine quadratische Ansatzfunktion φ ermittelt man für den Approximationsfehler der ersten Ableitung

$$\vec{gr}_j - \mathbf{d}\varphi(\vec{x}_j) = M^{-1} \cdot \sum_{k \in N_j} \triangle\vec{x}_{jk} \langle \triangle\vec{x}_{jk} \triangle\vec{x}_{jk}^t, \mathbf{d}^2\varphi \rangle$$

Liegen sich je zwei Schwerpunkte der Nachbarschaft gegenüber, d. h. $\forall k \in N_j \; \exists l \neq k$: $\triangle\vec{x}_{jk} = -\triangle\vec{x}_{jl}$, so verschwindet der Fehlerterm erster Ordnung. Obwohl diese Annahme für konkrete Rechenfälle unrealistisch ist, zeigt die Gradientenformel in dieser Form vorteilhafte Möglichkeiten der Fehlerauslöschung.

Für die in Kapitel 6 dargestellten Ergebnisse wurden die Abstandsvektoren skaliert mit dem Quadrat ihrer Norm ($||\triangle\vec{x}_k||^{-2}$) als Gewichtsvektoren verwendet.

Skalierung des Gewichtsvektors

Neben der Richtung des Gewichtsvektors kann auch seine Länge angepasst werden. Dies entspricht einer *relativen* Auf- oder Abwertung des Einflusses einzelner Nachbarn auf den zu ermittelnden Gradienten. Eine gemeinsame Skalierung aller Gewichtsvektoren hat keinen Einfluss auf die Approximation der ersten Ableitung, da die Formel invariant unter regulärer Transformation ist (siehe unten).

Da die Differenz der benachbarten Zustände in den Gradienten eingeht, erscheint es vernünftig, diese Differenz mit einem Faktor $||\triangle\vec{x}_{jk}||^p$ mit $p \leq -1$ abzuwerten. Für $p > -1$ werden die weiter entfernten Nachbarn aufgewertet, für $p = -1$ gehen nur die Steigungen in den Gradienten ein und für $p < -1$ wird der Einfluss weiter entfernter Nachbarn abgewertet.

Wird der Gewichtsvektor $\triangle\vec{x}_{jk}$ gewählt, enthält dieser selbst noch einmal den Abstand, sollte also skaliert werden, während die Normalenvektoren bereits eine natürliche Größe im Bezug aufeinander haben.

Invarianz unter regulärer Transformation

Die Definition des Gradienten, Glg. 4.20 zeigt, dass der ermittelte Gradient invariant unter einer Transformation der Gewichtsvektoren mit einer beliebigen invertierbaren Matrix $T \in R^{d \times d}$ ist.

$$\vec{gr} = \left(\sum_{k \in N} T\vec{w}_k \triangle\vec{x}_k^t \right)^{-1} \left(\sum_{k \in N} T\vec{w}_k (\bar{\varphi}_k - \bar{\varphi}_j) \right)$$

Konsistenz mit Finiten-Differenzen

Auf einem orthogonalen, äquidistanten, strukturierten Gitter erhält man bei Gewichtung mit Knotenabständen die bekannten, zentralen ersten Ableitungen wieder. Dieser Ansatz kann also auch als eine Verallgemeinerung der Finite-Differenzen-Technik aufgefasst werden.

4.4 Randbedingungen für die Euler-Gleichungen

Die Untersuchung von Systemen quasi-linearer partieller Differentialgleichungen erster und zweiter Ordnung und ihrer Randbedingungen ist eng verknüpft mit dem Konzept der Charakteristiken, vgl. [13, 26]. Charakteristiken sind Familien von (Hyper-)Flächen in Raum und Zeit, entlang denen bestimmte Eigenschaften der Lösung konstant sind oder bestimmte Ableitungen unstetig sein können.

Die instationären Euler-Gleichungen sind ein hyperbolisches System von Erhaltungsgleichungen erster Ordnung, welches auf einem beschränkten Gebiet mit geeigneten Anfangs- und Randbedingungen ("Cauchy-Problem") versehen werden muss. Die Analyse der Charakteristiken zeigt, dass für diese Differentialgleichung nicht an allen Rändern genausoviele Bedingungen ("physikalische Bedingungen", vgl. [25]) vorgegeben werden dürfen wie Gleichungen gegeben sind. Während die Zahl physikalischer Anfangsbedingungen mit der Zahl der Differentialgleichungen übereinstimmt, hängt die Zahl physikalischer Randbedingungen von den Eigenwerten der Funktionalmatrix

$$A(\varphi, \vec{s}) = \sum_{i=1\ldots d} s_i \cdot \frac{\partial F_i}{\partial \varphi}(\varphi) =: \sum_{i=1\ldots d} s_i \cdot A_i(\varphi)$$

ab und kann kleiner sein als die Zahl der Gleichungen.

Die Implementierung des Finite-Volumen-Verfahrens hingegen beruht auf der Bereitstellung eines vollen Satzes von Zustandsvariablen für die Berechnung der Rekonstruktion und der Flüsse. Die auf die Dimension des Zustandsvektors fehlenden Bedingungen werden als "numerische" Randbedingungen bezeichnet (vgl. [25]) und müssen kompatibel mit den Eigenschaften der Differentialgleichung und der Diskretisierung im Inneren des Rechengebietes gewählt werden.

4.4.1 Numerische Randbedingungen und Rekonstruktion

Wird in dem in Glg. 4.9 beschriebenen Finite-Volumen-Verfahren aus den umgebenden Schwerpunktswerten eine lineare Rekonstruktion oder eine Rekonstruktion höherer Ordnung ermittelt, so ist es wünschenswert, diese Rekonstruktion auch zur Bestimmung der numerischen Randbedingungen heranzuziehen. Da die Randwerte in die Bestimmung der Rekonstruktion eingehen und die numerischen Randbedingungen unter Verwendung der Rekonstruktion ermittelt werden sollen, entsteht in den Kontrollvolumen Ω_j, die an die Ränder des Rechengebietes $\partial\Omega_b$ grenzen, eine wechselseitige Abhängigkeit zwischen dem Randwert und der Rekonstruktion des inneren Kontrollvolumens.

Da ein Kontrollvolumen an viele Randseiten grenzen kann, kann die Forderung nach der Kompatibilität der Rekonstruktion φ_j mit den Randwerten $\bar{\varphi}_b$

$$\varphi_j(\vec{x}_b) = \bar{\varphi}_b \,, (\vec{x}_b \in \partial\Omega_b) \tag{4.22}$$

nicht erfüllt werden. Deshalb wird der Randzustand mit Hilfe der Rekonstruktion aus dem angrenzenden, inneren Kontrollvolumens an den Integrationspunkt der Wandseite extrapoliert und entsprechend der physikalischen Randbedingung modifiziert. Die wiederholte Bestimmung der Rekonstruktion und Extrapolation der Randwerte beschleunigt bei subsonischen Strömungen die Konvergenz gegen eine stationäre Lösung. Die in Kapitel 6 gezeigten Lösungen wurden mit drei Wiederholungen gewonnen.

4.4.2 Implementierung der Randbedingungen

Randbedingungen durch Extrapolation

Zur Implementierung der numerischen Randbedingungen werden besondere Rand-Kontrollvolumen in das Gitter eingeführt. Dadurch grenzen auch die Randoberflächen des Gitters an zwei Kontrollvolumen, so dass sich eine einheitliche Grundlage für die Berechnung der Flüsse und der Rekonstruktionen am Rand und im Inneren ergibt. In der Literatur finden sich eine Reihe verschiedener Bezeichnungen für diese Technik: die zusätzlichen Maschen werden als künstliche ("artificial"), Geister- ("ghost") oder Halo-Volumen bezeichnet.

Die Zustande für diese Randvolumen werden – je nach Art der Randbedingung – aus äußeren Vorgaben, Werten aus dem Rechengebiet oder einer Mischung bestimmt und an den Integrationspunkten der Randfläche lokalisiert. Insbesondere bedeutet dies, dass Bedingungen an die Ableitungen nur über die geeignete Wahl der Zustände implementiert werden können, d. h. dass Neumann-Randbedingungen in Dirichlet-Randbedingungen umgewandelt werden müssen. Dabei muss beachtet werden, dass die Vorgabe von Werten, die inkompatibel mit den auslaufenden Charakteristiken sind, zu Reflektionen führt, weil Wellen aus dem Inneren des Rechengebietes dieses nicht in der angemessenen Weise verlassen können.

Wandrandbedingungen

Für einen Wandrand vereinfacht sich die Flussfunktion deutlich, weil kein konvektiver Fluss durch die feste Wand stattfinden soll. Deshalb reduziert sich der Flussvektor auf die Druckkomponente des Impulsflusses

$$f(\bar{\varphi}_w, \vec{s}) = \begin{pmatrix} 0 \\ p_w \vec{s} \\ 0 \end{pmatrix}$$

Der Druck p_w am Rand wird aus dem Randzustand $\bar{\varphi}_w$ ermittelt. Der Randzustand φ_w wird durch Modifikation des aus dem angrenzenden, inneren Kontrollvolumen Ω_j rekonstruierten Zustands $\varphi_{jw} = \varphi_j(\vec{x}_w)$ am Integrationspunkt $\vec{x}_w$ der Randfläche $\partial\Omega_w$ gewonnen. Der Randzustand wird so bestimmt, dass

$$\langle \vec{v}_w, \vec{s} \rangle = 0 \tag{4.23}$$
$$c_w = c_{jw} \tag{4.24}$$
$$s_w = s_{jw} \tag{4.25}$$
$$H_w = H_{jw} \tag{4.26}$$

gilt. Dabei bezeichnet $\vec{v}$ den Geschwindigkeitsvektor, c die Schallgeschwindigkeit, s die Entropie, H die Totalenthalpie und $\vec{s}$ den Oberflächennormalenvektor.

Die Korrektur des Geschwindigkeitsvektors bereitet besonders bei starken Anströmungen der Randoberfläche Probleme. Dieser Fall tritt normalerweise am Anfang einer Simulation auf sehr groben Gittern auf. Starke Änderungen des Geschwindigkeitsrandwertes können über das Zeitschrittverfahren zu unphysikalischen Zuständen (z. B. negative Dichte, negativer Druck) führen. Deshalb wird die Norm der Geschwindigkeitskorrektur bezogen auf die Schallgeschwindigkeit begrenzt.

Fernfeld- und Überschall-Randbedingungen

An einem Fernfeld- oder Überschall-Einströmrand werden alle Zustandswerte fest vorgegeben. Zur Flussberechnung wird der genäherte Riemann-Löser wie im Inneren des Rechengebietes benutzt. Der Riemann-Löser sorgt dann für die korrekte Auswahl der Charakteristiken, so dass diese Technik auch für Unterschallbedingungen im Fernfeld benutzt werden kann. Deutlich sichtbare Probleme zeigt dieser Ansatz, wenn ein Schock den Fernfeldrand kreuzt.

An einem Überschall-Ausströmrand wird der aus dem angrenzenden inneren Kontrollvolumen rekonstruierte Zustand zur Flussberechnung herangezogen. Soll die gleiche Flussberechnung wie im Inneren des Gebietes benutzt werden, so kann das Randvolumen mit dem gleichen Wert besetzt werden. Aufgrund der Konsistenz des numerischen Flusses mit dem exakten Fluss entspricht dieses (aufwändigere) Vorgehen der Auswertung der gegebenen Flussfunktion.

4.5 Diskretisierung in der Zeit

Nach Diskretisierung der räumlichen Ableitungen in Glg. 4.2 verbleibt das Zeitintegral des Residuums

$$\int_{t_0}^{t_1} r_j(t)\,\mathbf{d}t\,,\ r_j(t) = \sum_{k\in N_j} f^*(\varphi_j(t,\vec{x}_{jk}),\varphi_k(t,\vec{x}_{jk}),\vec{s}_{jk})$$

in Glg. 4.9 und wird durch eine weitere Quadraturformel approximiert. Die getrennte Diskretisierung der räumlichen und zeitlichen Ableitungen wird als Semi-Diskretisierung oder "method of lines" bezeichnet.

4.5.1 Zeitschrittverfahren

Bezieht die Quadraturformel neben den bekannten Zuständen zur Zeit t_0 keine unbekannte Zustände der Zeit $t > t_0$ ein, so erhält man ein explizites, eventuell mehrstufiges Zeitschrittverfahren (Runge-Kutta-Verfahren). Das einfachste Verfahren dieser Klasse ist das einstufige, vorwärtsgerichtete Eulerverfahren, welches das Zeitintegral durch

$$\int_{t_0}^{t_1} r_j(t)\mathbf{d}t \approx (t_1 - t_0)\, r_j(t_0) \tag{4.27}$$

approximiert. Mit dieser Approximation ist der Zeitschritt durch eine CFL-Bedingung beschränkt, die die Stabilität garantiert. Diese Bedingung stellt sicher, dass die sich ausbreitenden Wellen die Umgebung des Kontrollvolumens innerhalb des betrachteten Zeitintervalls nicht verlassen.

Untersucht man die Stabilitätseigenschaften des beschriebenen Verfahrens für die skalare Konvektion, $\partial_t\varphi + \mathrm{div}(\varphi\vec{v})$ mit konstantem Geschwindigkeitsfeld $\vec{v}$ und konstanter Rekonstruktion, so kann die folgende Courant-Friedrichs-Levi-Bedingung für einen stabilen Zeitschritt abgeleitet werden:

$$\triangle t \cdot \frac{\sum_{k\in N_j} |\mu_{jk}|\,\lambda(\partial\Omega_{jk})}{\lambda(\Omega_j)} \leq 1\,. \tag{4.28}$$

Dabei ist der Eigenwert der Funktionalmatrix $A(\varphi_M, \vec{s}_{jk})$ aus dem Roe-Splitting, vgl. Glg. 4.11, mit μ_{jk} bezeichnet. Diese Bedingung hat sich auch für das System der Euler-Gleichungen als robust erwiesen, wobei $|\mu_{jk}|$ durch den betragsgrößten Eigenvektor der Funktionalmatrix ersetzt wird.

Variieren die Ausbreitungsgeschwindigkeiten der Wellen oder die Größen der Kontrollvolumen, so variiert auch die Größe des maximal zulässigen, lokalen Zeitschrittes. Auf einem adaptiv verfeinerten Gitter entsteht so eine globale Beschränkung für den Zeitschritt.

Mehrstufige Runge-Kutta-Verfahren erlauben es, die Ordnung der zeitlichen Diskretisierung zu erhöhen und/oder vergrößern den Stabilitätsbereich des Zeitschrittverfahrens.

4.5.2 Zeitschrittbeschränkung in mehreren Dimensionen

Für die in Abschnitt 4.6 diskutierten Variablensätze konnte eine Alternative zu Glg. 4.28 gefunden werden, welche auch die lineare Rekonstruktion mit einbezieht. Dazu betrachten wir die Approximation der homogenen, skalaren, hyperbolischen Erhaltungsgleichung

$$\partial_t \varphi + \operatorname{div} F(\varphi) = 0$$

mit dem stetig differenzierbaren Flusstensor $F : R \to R^d$, auf einem beschränkten Gebiet $\Omega \subset R^d$ durch das Finite-Volumen-Verfahren, Glg. 4.9, mit expliziter Zeitintegration, Glg. 4.27. Um einen Hinweis auf die Stabilitätseigenschaften des Verfahrens zu erhalten, sollen Abschätzungen für die Änderung des Zustandes

$$\triangle\bar{\varphi}_j^{n+1} := \bar{\varphi}_j^{n+1} - \bar{\varphi}_j^n$$

ermittelt werden. Die Beschränkung dieser Änderungen ist eine notwendige Bedingung für die Stabilität des Verfahrens.

Um die Darstellung zu vereinfachen, wird der Superskript n für den betrachteten Zeitschritt weggelassen. Zur Berechnung des numerischen Flusses wird das Flussdifferenzensplitting, Glg. 4.10, mit einem Integrationspunkt $\vec{x}_{jk}$ auf jeder Seite verwendet. Mit den Bezeichnungen

$$\begin{aligned}
\sigma_j &= \triangle t/\lambda_j \\
\varphi_l^{jk} &= \varphi_l(\vec{x}_{jk})\,,\ (k,l \in \bar{N}_j) \\
\triangle\varphi_{jk} &= \varphi_k^{jk} - \varphi_j^{jk} \\
\mu_{jk} &= \frac{\mathbf{d}f}{\mathbf{d}\varphi}(\varphi_M^{jk}, \vec{s}_{jk}) \\
\text{wobei: } \mu_{jk}\triangle\varphi_{jk} &= f(\varphi_k^{jk}, \vec{s}_{jk}) - f(\varphi_j^{jk}, \vec{s}_{jk}) \\
\mu_{jk}^- &= \frac{1}{2}(\mu_{jk} - |\mu_{jk}|) \le 0
\end{aligned}$$

erhält man aus Glg. 4.9 für das Zeitinterval $]t_n, t_{n+1}[$

$$\begin{aligned}
\triangle\bar{\varphi}_j^{n+1} &= -\sigma_j \sum_{k\in N_j} \frac{1}{2}\Big(f(\varphi_j^{jk}, \vec{s}_{jk}) + f(\varphi_k^{jk}, \vec{s}_{jk}) - |\mu_{jk}|\triangle\varphi_{jk}\Big) \\
&= -\sigma_j\Big(\sum_{k\in N_j} f(\varphi_j^{jk}, \vec{s}_{jk}) - \sum_{k\in N_j} -\frac{1}{2}\,(\mu_{jk} - |\mu_{jk}|)\triangle\varphi_{jk}\Big) \\
&= -\sigma_j\Big(\int_{\Omega_j} \operatorname{div} F(\varphi_j)\,\mathbf{d}\lambda - \sum_{k\in N_j} |\mu_{jk}^-|\triangle\varphi_{jk}\Big) \qquad (4.29)
\end{aligned}$$

Bezeichnet man die Differenz der Zustände mit $\triangle\bar{\varphi}_{jk} = \bar{\varphi}_k - \bar{\varphi}_j$ und deren Extrema mit

$$\triangle\bar{\varphi}_j^{\min} = \min_{k\in N_j} \triangle\bar{\varphi}_{jk} \quad \text{und} \quad \triangle\bar{\varphi}_j^{\max} = \max_{k\in N_j} \triangle\bar{\varphi}_{jk}$$

und findet $\alpha_j > 0$, $\beta_j > 0$ so, dass für die Rekonstruktion

$$\begin{gathered} \alpha_j \triangle\bar{\varphi}_j^{\min} \leq -\int_{\Omega_j} \operatorname{div} F(\varphi_j)\, \mathbf{d}\lambda \leq \alpha_j \triangle\bar{\varphi}_j^{\max} \\ \beta_j \triangle\bar{\varphi}_j^{\min} \leq \triangle\varphi_{jk} \leq \beta_j \triangle\bar{\varphi}_j^{\max} \end{gathered} \tag{4.30}$$

gilt, so erhält man aus Glg. 4.29

$$\sigma_j \left(\alpha_j + \beta_j \sum_{k\in N_j} |\mu_{jk}^-|\right) \triangle\bar{\varphi}_j^{\min} \leq \triangle\bar{\varphi}_j^{n+1} \leq \sigma_j \left(\alpha_j + \beta_j \sum_{k\in N_j} |\mu_{jk}^-|\right) \triangle\bar{\varphi}_j^{\max}\,.$$

Die Rekonstruktion kann so limitiert werden, dass Bedingung 4.30 für unabhängig von j und n beschränkte α_j, β_j erfüllt ist. Damit erhält man einen positiven Zeitschritt, der mit der CFL-Beschränkung

$$\frac{\triangle t}{\lambda_j} \left(\alpha_j + \beta_j \sum_{k\in N_j} |\mu_{jk}^-|\right) \leq 1 \quad \forall j\,, \tag{4.31}$$

zu einem Verfahren führt, das die Zustandsänderungen auf den Bereich einschränkt, der durch die Zustandsdifferenzen der Umgebung $\bar{N}_j$ gegeben ist. Damit ist die Erzeugung neuer Extrema ausgeschlossen.

Im Gegensatz zu den in der Literatur genannten Bedingungen [25, 7, 1] erzwingt Bedingung 4.30 an lokalen Extrema nicht, auf die konstante Rekonstruktion zurückzuschalten. An einem lokalen Maximum beispielsweise genügt es

$$\beta_j \sum_{k\in N_j} |\mu_{jk}^-| \triangle\bar{\varphi}_j^{\max} \leq \int_{\Omega_j} \operatorname{div} F(\varphi_j)\, \mathbf{d}\lambda$$

zu fordern, um die Zustandsänderung auf nicht-positive Werte einzuschränken, wenn

$$\triangle\bar{\varphi}_j^{\max} \leq \triangle\varphi_{jk} \leq 0$$

vorausgesetzt wird. Dies ist in glatten Bereichen eine plausible Annahme.

Insbesondere in mehreren Raumdimensionen oder für Systeme von Erhaltungsgleichungen kann die erste Bedingung eine wesentlich schwächere Einschränkung sein, dann nämlich, wenn die Klasse der Funktionen mit $\operatorname{div}(F(\varphi)) = 0$ nicht nur die konstanten Funktionen enthält. In diesem Fall kann eine Rekonstruktion aus dieser Klasse gewählt werden, ohne Einbußen in der Approximation hinnehmen zu müssen. Dies ist beispielsweise bei den Euler-Gleichungen möglich und wird in Abschnitt 4.6 diskutiert.

4.6 Wahl der Variablen für die Rekonstruktion

Durch Verbesserung der verwendeten Rekonstruktionstechniken konnte die Robustheit und Genauigkeit der Diskretisierung auch auf Gittern aufrecht erhalten werden, die nicht den

üblichen Qualitätskriterien genügen und deshalb mittels der in Kapitel 3.1 vorgestellten Technik automatisch erzeugt werden können. Ein zweiter Schwerpunkt der Bemühungen galt Rekonstruktionstechniken, die ohne Limiter auskommen.

Der Einfluss des Gitters auf das bisher beschriebene Verfahren konnte durch die Rekonstruktion in anderen Variablensätze als den konservativen Variablen deutlich reduziert werden. Mit der Änderung des Variablensatzes ist allerdings die konzeptuelle Frage verbunden, wie die Bedingung

$$\frac{1}{\lambda_j}\int_{\Omega_j}\varphi_j(\vec{x})\,\mathbf{d}\lambda = \bar{\varphi}_j$$

an die Rekonstruktion erfüllt werden kann. Solange nur lineare Rekonstruktionen betrachtet werden, kann man sich darauf zurückziehen, dass die Erhaltung des Mittels mit einem Fehler erster Ordnung gegeben ist. Für Rekonstruktionen höherer Ordnung bleibt nur der Hinweis darauf, dass keine Probleme sichtbar werden [2].

Durch die Wahl von Ansatzfunktionen, die die stationären Differentialgleichungen näherungsweise oder exakt erfüllen kann das Verfahren, ausser an Scherschichten, ohne Limiter benutzt werden.

4.6.1 Positivität der Variablen

Ein bekanntes Problem von Finite-Volumen-Verfahren für die Euler-Gleichungen, die Rekonstruktionen zur Erhöhung der räumlichen Genauigkeit einsetzen, ist die Kontrolle des Wertebereichs der Rekonstruktion. So führen negative Werte z. B. für die Dichte oder den Druck zum Zusammenbruch der Flussberechnung.

Aufgrund der quadratischen Abhängigkeit des Drucks von den konservativen Variablen werden häufig die primitiven Variablen ρ, $\vec{v}$ und p zur Rekonstruktion benutzt. In beiden Fällen ist es weiter notwendig Limiter dazu einzusetzen, unphysikalische Werte zu vermeiden [23, 1, 69, 64].

Die Ursache für die Schwierigkeit physikalisch sinnvolle Werte zu garantieren ist in der Formulierung der Rekonstruktion als polynomiale, also *reellwertige* Approximation zu suchen, die also auch negative Werte liefern kann. Eine mögliche Lösung des Problems, die im Rahmen dieser Arbeit entwickelt und intensiv getestet wurde, ist es, die Approximationsaufgabe nicht auf positive Variable, sondern auf reellwertige Variable anzuwenden. Dieses Vorgehen ist besonders wichtig, weil die skalaren Größen, die zur Beschreibung der Euler-Gleichungen herangezogen werden, also Dichte, Druck, innere Energie, Enthalpie, Machzahl, Schallgeschwindigkeit, usw. positive Zahlen sind.

4.6.2 Rekonstruktion reellwertiger Variablen

Wird zur Rekonstruktion beispielsweise der Variablensatz

$$\tilde{\varphi} = (\ln(\rho), \vec{v}, \ln(e))^t \tag{4.32}$$

unter Verwendung der inneren Energie $e = E - \frac{1}{2}||\vec{v}||^2$ herangezogen, so erhält man für jeden Vektor $\tilde{\varphi} \in R^m$ einen physikalisch zulässigen Zustandswert. Damit hat auch jedes beliebige Polynom in diesen Werten physikalisch zulässige Werte, d. h. dass eine Abschwächung des Gradienten (verbunden mit der Reduktion der Ordnung des Verfahrens)

nicht mehr notwendig ist, um die Flussberechnung durchführen zu können. Numerische Experimente mit diesem Variablensatz zeigten eine überzeugende Steigerung der Robustheit des gesamten Verfahrens.

Mit dieser Wahl der Rekonstruktion erhält man ein Verfahren, das auch in Gegenwart von Schocks ohne Limiter benutzt werden kann, ohne instabil zu werden. Wird kein Limiter benutzt treten allerdings leichte Oszillationen auf. Ein ähnliches Vorgehen wird in der Arbeit von Reichert [46] vorgeschlagen. Dort wird ein Hyperbelansatz auf charakteristischen Variablen vorgeschlagen. Dieser Ansatz ist äquivalent zu einer linearen Rekonstruktion, die durch den von van Albada [4] vorgeschlagenen Limiter begrenzt wird.

Auch der folgende Variablensatz,

$$\tilde{\varphi} = \left(s, \frac{\vec{v}}{c}, \ln(H) \right)^t \tag{4.33}$$

unter Verwendung der Schallgeschwindigkeit c und der Entropie $s = \ln(p/\rho^\gamma)$, führt für jeden Vektor $\tilde{\varphi} \in R^m$ auf einen physikalisch zulässigen Zustandswert. Die Skalierung des Geschwindigkeitsvektors erweist sich besonders an Schocks als vorteilhaft.

4.6.3 Limitierung des Gradienten

Soll das Finite-Volumen-Verfahren an Unstetigkeiten keine Oszillationen zeigen, so müssen die aus den Werten der Umgebung berechneten Gradienten limitiert werden. Dabei wird sichergestellt, dass die Rekonstruktion an den Integrationspunkten der Oberflächen, den maximalen Umgebungswert nicht überschreitet und den minimalen Umgebungswert nicht unterschreitet. Dadurch kann die Erzeugung neuer Extrema in der Lösung unterbunden werden. Gleichzeitig reduziert der Limiter jedoch die Ordnung der Diskretisierung an lokalen Extrema.

Ein weiteres Problem dieser Technik ist der mit der Limitierung verbundene "Schaltvorgang", der Oszillationen in der Lösungsentwicklung auslösen und so die Konvergenz gegen eine stationäre Lösung verhindern kann.

4.6.4 Lokale Lösungen als Ansatzfunktionen

In Zusammenhang mit der Stabilitätsbetrachtung des vorherigen Abschnittes 4.5 erscheint es vernünftig, lokale Lösungen der stationären Erhaltungsgleichung

$$\int_{\Omega_j} \operatorname{div} F(\varphi_j)\, \mathbf{d}\lambda = 0$$

als Ansatzfunktionen zu verwenden. So ermittelt man für die stationären Euler-Gleichungen durch Einsetzen der Massenerhaltungsgleichung in die Energieerhaltungsgleichung, dass die Enthalpie entlang einer Stromlinie konstant ist

$$\langle \nabla H, v \rangle = 0$$

Aus der Gleichung von Crocco,

$$0 = \nabla H - \frac{c^2}{\gamma(\gamma-1)s} \nabla s + \operatorname{rot}\vec{v} \times \vec{v} \tag{4.34}$$

können weitere Bedingungen abgeleitet werden. Da die betrachteten Strömungen isenthalp sind, kann der Gradient der Enthalpie zu Null gesetzt werden und es verbleibt eine Beziehung zwischen dem Entropiegradienten und der Rotation des Geschwindigkeitsfeldes, die nicht an Unstetigkeiten gilt. Aus Glg. 4.34 ermittelt man, dass die Entropie entlang der Stromlinie konstant ist,

$$\langle \nabla s, v \rangle = 0$$

Multipliziert man Glg. 4.34 mit einer Richtung, die senkrecht zur Geschwindigkeit ist, so erhält man schließlich eine Beziehung zwischen rot$\vec{v}$, $\vec{v}$, und ∇s. Ein erster Ansatz zur Näherung dieser Beziehung besteht darin, isentrope Bedingungen anzunehmen, d. h. $\nabla s = 0$. In diesem Fall ist das Geschwindigkeitsfeld rotationsfrei, d. h. rot $\vec{v} = 0$.

Diese Überlegungen führen auf die Wahl des Variablensatzes

$$\tilde{\varphi} = (s, \vec{v}, \ln(H))^t \ , \tag{4.35}$$

weil die diskutierten Gradienten mit Hilfe der Rekonstruktionstechniken aus Abschnitt 4.3.2 direkt ermittelt werden können. Durch Einsetzen der Bedingungen

$$\begin{aligned} \nabla(\ln H) = \frac{\nabla H}{H} &= 0 \\ \langle \nabla(s), \vec{v} \rangle &= 0 \\ \partial_r v_s \leftarrow \frac{1}{2}(\partial_r v_s + \partial_s v_r) \quad r, s &= 1...d \end{aligned} \tag{4.36}$$

erhält man eine Ansatzfunktion, welche die stationären Euler-Gleichungen näherungsweise, unter der Annahme isenthalper und isentroper Bedingungen sogar exakt erfüllt.

Während die Annahme isenthalper Bedingungen für die in Kapitel 6 vorgestellten Testfälle erfüllt ist, ist die Annahme isentroper Bedingungen nur in den subsonischen Fällen gerechtfertigt. Die unzutreffende Behandlung der Rotation in den trans- und supersonischen Testrechnungen führt jedoch nicht zu einem Verlust an Genauigkeit.

Obwohl der Variablensatz prinzipiell unphysikalische Zustände darstellen kann, sind solche in den Testrechnungen im wesentlichen nicht aufgetreten. Aufgrund der in der Anfangsphase einer Simulation auftretenden sehr groben Gitter ist jedoch eine Limitierung des Geschwindigkeitsgradienten notwendig. Ein negativer Einfluss des Geschwindigkeitslimiters auf die Konvergenz gegen die stationäre Lösung konnte nicht festgestellt werden. Eine Limitierung des Entropiegradienten (senkrecht zum Geschwindigkeitsfeld) unterdrückt Oszillationen der Entropie an Scherschichten, kann aber zu den bekannten Konvergenzschwierigkeiten führen.

Die in Kapitel 6 vorgestellten Rechnungen wurden mit dem Variablensatz 4.35 durchgeführt. Zur “Limitierung” der Rekonstruktion wurden die Beziehungen aus Formel 4.36 verwendet.

Kapitel 5

Implementierungsaspekte

Der erste Abschnitt beschreibt verschiedene wichtige Eigenschaften und Merkmale moderner numerischer Methoden und bewertet Vorzüge und Mängel verschiedener Programmiersprachen für die Implementierung solcher Verfahren. Daran schließt sich eine Beschreibung wesentlicher Merkmale des Programmentwurfs an: das Datenmodell zur Repräsentation des Gitters und der Lösung, die Organisation des dynamischen Speichers zur effizienten Unterstützung adaptiver Mechanismen, den Einsatz objekt-orientierter Techniken zur Strukturierung des Codes sowie Aspekte der Optimierung. Der letzte Abschnitt beschreibt die "Distributed-Memory"-Parallelisierung des Verfahrens mittels des in [9] entwickelten Konzeptes dynamischer, verteilter Graphen.

5.1 Anforderungen selbst-adaptiver Verfahren

Selbst-adaptive Verfahren führen innerhalb eines Rechenlaufs Verfeinerungen und Vergröberungen in der Diskretisierung durch. Die *schwankenden Speicheranforderungen* während der Laufzeit werfen die Frage nach der Unterstützung von Techniken dynamischer Speicherverwaltung auf.

Ein zweiter Aspekt adaptiver Verfahren ist das Management unstrukturierter Nachbarschaftsbeziehungen. Diese Fragestellung führt auf die Diskussion der verfügbaren Zeigerkonzepte und der Strategien zur Reduktion der Fehleranfälligkeit dynamischer Programmierung.

Für die Implementierung technisch-wissenschaftlicher Anwendungen spielen aber auch ganz traditionelle Datenstrukturen, nämlich Felder, eine herausragende Rolle. Felder stellen eine wichtige Abstraktion für diese Anwendungen dar, und ihre Implementierung durch den Compiler beeinflusst in hohem Maße die erreichbare Rechenleistung.

5.1.1 Dynamische Speicherverwaltung

Je stärker der Speicherbedarf einer Anwendung schwankt, desto hinderlicher wird eine à priori, d. h. zum Zeitpunkt der Compilierung festgelegte Dimensionierung des Speichers für die Formulierung der gewünschten Algorithmen. Damit erhebt sich die Frage nach den Möglichkeiten zur dynamischen Verwaltung des Arbeitsspeichers, d. h. nach einer

Schnittstelle zur Speicherverwaltung des Betriebssystems, der dynamischen Verwaltung des vom Betriebssystem bereits angeforderten Speichers und den Möglichkeiten, diesen Speicher in der Anwendung zu benutzen.

Statische Dimensionierung

Solange die Schwankungen des Speicherbedarfes vorhersehbar sind, können die benötigten Datenstrukturen zur Compilezeit dimensioniert werden. Um häufiges Recompilieren zu vermeiden, bietet sich eine großzügige Dimensionierung an, die zu einer Belastung von Systemresourcen führt, wenn das Betriebssystem keine virtuelle Speicherverwaltung bietet. Aber auch mit virtueller Speicherverwaltung wechseln sich durch die Überdimensionierung der Datenstrukturen benutzte und unbenutzte Bereiche des Adressraums ab und können so die Wirkung der virtuellen Speicherverwaltung des Betriebssystems beeinträchtigen. Ist der Compiler auf derartige Programmiertechniken nicht vorbereitet, kann dies zur Reservierung des Speichers im Executable führen, d. h. zu sehr großen Dateien und langen Ladezeiten beim Programmstart. Werden Rechenläufe über ein Batch-System gestartet, muss darüberhinaus eine entsprechend groß dimensionierte Jobklasse ausgewählt werden.

Kann der Speicher zur Laufzeit dimensioniert und angefordert werden, treten diese Schwierigkeiten nicht auf. Dazu wird eine Schnittstelle zur Speicherverwaltung des Betriebssystems benötigt sowie die Möglichkeit der Programmiersprache, den so erhaltenen Speicher in der Anwendung ansprechen zu können. Es ist ein wesentliches Defizit der für technisch-wissenschaftliche Anwendungen überwiegend benutzten Programmiersprache F77, dass sie die dazu benötigten Merkmale nicht aufweist.

Dynamische Dimensionierung

Aufgrund der Komplexität der Algorithmen und der Datenstrukturen werden häufig eine Reihe unterschiedlich dimensionierter Felder benötigt, deren Größe unter Umständen erst zur Laufzeit bekannt wird oder die nicht während der gesamten Laufzeit benötigt werden (temporärer Speicher). Dies führt auf die Programmiertechnik, einen großen Speicherblock zu reservieren und die benötigten Datenstrukturen auf das verfügbare Speicherreservoir abzubilden. Die für diese Abbildung benötigten Algorithmen sind in der Regel nicht anwendungsspezifisch und können vorteilhaft in einer Systembibliothek ("malloc"-Bibliothek) zusammengefasst werden, wieder vorausgesetzt, die für die Anwendung verwendete Programmiersprache unterstützt die Verwendung des so erhaltenen Speichers. Das Fehlen geeigneter Sprachunterstützung in F77 führt zu Implementierungen, die die ohnehin komplexen Strukturen moderner numerischer Verfahren (z. B. Multigrid) zusätzlich mit fehleranfälligen Zugriffsstrukturen auf den über Indexfelder verwalteten Speicher durchsetzen.

Rekursive Prozeduraufrufe

Ein anderer Aspekt dynamischer Speicherverwaltung betrifft die Bereitstellung von Speicherbereichen, die einer Funktion bzw. einem Unterprogramm zugeordnet sind. Für die Implementierung komplexer Algorithmen ist die Möglichkeit rekursiver Prozeduraufrufe von Vorteil, weil sie die Komplexität der Implementierung reduziert. Da in F77 der Speicherplatz für die lokalen Variablen statisch angelegt wird, d. h. einmal für die Funktion und

nicht einmal für den Prozeduraufruf, ist ein einfach zu handhabender Zugang zu rekursiven Prozeduraufrufen versperrt.

5.1.2 Indizes und Zeiger

Sowohl für strukturierte als auch unstrukturierte Gitter ergibt sich durch die Adaption des Gitters die Notwendigkeit, adaptierte und unveränderte Bereiche des Gitters miteinander in Bezug zu setzen. Dies führt unabhängig von der zu Grunde liegenden Gitterstruktur auf die Darstellung *unstrukturierter Nachbarschaftsbeziehungen*, entweder zwischen einzelnen Gittermaschen oder zwischen Gitterblöcken.

Die Darstellung von unstrukturierten Nachbarschaftsbeziehungen – in der Datenbank-Terminologie als 1:n-Beziehungen bekannt – führt unmittelbar auf das Konzept des Zeigers. Darunter soll eine Datenstruktur verstanden werden, mit der eine beliebige andere Datenstruktur der Anwendung angesprochen werden kann.

Sind die möglichen Ziele eines Zeigers in einem einzigen Feld untergebracht, so lässt sich ein Zeiger durch einen Feldnamen und einen Feldindex implementieren. Ist das Feld durch wiederholte Verfeinerung jedoch ausgeschöpft, bietet sich keine Möglichkeit, weitere Felder zur Speicherung zusätzlicher Ziele anzusprechen. Dann muss ein neues, größeres Feld angefordert werden und die vorhandenen Einträge müssen in das neue Feld umkopiert werden, bevor das Ausgangsfeld für einen anderen Zweck weiterverwendet werden kann. Dieses Vorgehen ist problematisch, weil zum Zeitpunkt des Umkopierens das Ausgangsfeld und das neue Feld gleichzeitig benötigt werden und sich der Speicherbedarf der Anwendung verdoppeln kann. Auch ist eine effiziente Weiterverwendung des Ausgangsfeldes durch Rückgabe an eine Speicherverwaltungsroutine nicht sichergestellt. Der zurückgegebene Speicherplatz wird keinesfalls an das Betriebssystem zurückgegeben, sondern für folgende Speicheranforderungen weiterverwendet. Durch wiederholtes Anfordern und Freigeben von Speicherbereichen entstehen immer mehr, immer kleinere Speicherblöcke, und der Verwaltungsaufwand kann um Größenordnungen steigen, vgl. [32].

Das Entfernen kleiner, unbenutzter Adressbereiche ("garbage collection") durch Zusammenschieben der Datenstrukturen ist eine aufwändige und äußerst komplexe Operation. In Programmiersprachen, deren Zeiger für die Anwendung keine Adresswerte sichtbar machen (z. B. F90, Java) kann dies durch die Speicherverwaltung des Laufzeitsystems geschehen (muss aber nicht!). Sind die Adresswerte der Zeiger sichtbar, besitzt nur die Anwendung selbst das notwendige Wissen für einen derartigen Vorgang. Hat man die Verwaltung der Speicherblöcke einer Bibliothek überlassen, ist dies praktisch unmöglich, weil kein Zugriff auf die Organisation der Speicherblöcke gewährt wird. Dies reduziert die Möglichkeiten zur Optimierung auf das Zusammenlegen benachbarter, freier Blöcke, wenn die benutzte Bibliothek dies unterstützt.

Stellt die verwendete Programmiersprache Zeiger zur Verfügung, so lässt sich die Problematik des Umkopierens komplett vermeiden, vgl. Abschnitt 5.2.2. Leider bieten die im Supercomputingbereich verfügbaren Programmiersprachen in diesem Bereich keine Lösungen, die die erforderliche Funktionalität mit effizienten und performanten Einsatzmöglichkeiten verbinden. Da der F77-Standard Zeiger nicht vorsieht, werden Zeiger dort nur in hersteller-spezifischen Erweiterungen unterstützt.

In der für ihre Zeiger bekannten Programmiersprache C ist das Zeigerkonzept so flexibel gehalten, dass die zur Optimierung notwendigen Aussagen über die Abwesenheit von

Datenabhängigkeiten vom Compiler in der Regel nicht aus dem Programmtext ermittelt werden können. Hierbei spielt auch eine Rolle, dass die Sprache C keine Feldabstraktion besitzt, sondern Felder als Zeiger behandelt. Dadurch wird der Unterschied zwischen einem Zeiger auf ein Feld und einem Zeiger auf ein Feldelement für den Compiler unsichtbar. So ist ohne externe Zusatzinformation aus dem Programmtext nicht ermittelbar, ob zwei Felder, die an eine Funktion übergeben werden, disjunkt sind oder nicht. Dadurch sind die Möglichkeiten zur automatischen Optimierung von Schleifen – insbesondere auf Vektorrechnern – stark eingeschränkt. Das Fehlen einer intuitiven Feldabstraktion macht die Benutzung von mehrdimensionalen Feldern, wie sie für technisch-wissenschaftliche Anwendungen typisch sind, fehleranfällig. Sowohl die Zeigerimplementierung als auch die fehlende Feldabstraktion wurden aufgrund von Kompatibilitätsanforderungen unverändert von C nach C++ übernommen.

Mit der Sprache F90 wurde der Versuch unternommen, eine für numerische Anwendungen geeignete Zeigerimplementierung einzuführen. Der wesentliche konzeptuelle Unterschied zwischen F90-Zeigern und ihren C-Pendants besteht darin, dass es in F90 keinen Zugriff auf die Adresswerte und damit keine Zeigerarithmetik gibt. Diese Zeiger können also nur zum Referenzieren des Zeigerziels benutzt werden. Zusätzlich wurden Attribute eingeführt, die Ziele von Zeigern kennzeichnen. Auch ist der Unterschied zwischen einem Zeiger auf ein Feldelement und einem Zeiger auf ein Feld klar ersichtlich. Damit bleiben die vorhandenen Strategien zur Schleifenoptimierung weiter anwendbar. Diese gut durchdachte Konzeption wird durch die Tatsache konterkariert, dass ein F90-Zeiger in den bisher getesteten F90-Compilern Zusatzinformationen enthält, die acht Speicherworte benötigt und die Rechenleistung drastisch beeinträchtigt, vgl. [32]. Diese Eigenschaft macht den F90-Zeiger für Anwendungen mit vielen Zeigern – z. B. für unstrukturierte Gitter – uninteressant.

5.1.3 Konstruktoren und Destruktoren

Die vorangegangene Diskussion von Zeigerimplementierungen hat einen wesentlichen Aspekt der dynamischen Programmierung mit Zeigern oder Indizes ausgespart, nämlich die Fehleranfälligkeit dieser Techniken. Besondere Aufmerksamkeit muss deshalb der korrekten Initialisierung, der Belegung und dem Ungültigmachen von Zeigern geschenkt werden.

Für die korrekte Funktion eines Programms ist es unabdingbar, dass die benutzten Zeiger auf gültige Datenstrukturen zeigen oder als ungültig markiert sind. Wird dies auch nur in einem Fall unterlassen, hängt die Funktion des Programms von mehr oder weniger zufälligen Zuständen des Arbeitsspeichers ab. Die Ursachen dadurch auftretender Programmfehler sind in begrenzter Zeit nur schlecht oder gar nicht aufzufinden, weil zwischen der Verursachung und dem Sichtbarwerden eines Fehlers viele Operationen ausgeführt worden sein können. Zerstört etwa eine Prozedur durch einen fehlbesetzten Zeiger Daten einer aufrufenden Prozedur, so wird diese Änderung erst nach der Rückkehr in die aufrufende Prozedur sichtbar. Ein ähnliches Problem entsteht, wenn ein Zugriff auf einen dynamisch allokierten Speicherbereich über den angeforderten Bereich hinausgreift. Nach wiederholten Speicheranforderungen und -freigaben ist die Verteilung der Datenstrukturen im Speicher so unübersichtlich, dass fehlerhafte Speicherzugriffe Fehler in Daten jedes beliebigen Teils eines Programms hervorrufen können.

Im Gegensatz zu fehlerhafter Besetzung “numerischer” Daten schaffen fehlbesetzte Zeiger durch Herstellung unbeabsichtigter, also auch unvermuteter Datenabhängigkeiten, äußerst

unübersichtliche Fehlersituationen. Dies kann zu Fehlern führen, die auch unter Verwendung von Debuggern oder speziell für diesen Zweck vorbereiteter Compiler nicht zuverlässig aufzuspüren sind.

Ein analoges Problem ergibt sich bei der Zerstörung von Datenstrukturen, auf die von anderer Stelle gezeigt wird. Gleichzeitig mit der Zerstörung der Zieldatenstruktur müssen alle diese Zeiger ungültig markiert werden, damit bei anderweitiger Verwendung des Speichers keine unerwünschten Datenabhängigkeiten entstehen.

Der Ansatz, die Besetzung von Zeigern bzw. von als Zeiger benutzten Indizes der Sorgfalt des Programmierers zu überlassen, wie es für F77, F90 und C der Fall ist, führt bei genügend komplexen Datenstrukturen leicht zur Einstellung eines Projektes aufgrund unauffindbarer Fehler. Es ist deshalb von allerhöchster Bedeutung, Mechanismen zu nutzen, die solche Besetzungen zuverlässig durchführen. Ein solches Konzept existiert in Form eines Objekt-Lebenszyklus in C++. Dabei garantiert der Compiler Aufrufe an sogenannte Konstruktoren, wenn ein Objekt erzeugt, kopiert oder zerstört wird. Diese Aufrufe werden beim Übersetzen des Programms vom Compiler an geeigneter Stelle eingesetzt und erlauben die zuverlässige Verwaltung verzeigerter Strukturen. Wie F90 kennt die Programmiersprache Java Datenstrukturen, dynamischen Speicher und Zeiger, ohne jedoch Speicheradressen für die Anwendung sichtbar zu machen. Zusätzlich wird das in F77, F90 und C fehlende Konstruktor-Konzept aus C++ übernommen, es fehlen aber Desktruktoren und eine effiziente Implementierung für Felder von Datenstrukturen.

5.1.4 Felder

Felder gehören zu den wenigen, grundlegenden Datenstrukturen, die sich auf Rechenanlagen effizient implementieren lassen. Sie sind darüberhinaus mit den grundlegenden Konzepten der linearen Algebra verwandt und gehören so zum Handwerkszeug technisch-wissenschaftlicher Programmierung. Insbesondere im Bereich mehrdimensionaler Felder unterscheiden sich die in Fortran (F77 bzw. F90) und C/C++ bereitgestellten Feldabstraktionen deutlich. Da in C/C++ Felder im eigentlichen Sinne gar nicht vorhanden sind, sondern als abkürzende Schreibweise für Zeiger definiert sind, bereitet die Verwendung variabel dimensionierter, mehrdimensionaler Felder Schwierigkeiten. So fehlt es beispielsweise an der Möglichkeit, solche Felder mit ihren Dimensionen in Prozeduren zu übergeben, oder solche Felder als lokale Variable zu deklarieren, ohne Zuflucht zur dynamischen Speicherverwaltung nehmen zu müssen. Die Seiteneffekte, die im Hinblick auf die Programmkomplexität und die Rechenleistung von der dynamischen Speicherverwaltung ausgehen, lassen die über 30 Jahre alte Feldabstraktion aus Fortran äußerst modern erscheinen.

Ein zweiter Aspekt der Anwendung von Feldern in technisch-wissenschaftlichen Anwendungen betrifft die Abbildung mehrdimensionaler Felder in den Speicher. Für die Anwendung von besonderer Bedeutung sind Felder von Datenstrukturen. Da F77 keine Datenstrukturen kennt, werden zu diesem Zweck zweidimensionale Felder mit einer großen (ersten) und einer kleinen (zweiten) Dimension benutzt, wobei die kleine Dimension die Komponenten der Datenstruktur enthält. Bei dieser Organisation des Feldes liegen alle Daten *einer* Komponente sequenziell im Speicher, d. h. die Daten liegen als Struktur von Feldern im Speicher, nicht als Feld von Strukturen. Diese Organisation ist besonders für Cache-basierte Rechnerarchitekturen von Vorteil, weil die in Schleifenkörpern angesprochenen Komponenten sequenziell im Speicher liegen, die aus dem Hauptspeicher geladenen

Cache-Lines also optimal ausgenutzt werden können. Zusätzlich muss beachtet werden, dass die aufeinanderfolgenden Felder so im Speicher lokalisiert sind, dass beim Zugriff auf verschiedene Komponenten zum gleichen ersten Index keine Cache-Konflikte entstehen.

In Programmiersprachen, die Datenstrukturen kennen, bietet sich dagegen die komplementäre Organisation an, also die Organisation als Feld von Strukturen, weil der Compiler auf Ebene der Datenstruktur sinnvolle Abstraktionen einfach unterstützen kann. Besonders deutlich wird dies, wenn die notwendige Datenstruktur selbst aus einer Hierarchie von Abstraktionen aufgebaut ist. Für jede Schicht können die notwendigen Operationen als Methoden auf einem sequenziell im Speicher lokalisierten Objekt implementiert werden. Betrachtet man den für eine Komponente vergebenen Namen als Feldindex, so wird deutlich, dass die Indizierung in diesem Fall umgekehrt verläuft.

Für das Software-Design liegt der Entwurf als Feld von Datenstrukturen dem zu lösenden Problem am nächsten und führt zu klar voneinander abgegrenzten Abstraktionen mit klaren Schnittstellen und übersichtlicher Struktur des Datenflusses. Diesen Vorteilen steht das Risiko verschwendeter Memorybandbreite gegenüber, wenn ein ungünstig verteilter Auszug der Datenstruktur benutzt wird. Ob dies der Fall ist, läßt sich im konkreten Fall jedoch nur schwierig ermitteln. Es muss allerdings betont werden, dass zwischen dem Layout der Datenstrukturen im Speicher und den Anforderungen des Software-Designs kein zwingender Zusammenhang besteht. Eine geeignete Programmiersprache und/oder einen geeigneten Compiler vorausgesetzt ließen sich beide Vorteile nutzen. Auch in dieser Hinsicht greift der Entwurf von F90 zu kurz. Zwar wurde dem Software-Entwickler der Zugriff auf das Speicherlayout durch das Design der Zeiger und die Definition der Datenstrukturen ohne Layout entzogen und so die Voraussetzungen geschaffen. Da das Speicherlayout aber den Compilerbauern als Option überlassen wurde, finden sich in gegenwärtigen Implementierungen keine Ansätze für eine solche Lösung. In C und C++ ist diese Thematik aufgrund der fehlenden Feldabstraktion gar nicht diskutierbar. Zwar bieten sie genügend Flexibilität, einen solchen Wechsel des Layouts zu implementieren, für den Compiler würden sich die Zugriffsmechanismen aber so kompliziert darstellen, dass keine realistische Aussicht auf eine erfolgreiche Schleifenoptimierung, z. B. Vektorisierung, besteht.

5.2 Programmentwurf

Eine Finite-Volumen-Diskretisierung läßt sich unabhängig von der Dimension des Raumes aus den Begriffen Kontrollvolumen, Oberfläche, Schwerpunkt und Integrationspunkt aufbauen. Ein wichtiges Entwurfsziel war deshalb die Unabhängigkeit der Implementierung von der Raumdimension. Ein weiterer Entwurfsschwerpunkt ist die effiziente Unterstützung adaptiver Verfahren auf Höchstleistungsrechnern. Die Implementierung des Verfahrens in C++ erlaubt objekt-orientierte Konzepte in den Entwurf des Codes einzubringen, die wesentlich zur Stabilität des Codes beitragen. Abschließend sollen Aspekte der Optimierung diskutiert werden.

5.2.1 Gitterobjekte

Die Gitterobjekte (Knoten, Kanten, Facetten, Zellen, Oberflächen und Kontrollvolumen) werden aus einem Schichtenmodell aufgebaut.

Auf der untersten Ebene liegen die Schnittstellen zur dynamischen Speicherverwaltung und dem Management der verteilten Datenstrukturen. Die dynamische Speicherverwaltung stellt über ihre eigentliche Funktion hinaus Objektindizes zur Verfügung, die eine effiziente Linearisierung der Objektverweise erlauben. Die darüberliegende Schicht stellt Eigenschaften zur Verfügung, die allen Gitterobjekten gemeinsam sind. Darunter fallen beispielsweise manipulierbare Flags und ein Schlüsselfeld, um temporär benötigte Daten mit einem Objekt zu assoziieren. Die nächste Ebene stellt die geometrischen Eigenschaften der Gitterobjekte (Größe, Schwerpunkt, Richtung) sowie die Konnektivitätsrelationen zur Verfügung. Diese Eigenschaften sind im Wesentlichen für alle Objekttypen gleich, weisen aber geringfügige Unterschiede auf. So enthalten beispielsweise Knoten keine Abwärtskonnektivität und Zellen keine Aufwärtskonnektivität. Die oberste Schicht dient der Implementierung des Finite-Volumen-Verfahrens und enthält die dazu notwendigen Zustandsvariablen und Methoden.

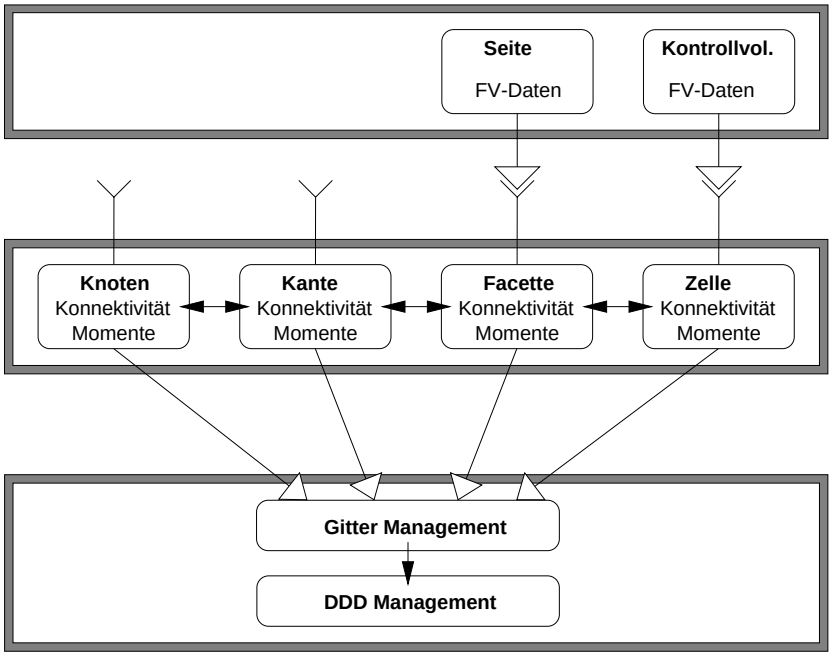


Abbildung 5.1: Datenmodell mit drei Schichten für Speichermanagement, Geometrie und Diskretisierung (von unten nach oben)

Die verschiedenen Schichten sind durch Vererbung aufeinander aufgesetzt. Die Datenstruktur kann zur Compilezeit auf die gewünschte Raumdimension angepasst werden. Für eine 3D-Anwendung werden Kontrollvolumen von Zellen und Oberflächen von Facetten abgeleitet, für eine 2D-Anwendung dagegen von Facetten und Kanten. Die Unabhängigkeit der Implementierung von der Raumdimension wird schließlich dadurch erreicht, dass das gesamte numerische Verfahren im Wesentlichen in den Begriffen des Kontrollvolumens und seinen Oberflächen formuliert werden kann. Die darüberhinaus benötigten dimensionsabhängigen geometrischen Konzepte werden von allen Gitterobjekten in gleicher Weise implementiert und sind deshalb ebenfalls unabhängig von der Dimension verfügbar. Abbildung 5.1 veranschaulicht diesen Vorgang durch “Steckverbindungen” zwischen der geometrischen und numerischen Schicht des Datenmodells.

Die objekt-orientierte Implementierung unterstützt die klare Abgrenzung der Schichten

voneinander und erlaubt komplexe Funktionalität hinter einfach zu handhabenden Schnittstellen zu verbergen. So lassen sich an vorhandenem Code auch mit wenig Programmiererfahrung einfach Änderungen vornehmen.

Der Umfang des Modells mit vier Schichten für vier Objekttypen verteilt das gesamte Datenmodell aber über mehrere Verzeichnisse und zahllose Dateien und erschwert so das Verständnis für den Zusammenhang zwischen Details und dem Modell als Ganzem. Hinzu kommt, dass die Struktur des Datenmodells mit ähnlichen, aber nicht gleichen Datentypen zum breitgestreuten Einsatz des Präprozessors führt. Statt ähnlichen Code mehrfach auszuschreiben, wird der Code parametrisiert (sog. “templates”). Andernfalls ist es nahezu unmöglich sicherzustellen, dass bereits gefundene Fehler im Code einer Klasse gleichzeitig auch in den übrigen Klassen korrigiert werden. Wird ein Template geändert, so ändern sich damit die Implementierungen aller beteiligten Datentypen. Diese Programmiertechnik reduziert den Umfang des betroffenen Quellcodes im vorliegenden Fall um den Faktor vier und trägt damit zur Reduktion der Komplexität bei, weil Unterschiede und Gemeinsamkeiten in der Formulierung des Codes deutlich zum Ausdruck kommen. Diese Verdichtung des Quellcodes bewirkt allerdings auch eine deutlich verschlechterte Lesbarkeit, solange die zu Grunde liegenden Abstraktionen nicht klargeworden sind.

5.2.2 Speicherorganisation

Die hohen Anforderungen an die Adaptivität im betrachteten Verfahren erfordern schnelle Verfahren für die Allokierung und Freigabe von dynamischem Speicher. Ein Geschwindigkeitsvorteil einer eigenen Implementierung gegenüber der Systembibliothek kann schon deshalb erwartet werden, weil Nutzen aus der gleichmäßigen Struktur der Gitterobjekte gezogen werden kann.

Für jeden Objekttyp wird ein eigene Speicherverwaltung bereitgestellt, damit jeweils nur Objekte gleicher Größe behandelt werden müssen. Die Gitterobjekte werden in Feldern geeigneter Länge organisiert, welche untereinander durch eine verkettete Liste verbunden sind. Das Auffinden freier Plätze in den Feldern kann beispielsweise über Bitfelder und die Aufbewahrung von Informationen über den letzten Zugriff (“round-robin”-Strategie) effizient organisiert werden, so dass nur in wenigen Fällen die verkettete Liste durchlaufen werden muss. Dabei muss darauf geachtet werden, dass das zuletzt angelegte Feld, welches am wahrscheinlichsten noch freien Platz enthält, als Erstes durchsucht wird.

Ein ähnlicher Ansatz wird zur Speicherung der Konnektivität verwendet. Da es für die verwendete Gitterstruktur notwendig ist, Objekte mit beliebig vielen Oberflächen darzustellen, wird eine effiziente Strategie zur Speicherung der Relationen benötigt. Ohne spezielle Vorkehrungen entsteht, wie bereits in Abschnitt 5.1.2 dargestellt, die Schwierigkeit vorhandene Relationenlisten bei Bedarf zu verlängern. Deshalb werden die Relationen in Blöcken typischer Größe allokiert und durch eine verkettete Liste untereinander verbunden. Auf diese Weise kann eine vorhandene Liste jederzeit durch Anfügen neuer Blöcke erweitert werden, ohne dass vorhandene Einträge umkopiert werden müssten. Gleichzeitig ist dadurch die Wiederverwendbarkeit freigegebener Blöcke für nachfolgende Speicheranforderungen sichergestellt.

5.2.3 Schleifen und Iteratoren

Das Finite-Volumen-Verfahren ist aus Schleifen über die Oberflächen bzw. die Kontrollvolumen des Gitters aufgebaut. So wird beispielsweise die Flussberechnung in einer Schleife über alle Oberflächen ausgeführt, die Flussbilanz und die Rekonstruktion in einer Schleife über alle Kontrollvolumen ermittelt.

Die typspezifische Speicherverwaltung lässt sich zur Formulierung der notwendigen Schleifen einsetzen. Sie können unter Ausnutzung dieser Organisation mittels zweier verschachtelter Schleifen formuliert werden. Die äußere Schleife läuft dabei über die verkettete Liste und die innere Schleife über das Feld, dessen Länge auf die verwendete Rechnerarchitektur angepasst wird.

Da diese Schleifen sehr häufig im Programm auftauchen, kann der Quellcode durch Verwendung einer Schleifenabstraktion, den sog. Iteratoren, wesentlich vereinfacht werden, [58, 22]. Ein Iterator ist ein abstrakter Datentyp, der den sequenziellen Zugriff auf die einzelnen Gitterobjekte in Form einer einfachen Zählschleife darstellt. Die Komplexität der zu Grunde liegenden Speicherorganisation wird dadurch hinter der Schnittstelle des Iterators verborgen. Die dadurch bewirkte Vereinfachung des Quellcodes erlaubt die fehlerfreie Formulierung der Schleifen unabhängig von der Komplexität der Speicherorganisation. Iteratoren bilden eine Schnittstelle zur Speicherverwaltung, die in der Implementierung der Systembibliotheken fehlt.

Die für das Verfahren typischen Schleifenkörper enthalten Zugriffe auf die Konnektivitätsrelationen. So werden beispielsweise für die Flussberechnung auf einer Oberfläche die Rekonstruktionen aus den angrenzenden Kontrollvolumen benötigt, für die Flussbilanzierung die Flüsse aus den begrenzenden Oberflächen oder für die Gradientenbildung die Zustände benachbarter Kontrollvolumen. Für den Zugriff auf die Relationen werden wiederum Iteratoren bereitgestellt, die einen kontrollierten Zugriff auf benachbarte Gitterobjekte sicherstellen.

Da die Flussberechnung den überwiegenden Anteil der gesamten Rechenzeit in Anspruch nimmt, wird für den dabei notwendigen Zugriff von Oberflächen auf die angrenzenden Kontrollvolumen ein eigener Mechanismus eingesetzt, der die Tatsache ausnutzt, dass immer zwei Kontrollvolumen angrenzen. Dabei stellt sich die Schleife auch für den Compiler als einfache Zählschleife dar und erlaubt so ihre Vektorisierung.

Die abstrakte Struktur des Schleifenkörpers besteht aus einem Lesevorgang ("Gather"-Operation) auf die über die Konnektivitätsrelation benachbarten Gitterobjekte und einer abschließenden Veränderung des betrachteten Gitterobjektes selbst. Dadurch entstehen keine Datenabhängigkeiten zwischen Lese- und Schreibvorgängen. Folgt die Implementierung diesem Konzept, so muss die Konnektivität zwischen Kontrollvolumen und ihren Oberflächen kommutativ ausgeführt werden, und für die Ergebnisse einer Schleifeniteration muss jeweils ein Speicherplatz im betrachteten Objekt vorgesehen werden. Der höhere Bedarf an Speicherplatz geht dabei mit erhöhter Flexibilität und vereinfachter Struktur des Quellcodes einher. Diesem Aspekt wurde hier höheres Gewicht beigemessen, da die notwendigen Algorithmen aufgrund der neuen Gitterstruktur nicht von vorneherein klar waren.

5.2.4 Module

Die für objekt-orientierte Programmierung typische Vorgehensweise, Funktionen als Methoden bestimmten Klassen zuzuordnen, erweist sich im Zusammenhang mit dem vielschichtigen Datenmodell und den Abstraktionen, die von der dynamischen Speicherverwaltung bereitgestellt werden, als unpraktisch. Dies soll am Beispiel der Flussberechnung erläutert werden.

Die Funktion zur Berechnung des Flusses würde in natürlicher Weise der Klasse zugeordnet werden, die die Oberfläche implementiert. Die Schleife über die von der dynamischen Speicherverwaltung in einem Feld gespeicherten Oberflächen würde der Feld-Klasse als Methode zugeordnet, während die verkettete Liste der Felder wiederum ihrer Klasse zugeordnet würde. Für die gesamte Operation würde in der Klasse, die das Gitter implementiert, ebenfalls eine Methode als Schnittstelle vorgesehen, die sich dort zusammen mit einer großen Zahl ähnlich entstandener Methoden wiederfindet. Die Abstraktionen des numerischen Verfahrens – etwa das Zeitschrittverfahren, die Gradientenberechnung oder die Gitterteilung – treten in der Implementierung als solche nicht auf.

Die Zuordnung der Methoden zu Objektklassen erweist sich deshalb als kontraproduktiv für die praktische Durchführung der Implementierung, die sich notwendigerweise an den Funktionseinheiten des numerischen Verfahrens orientiert. Es entstehen Abhängigkeiten der Klassendeklarationen quer durch alle Klassen der Implementierung und den gesamten Quellcode, die bei jeder relevanten Änderung zu vollständiger Recompilation führen. Bei genauerer Betrachtung stellen die so verteilten Methoden gar keine Schnittstellen im Sinne des objekt-orientierten Entwurfs dar. Eine isolierte Flussmethode in der Schnittstelle der Oberflächenklasse erfüllt das Entwurfsziel der gekapselten Abstraktion nicht. Im Gegenteil, Implementierungsdetails der Diskretisierung werden dadurch in den öffentlichen Schnittstellen der Klassen sichtbar.

Um dies zu vermeiden, wurden die entsprechenden Funktionen in separaten Klassen zusammengefasst, die die Abstraktionen des Iterationsverfahrens implementieren. In der Regel enthalten sie nur eine öffentliche Methode, z. B. eine Methode zur Durchführung der Flussberechnung, und kapseln so die Details der Implementierung. Außer einer Referenz auf das zu bearbeitende Gitter und eventuell von mehreren Methoden der Klasse gemeinsam benutzte Konfigurationsparameter enthalten diese Klassen keine eigenen Daten und legen so die Bezeichnung Modul nahe.

Die Beschreibung dieses Entwurfsmusters ist leider nicht Gegenstand der Standardtexte zum objekt-orientierten Entwurf, [10, 51, 65], die zwar explizite Anleitung zum Verteilen der Funktionen auf Klassen geben, sich zur Thematik von modularen Strukturen aber nicht äußern. Dieses Muster ist dem “block-structuring idiom”, vgl. [12], und dem “visitor pattern”, vgl. [22] verwandt.

5.2.5 Vektorisierung

Die Organisation der Speicherverwaltung mittels typspezifischer Felder und die Formulierung der Schleifen ohne Datenabhängigkeiten wurde motiviert durch den Wunsch, vektorisierbaren Quellcode zu erhalten. Praktisch scheitert die Vektorisierung jedoch am Entwicklungszustand der verfügbaren vektorisierenden Compiler und der fehlenden Bereitschaft

der Rechnerhersteller, diese weiterzuentwickeln. Der gegenwärtige Stand der Technik erlaubt wenig mehr als die Vektorisierung von F77-Code beziehungsweise deren äquivalente Formulierung in F90, C oder C++. Sobald jedoch Strukturen oder Zeiger ernsthaft verwendet werden, scheitert die Vektorisierung.

Dieser Zustand hat die folgenden Ursachen: Obwohl das Konzept der Datenstrukturen nun schon in das vierte Jahrzehnt geht, werden Operationen auf Datenstrukturen auch in Compilern für skalare Rechnerarchitekturen bis jetzt nicht unterstützt. Selbst die einfachsten Techniken, etwa die Elimination überflüssiger Strukturvariablen oder Strukturzuweisungen sind nicht vorhanden. Der zweite Aspekt besteht in der für die automatische Analyse von Datenabhängigkeiten fatalen Spezifikation der C-Zeiger. Durch die zu freizügige Spezifikation der Zeiger ohne Einschränkungen an mögliche Ziele und Typumwandlungen sind auch abwegige Konstruktionen syntaktisch möglich und erzwingen höchste Vorsicht bei Abhängigkeitsanalysen durch den Compiler. Gleichzeitig fehlen in der Regel Möglichkeiten, dem Compiler eine "vernünftige" Verwendung von Zeigern durch besondere Deklarationen zu garantieren (z. B. Cray "restrict"-Zeiger). Ein weiteres Problem der Optimierung – insbesondere für objekt-orientierte Programmiersprachen und die Vektorisierung – sind die von Compilern bereitgestellten Möglichkeiten des Funktions-"Inlining". Diese gehen in der Regel nicht über die Entfernung des Funktionsaufrufs hinaus. Selbst der für die Argumentübergabe bzw. für die Rückgabe von Funktionswerten benutzte Code findet sich unverändert im Assemblercode wieder. Die dabei vom Compiler eingefügten Strukturzuweisungen oder Zeiger verhindern anschließend jede weitere Optimierung oder Vektorisierung. In diesem Bereich heben sich der Gnu-Compiler und der KAI-Compiler von den Compilern der Hardwarehersteller sehr positiv ab.

Ein weiteres Problem für die Optimierung ist die nahezu ideologische Konzentration auf automatische, d. h. vom Compiler aus dem Quellcode ermittelbare Optimierungen. Die in den hier diskutierten Programmiersprachen enthaltenen Designprobleme können so nicht durch geeignete Direktiven überbrückt werden. Da die auf den aktuellen Rechnerarchitekturen relevanten Optimierungen auf der Analyse von Datenabhängigkeiten beruhen, werden Algorithmen, die diese Abhängigkeiten nicht durch einfache Indexzugriffe ausdrücken, von der Optimierung ausgeschlossen.

5.3 Parallelisierung

Ausgehend von einer Charakterisierung paralleler Rechnerarchitekturen wird die Domain-Decomposition-Parallelisierung für eine Distributed-Memory-Architektur beschrieben. Sie setzt auf dem Konzept eines dynamischen, verteilten Graphen auf, welches in der Arbeit von Klaus Birken, [9] formuliert und implementiert wurde. Die entwickelten Strategien zur Organisation der Überlappungszone der verteilten Gitter und zur parallelen Gitteradaption werden dargestellt.

5.3.1 Rechnerarchitekturen

Die gegenwärtig verfügbaren Parallelrechner lassen sich nach der Organisation des Hauptspeichers klassifizieren in Architekturen mit global adressierbaren und gleichmäßig erreichbarem Speicher (Shared-Memory), Architekturen mit verteiltem, global adressierbarem

Speicher und adressabhängiger Zugriffszeit (Distributed-Shared-Memory) und Architekturen mit verteiltem, nur lokal adressierbarem Speicher (Distributed Memory). Diese Klassifikation erscheint sinnvoll, weil das Speichersystem die einsetzbaren Programmiermodelle und die Rechenleistung der Architektur bestimmt.

Shared-Memory-Architekturen entstanden aus Vektorrechnern, die zuerst durch Einsatz mehrerer Prozessoren den Durchsatz sequenzieller Anwendungen erhöhten und schließlich zu Parallelrechnern ausgebaut wurden. Ihr wesentliches Merkmal sind hocheffiziente Speicherarchitekturen, die hohe Anforderungen an den Datendurchsatz befriedigen können. Da die nutzbare Rechenleistung moderner Rechnerarchitekturen nicht durch die verfügbare Kapazität an Rechenoperationen, sondern durch die verfügbare Bandbreite in den Speicher begrenzt wird, haben sich diese Architekturen im Produktionsbereich fest etabliert. Da der Skalierung dieser Technik im Bezug auf die Zahl der effizient nutzbaren Prozessoren und die Größe des Speichers Grenzen gesetzt sind, wurden Architekturen mit verteiltem Speicher entwickelt, deren Prozessortechnologie aus dem Workstation-Bereich übernommen wurde. Die Verbindung zwischen den einzelnen Prozessoren wird mittels eines Kommunikationsnetzwerkes durch explizites Versenden von Nachrichten hergestellt und bleibt der Anwendung überlassen ("message passing"). Die Notwendigkeit, die Adressierung von nicht-lokalem Speicher und die Erhaltung der Konsistenz von Datenkopien durch einen explizit zu programmierenden und anwendungsspezifischen Nachrichtenmechanismus zu implementieren, führt zu äußerst komplexen Programmen und hohem Aufwand für die Parallelisierung. Deshalb stellen Distributed-Shared-Memory-Architekturen auf einem verteilten Speichersystem einen einfach zu programmierenden, konsistenten, globalen Adressraum zur Verfügung. Das Konsistenzprotokoll begrenzt den Einsatz dieser Technik bisher auf mittlere Prozessorzahlen. Ein hersteller-übergreifendes Programmiermodell ("OpenMP") befindet sich gerade in der Einführung.

Am Ausgangspunkt der Parallelisierung stand die mangelnde Unterstützung für vektorisierende C bzw. C++ Compiler auf den seinerzeit vorhandenen Shared-Memory-Vektorrechnern. So boten Distributed-Memory-Systeme eine willkommene Alternative. Darüberhinaus lassen sich Anwendungen, die mittels Message-Passing für Distributed-Memory-Architekturen parallelisiert wurden, auch auf neueren parallelen Architekturen effizient ausführen, weil die explizite Formulierung Datenlokalität der Anwendung leistungssteigernd zur Geltung kommt und das Message-Passing-Programmiermodell weiterhin unterstützt wird.

5.3.2 Charakteristika der Anwendung

Numerische Lösungsverfahren für partielle Differentialgleichungen basieren auf einer räumlichen Diskretisierung des Problemgebiets. Dabei werden die für die Diskretisierung der Differentialoperatoren und das numerische Verfahren benötigten Daten konzeptuell den Objekten des Gitters zugeordnet. Die gängigen Diskretisierungen (FD, FE, FV) setzen auf den Daten lokaler Nachbarschaften in der Gittertopologie auf, die durch Iterationen über das gesamte Gitter einer Lösung zugetrieben werden. Für die Portierung solcher Verfahren auf Rechner mit vielen Prozessoren (Parallelisierung) liegt deshalb eine Datenpartitionierungsstrategie nahe. Dabei führen die beteiligten Prozessoren dasselbe Programm auf geringfügig überlappenden Gitterpartitionen aus.

Die Implementierung der in Kapitel 3 beschriebenen Gitterstruktur ist geprägt von einem vierschichtigen Modell von geometrischen Objekten (Knoten, Kanten, Facetten, Zellen),

welche mittels Referenzen auf Objekte der nächstniedrigeren Schicht eine "besteht-aus"-Relation beschreiben und mittels Referenzen auf Objekte der nächsthöheren Schicht eine "Oberfläche-von"-Relation. Diese Gitterstruktur kann als Graph aufgefasst werden, dessen Knoten die Gitterobjekte bilden und dessen Kanten durch die Relationen zwischen diesen Objekten beschrieben sind.

Zusätzlich zur Komplexität der Datenstruktur wird die Parallelisierung durch die Wahl eines selbst-adaptiven Verfahrens erschwert. So muss nach einer Adaption des Gitters auch eine Umverteilung erfolgen, damit der Arbeitsaufwand möglichst gleichmäßig auf die Prozessoren verteilt bleibt. Diese Umverteilung erfordert den Transfer von Objekten und Objektreferenzen über die Grenzen der lokalen Adressräume und eine konsistente Anpassung des Überlappungsbereichs.

5.3.3 Distributed, Dynamic Data

Das in [9] beschriebene Modell zur effizienten Parallelisierung von Algorithmen auf komplexen, dynamischen Datenstrukturen (*Distributed, Dynamic Data*, kurz: DDD) liefert eine Grundlage, auf der die Parallelisierung des hier beschriebenen Verfahrens überhaupt erst möglich erscheint.

Die dort implementierte Abstraktion des verteilten Graphen ist unabhängig von den Anforderungen der sequenziellen bzw. datenparallelen Numerik. Deshalb läßt sich das Modell in ein vorhandenes sequenzielles Programm problemlos integrieren. Dazu wird eine zusätzliche Datenstruktur in das Datenmodell der Anwendung aufgenommen und die Struktur der Gitterobjekte gegenüber dem Modell deklariert. Darüberhinaus wird der sequenzielle Algorithmus um Synchronisationsaufrufe angereichert, die den Datentransfer in den Überlappungsbereichen der lokalen Gitter anstoßen. An der Strukur des sequenziellen Programms sind *keine* Änderungen notwendig, so dass die Weiterentwickelbarkeit der Anwendung auch nach der Parallelisierung sichergestellt ist.

Der für die Anwendung schwierige Teil der Parallelisierung besteht in der Implementierung der Schnittstelle zwischen der Anwendung und der DDD-Bibliothek. Werden im Überlappungsbereich der lokalen Gitter Änderungen vorgenommen – etwa durch Verfeinerung und Vergröberung des Gitters – oder wird der Überlappungsbereich aufgrund der Ergebnisse der Lastbalancierung geändert, so müssen von der Anwendung beispielsweise Objekttransfers angestoßen werden. Ferner müssen Callback-Funktionen ("handler") für diese Transferoperationen implementiert werden, die es erlauben, mit den Objekten assoziierte Daten zu versenden, die nicht im Objekt selbst gespeichert sind. Dies trifft insbesondere für solche Daten zu, die dynamisch dimensioniert sind und deshalb nicht im Objekt selbst gespeichert werden können. Die Wahl einer geeigneten Strategie für das Zusammenspiel zwischen DDD-Operationen und den Handlern sowie der Test der Implementierung und die Identifikation von Fehlern erfordern große Umsicht.

Ein anderer Problemkreis betrifft die Optimierung der Datensynchronisation. Der Standardaufruf synchronisiert alle als global deklarierten Bereiche eines in der Überlappungszone liegenden Objekts. Wird aber nur ein kleiner Teil des Objekts geändert, erscheint eine spezialisierte Synchronisationsfunktion wünschenswert, die nur geänderte Bereiche des Objekts aktualisiert. Werden dabei Änderungen übersehen, führt dies zu schwer lokalisierbaren numerischen Fehlern. Dieses Phänomen tritt auch auf, wenn die Parallelisierung nach einer Weiterentwickelung des sequenziellen Algorithmus aktualisiert werden soll.

Die beschriebenen, durchaus schwierigen Portierungsprobleme müssen vor dem Hintergrund der möglichen Alternativen bewertet werden. Eine Distributed-Memory-Parallelisierung, die die Kommunikationsbedürfnisse ohne die in [9] entwickelten Abstraktionen unter direkter Verwendung von Message-Passing-Konstrukten befriedigt, wäre allein aufgrund der Komplexität der sequenziellen Software gescheitert. Im Nachhinein erscheint der Ansatz naiv, Distributed-Memory-Rechner auf diese Weise zu programmieren. Der in diesem Zusammenhang geprägte Begriff der "Wegwerfparallelisierung" beschreibt das Problem, dass sich in so entstandener Software leicht ein unentwirrbares Durcheinander aus numerischen und verteilten Aspekten der Implementierung einstellt und so die Verifikation des parallelen Codes und seine algorithmische Weiterentwickelung gefährdet.

Der Einsatz des DDD-Datenmodells isoliert die Anwendung durchgängig von den Schwierigkeiten, die durch den fehlenden globalen Adressraum entstehen und hebt den Vorgang der Parallelisierung damit auf ein der Anwendung näheres Abstraktionsniveau.

DDD stellt mit Hilfe globaler Objektkennungen eine Identifikation von Objekten über Prozessorgrenzen hinweg sicher. Mit Hilfe einer Anmeldung der verwendeten Datenstrukturen bei der DDD-Bibliothek wird es möglich, Objekte vom Adressraum eines Prozessors in den Adressraum eines anderen zu transferieren. Die dazu notwendigen Nachrichten baut DDD aus der Objektanmeldung selbstständig auf. Dabei werden Referenzen auf andere Objekte, z. B. auf benachbarte Gitterknoten, anhand der globalen Kennung automatisch zwischen den verschiedenen lokalen Adressräumen umgerechnet.

Die Implementierung des Überlappungsbereichs erfordert die Speicherung von Objektkopien auf den beteiligten Prozessoren und damit die Kommunikation bzw. Synchronisation zwischen diesen Kopien. DDD stellt eine logische Verbindung (sog. Interfaces) zwischen diesen Kopien und Methoden zur Synchronisation dieser Kopien zur Verfügung. Wiederum können die dazu notwendigen Nachrichten aus der Objektanmeldung abgeleitet werden. Darüberhinaus werden diese Interfaces nach einer Umverteilung des Gitters selbstständig aktualisiert.

Diese Leistungen von DDD sind verfügbar, nachdem eine kleine, DDD-spezifische Datenstruktur in die Datenstrukturen der Anwendung integriert wurde und die Anwendung ihre Datenstrukturen bei DDD angemeldet hat. Diese beiden Schritte sind in Programmiersprachen, die benutzerdefinierte Datenstrukturen kennen, einfach zu realisieren.

5.3.4 Der Überlappungsbereich

Um direkte Zugriffe auf Objekte zu vermeiden, die im Adressraum anderer Prozessoren liegen, werden von den benötigten Gitterobjekten lokale Kopien erzeugt, die nach den Bedürfnissen der Anwendung synchronisiert werden.

Dabei stellen die Module der Anwendung verschiedene Anforderungen an die Größe des Überlappungsbereichs. Der hier gewählte Ansatz für die Finite-Volumen-Diskretisierung besteht in der doppelten Speicherung der auf der Prozessorgrenze gelegenen Oberflächen. Damit beide angrenzenden Prozessoren für diese Oberfläche die Flussberechnung durchführen können, wird noch eine zusätzliche Schicht von Kontrollvolumen gespeichert, die außer der Konnektivität zur betrachteten Oberfläche keine weiteren Relationen benötigt. Da dieser Ansatz die doppelte Berechnung der Flüsse auf der Prozessorgrenze erfordert, wurde vorgeschlagen, die Flussberechnung nur auf einer Seite der Prozessorgrenze durchzuführen. Auf dieser Seite wird dann wie zuvor die zusätzliche Schicht von Kontrollvolumen benötigt.

Auf der anderen Seite wird lediglich eine Kopie der Oberfläche vorgehalten, auf die die Ergebnisse der Flussberechnung übertragen werden. Dieser Ansatz ist zwar effizienter, aber aufgrund der Asymmetrie der Grenze auch schwieriger zu behandeln, weshalb dem symmetrischen Ansatz der Vorzug gegeben wurde.

Für die parallele Gitterteilung wurde der Überlappungsbereich um alle Objekte erweitert, die mit einem ihrer Oberflächenobjekte (Knoten, Kanten, Facetten) an die Prozessorgrenze stoßen. Der erweiterte Überlappungsbereich erlaubt die Teilung auf den benachbarten Prozessoren unabhängig voneinander auszuführen, wobei identische Resultate garantiert werden können.

5.3.5 Parallele Gitterteilung

Im Rahmen der Arbeit von Klaus Birken, [9] wurde für die parallele Gitterteilung die Strategie eingesetzt, die Teilung des Gitters in der Überlappungszone durch Blockieren der übrigen beteiligten Prozessoren zu sequenzialisieren. Dieser Ansatz läßt sich schnell realisieren und erfordert keine tiefgehende Einsicht in die Datenstrukturen der Anwendung. Aus Sicht der Anwendung ist diese Lösung aus mehreren Gründen unbefriedigend. Zuallererst stellt eine solche Strategie einen Fremdkörper in der ansonsten globalen, datenparallelen Sicht der Anwendung auf ihre Datenstrukturen dar. Die im Rahmen dieser Arbeit durchgeführte Erweiterung der Parallelisierung von 2D auf 3D hätte deshalb besondere Probleme bereitet. Ferner ist am Ende jeder Blockade ein eigener aufwändiger Transfervorgang notwendig, um die durchgeführten Änderungen am Gitter auf die benachbarten Prozessoren auszubreiten.

Der hier entwickelte Ansatz orientiert sich streng an einer globalen, datenparallelen Sicht des Teilungsvorgangs. Dazu werden die Schnittflächen der Kontrollvolumen, die durch das Verfeinerungskriterium ermittelt wurden, im Überlappungsbereich vor der Teilung ausgetauscht. Dies versetzt jeden Prozessor in die Lage, die Teilung im Überlappungsbereich konsistent mit den benachbarten Prozessoren auszuführen. Die während der Teilung neu entstehenden Zellen, Facetten, Kanten und Knoten liegen in verschiedenen Adressräumen und erhalten zuerst unterschiedliche Objektkennungen. Da alle in die Teilung eingehenden Parameter zuvor synchronisiert wurden, entstehen diese Objekte auf den benachbarten Prozessoren in der gleichen Reihenfolge, anhand derer sie nach Abschluss der Teilung identifiziert werden können. Abschließend kann dann der Überlappungsbereich aktualisiert werden.

Die für die Identifikation notwendigen zusätzlichen Operationen sind objekt-spezifisch, lassen sich also einfach in die hierarchische Organisation des Teilungsalgorithmus eingliedern. Insbesondere sind diese Operationen nicht von der Raumdimension abhängig, so dass die Unabhängigkeit der Implementierung von der Raumdimension gewahrt bleibt. Die Identifikation von Objekten über Prozessorgrenzen hinweg wird von DDD mit einem eigenen Modul unterstützt. Dabei können unterschiedliche Merkmale als Identifikatoren eingesetzt werden, z. B. Objektkennungen, Ganzzahlen und Strings.

Kapitel 6

Numerische Ergebnisse

Die in Kapitel 3 beschriebene Gitterstruktur erlaubt es, das gesamte Rechengebiet durch eine einzige Masche zu repräsentieren. Ausgehend von dieser einen Masche und den Anfangs- und Randbedingungen kann die stationäre Gesamtlösung, d. h. das Gitter zusammen mit den Zustandswerten in selbst-adaptiver Weise erzeugt werden. Abschnitt 6.1 beschreibt die Strategie, die von diesem Ausgangszustand zur "fertigen" Lösung führt. Sie wird als integrierter Simulationsprozess bezeichnet, weil die Erzeugung des Gitters und der Zustandswerte in enger Kopplung von einem Werkzeug durchgeführt wird.

Abschnitt 6.2 zeigt eine Reihe von Testrechnungen, die belegen, dass sich trotz der ungewöhnlichen Gitter mit den vorgestellten Methoden in 2D genaue Ergebnisse erzielen lassen. Die untersuchten Testfälle umfassen subsonische, transonische und supersonische Rechnungen und zeigen einfache Kanalströmungen als auch die Umströmung eines Tragflügelprofils mit Klappe in 2D.

Erste Versuche in 3D wurden mit einer Überschall-Kanalströmung durchgeführt. Anhand eines DLR-F5 Tragflügels und einer Konfiguration aus Flugzeugkörper, Tragflügel, Triebwerksaufhängung und -gondel werden erste Versuche zur Verfeinerung von 3D-Geometrien gezeigt. Da die 3D-Verfeinerung noch nicht vollständig implementiert ist, konnten aber noch keine Rechnungen durchgeführt werden.

6.1 Der integrierte Simulationsprozess

Das Vorgehen zur Ermittlung einer stationären Lösung lässt sich grob in drei Phasen untergliedern: die Anfangsphase, die Adaptionsphase und die Abschlussphase. Der Anfangsphase fällt die Aufgabe zu, das gegebene Initialgitter unter Zuhilfenahme heuristischer Kriterien für die Adaptionsphase vorzubereiten. Während der Adaptionsphase wird ein Zyklus bestehend aus näherungsweiser Bestimmung der stationären Lösung, Bestimmung der zu adaptierenden Kontrollvolumen und Gitteradaption wiederholt. Die Abschlussphase dient der weiteren Verringerung des Residuums, bis die Lösung den stationären Zustand mit der gewünschten Genauigkeit erreicht.

6.1.1 Gitterverfeinerung in der Anfangsphase

Damit der Adaptionszyklus erfolgreich durchgeführt werden kann, muss ein Gitter bereitgestellt werden, auf dem mit dem gegebenen numerischen Verfahren eine Lösung ermittelt werden kann, die so genau ist, dass der gewählte Adaptionsindikator genügend Information über die notwendigen Veränderungen am Gitter liefern werden kann.

Für die in Abschnitt 6.2.4 vorgestellte Durchströmung eines Kanals genügt es beispielsweise, dass die gegenüberliegenden Wände bzw. Ein- und Auslauf des Kanals durch zwei Kontrollvolumen voneinander getrennt sind, d. h. es werden mindestens vier Kontrollvolumen benötigt. Wird die Adaptionsphase auf nur einem Kontrollvolumen gestartet, stehen für die lineare Rekonstruktion nicht genügend Informationen zur Verfügung und das Zeitschrittverfahren wird instabil. Durch wiederholte, uniforme Verfeinerung des Rechengebietes kann das Gitter hier geeignet verfeinert werden. Für komplexerere Konfigurationen, für die lokale Verfeinerungen zur Trennung gegenüberliegender Wände notwendig sind, kann die im folgenden Absatz vorgestellte Oberflächenverfeinerung eingesetzt werden.

Eine andere Situation ergibt sich bei Profilumströmungen. Die stark unterschiedlichen Strömungszustände im Fernfeld und an der Profiloberfläche (z. B. Überschall-Strömung im Fernfeld und Zustand im Staupunkt) sind auf einem Gitter mit sehr wenigen Kontrollvolumen nur wenige Kontrollvolumen voneinander entfernt. Dies führt zu starken Gradienten bei der Rekonstruktion, die die Strömung trotzdem nur schlecht repräsentieren und die Berechnung erschweren oder unmöglich machen. In den vorgestellten Berechnungen werden deshalb Kontrollvolumen, die an an eine Profiloberfläche grenzen mehrfach verfeinert. Im Unterschied zur Oberflächenverfeinerung bei kartesischen Gittern, die auch der Repräsentation der Geometrieoberfläche dient, genügen hier wenige Kontrollvolumen am Profil.

Die Oberflächenverfeinerung kann an den Rändern der Geometrie Kontrollvolumen erzeugen, deren Nachbarn wesentlich größer bzw. kleiner sind. Es zeigt sich, dass die Lösungsentwicklung dadurch gestört wird. Besonders deutlich sichtbar wird dies an der Entwicklung der Entropie, deren Transport entlang einer Profiloberfläche beeinträchtigt werden kann. Treten zusätzlich Oszillationen des Limiters auf, werden Entropieschwankungen in Kontrollvolumen eingetragen, die vom Profil weiter entfernt sind. Tritt eine solche Situation in einem Bereich auf, in dem alle Kontrollvolumen verfeinert werden, so bleibt das Größenverhältnis der Kontrollvolumen von der Verfeinerung unberührt, die Rekonstruktionen werden jedoch steiler. Um zu vermeiden, dass das Zeitschrittverfahren in der Adaptionsphase instabil wird, wird das Größenverhältnis benachbarter Kontrollvolumen in der Anfangsphase durch zusätzliche Verfeinerung auf einen Wert unter fünf begrenzt. In den in Abschnitt 6.2.2 vorgestellten Beispielen beginnt die Adaptionsphase dann mit etwa 90 bis 140 Kontrollvolumen im ganzen Rechengebiet.

Die beschriebene Problematik ist nicht auf Kontrollvolumen beschränkt, die direkt an den Gebietsrand angrenzen, sie wurde aber nur “in der Nähe” des Gebietsrandes beobachtet. Da diese Kontrollvolumen nicht einfach identifiziert werden können, wurde der Adaptionsindikator so modifiziert, dass das entstehen ungünstiger Größenverhältnisse im Verlauf der Adaptionsphase überall im Rechengebiet verhindert wird, vgl. Abschnitt 6.1.3.

6.1.2 Adaptionstechnik

Zur Adaption des Gitters an die Lösung wird die Technik der lokalen Verfeinerung und Vergröberung ("h-Methode") angewendet. Die zur Verfeinerung vorgesehenen Kontrollvolumen werden in 2D zweimal, in 3D dreimal entlang einer Schnittebene geteilt, die durch den Schwerpunkt des Kontrollvolumens verläuft. Ihre Richtung orientiert sich in den vorgestellten Berechnungen am lokalen Geschwindigkeitsvektor $\vec{v}$ und der dazu normalen Richtung $\vec{v}^{\perp}$ oder den Koordinatenrichtungen. Dabei wird diejenige Richtung gewählt, entlang der das Kontrollvolumen die größte "Ausdehnung" besitzt, um eine isotrope Verfeinerung unabhängig von der Form des Kontrollvolumens zu erreichen.

Zur Bestimmung der Ausdehnung werden die zweiten Momente des Kontrollvolumens

$$M_j^{(2)} := \int_{\Omega_j} (\vec{x} - \vec{x}_j)(\vec{x} - \vec{x}_j)^t \, \mathbf{d}\lambda \approx \sum_{k \in N_j} \langle (\triangle \vec{x}_{jk}, \vec{s}_{jk} \rangle \cdot \triangle \vec{x}_{jk} \cdot \triangle \vec{x}_{jk}^t$$

näherungsweise bestimmt. Dabei bezeichnet $\vec{x}_j$ den Schwerpunkt des Kontrollvolumens, $\triangle \vec{x}_{jk}$ die Differenz von Seitenschwerpunkt und $\vec{x}_j$, und $\vec{s}_{jk}$ die nach aussen gerichtete Normale der Seite. Als Normalenrichtung der Schnittebene wird die Richtung $\vec{w}$ gewählt, die

$$\vec{w}^t M_j^{(2)} \vec{w}\,, \quad \vec{w} \in \{\vec{v}, \vec{v}^{\perp}\}$$

maximiert. Die Wahl der Koordinatenrichtungen als Teilungsrichtungen führt zu Gittern die kartesischen Gitter sehr ähnlich sehen.

Auch andere Richtungswahlen wurden untersucht, wie die Richtung, in der die Rekonstruktion die Werte der Umgebung am schlechtesten approximiert, [14], oder entlang von Schockrichtungen, die aus den Rankine-Hugoniot-Gleichungen abgeleitet werden können. Die Tatsache, dass diese Richtungen sich mit jeder Verfeinerung langsam ändern, führt zu einer großen Zahl spitzwinkliger Dreiecke bzw. dreieckig geformter Kontrollvolumen im Gitter, deren Nachbarn nicht gleichmäßig in alle Raumrichtungen verteilt sind. Dies führt zu Problemen mit der Stabilität der Diskretisierung und der Konvergenz des Zeitschrittverfahrens. Die Verteilung der Nachbarschaft kann durch das Verschieben von Gitterknoten ("r-Methode") unter Beibehaltung der Gitterkonnektivität wesentlich verbessert werden und wurde für einfache Geometrien in [14] untersucht. Diese Technik bereitet bei komplexen Kontrollvolumen und in 3D jedoch Schwierigkeiten, vgl. Abschnitt 3.6.

6.1.3 Adaptionsindikator

Im Rahmen dieser Arbeit wurden Adaptionsindikatoren betrachtet, die die Approximationseigenschaften der Lösung bewerten. Dazu werden an jeder Seite die Rekonstruktionen der angrenzenden Kontrollvolumen verglichen. Für die Seite, die an die Kontrollvolumen Ω_j und Ω_k grenzt, und alle Kontrollvolumen Ω_j, $(j \in J)$, werden

$$c_{ij} = \max_{p \in \mathcal{S}(\partial\Omega_{jk})} ||\varphi_j(\vec{x}_p) - \varphi_k(\vec{x}_p)|| \in R^3 \tag{6.1}$$

$$r = ||\max_{j \in J} \varphi_j(\vec{x}_j) - \min_{j \in J} \varphi_j(\vec{x}_j)|| \in R^3 \tag{6.2}$$

$$c_j = max_{k \in N_j} \frac{|\Omega_j|}{|\Omega_k|} \frac{c_{ij}}{r} \in R^3 \tag{6.3}$$

$$\bar{c}_j = \max_{i=1..3} \langle c_j, \vec{e}_i \rangle \in R \tag{6.4}$$

bestimmt. Der Vergleich der Rekonstruktionen erfolgt nicht am Seitenschwerpunkt, sondern an den Schwerpunkten der Oberflächen der Seite bzw. an den Knoten der Seite, um Unterschiede im Gradienten entlang der Seite besser zu erfassen. Die Norm des Zustandsvektors bildet den Absolutbetrag der beiden skalaren Komponenten und die euklidische Norm der vektoriellen Komponente des Zustandsvektors. Als Bezugsgröße r für die Seitendifferenz c_{jk} wird die Differenz der komponentenweise gebildeten Extrema der Schwerpunktswerte der Kontrollvolumen herangezogen. Die Differenzen werden durch das Volumenverhältnis $|\Omega_j|/|\Omega_k|$ gewichtet. Dadurch wird die Verfeinerung von Kontrollvolumen beschleunigt, die größer als ihre Nachbarn sind und bei Kontrollvolumen verzögert, die kleiner als ihre Nachbarn sind. Der Vektor-Quotient ist ebenfalls komponentenweise zu verstehen, der Indikatorwert $\bar{c}_j$ entsteht schließlich aus der größten Komponente von c_j.

Verfeinerung und Vergröberung werden in der üblichen Weise durch zwei Schranken gesteuert. Liegt der Indikatorwert über der Verfeinerungsschranke, so wird das Kontrollvolumen verfeinert. Liegt er unterhalb der Vergröberungsschranke, so wird das Kontrollvolumen zur Vergröberung vorgesehen. Eine Vergröberung findet allerdings nur dann statt, wenn das andere Kontrollvolumen, welches an die letzte, durch Teilung eingefügte Seite angrenzt, ebenfalls zur Vergröberung vorgesehen ist, vgl. Abschnitt 3.5.

Die ursprüngliche Strategie, die Zustände jeweils in dem zur Rekonstruktion benutzten Variablensatz zu bewerten, wurde zugunsten der Differenz in in Entropie, Enthalpie und Geschwindigkeit aufgegeben. Insofern nähert sich der Adaptionsindikator dem Konzept Gradienten von Kenngrößen zu untersuchen.

6.1.4 Mängel des Indikators

Der Steuerung der Adaption durch den Adaptionsindikator liegen verschiedene Annahmen zugrunde, z. B. dass der verwendete Adaptionsindikator den lokalen Diskretisierungsfehler der Lösung misst und dass dieser Fehler durch lokale Verfeinerung abnimmt und durch lokale Vergröberung moderat zunimmt. Diese Annahmen sind im Prinzip nur für elliptische und parabolische Differentialgleichungen gerechtfertigt, [35, 38], für die Maximalprinzipien gelten. Hyperbolische Effekte, wie der Transport des Fehlers in benachbarte Kontrollvolumen bleiben bei den hier verwendeten, lokalen Betrachtungen ausser acht [56].

Ein weiterer, problematischer Aspekt des Adaptionsindikators zeigt sich an Unstetigkeiten der Lösung. An Unstetigkeiten, die entlang von Gitterlinien verlaufen, wie dies bei den transonischen Profilumströmungen in Abschnitt 6.2.2 Fall ist, wird die Unstetigkeit auf der Oberfläche des Kontrollvolumens oder durch ein Kontrollvolumen dargestellt, vgl. Abbildung 6.3. In diesen Bereichen reduziert die Verfeinerung den Adaptionsindikator nicht, d. h. dass dort bei jeder Adaption verfeinert wird. Um dies zu vermeiden, wird z. B. vorgeschlagen kleine Kontrollvolumen von der Verfeinerung auszunehmen, [36, 69].

Verläuft eine Unstetigkeit jedoch schräg zu den Gitterlinien, so wird sie durch drei bis vier Kontrollvolumen repräsentiert und zeigt einen relativ glatten Übergang, vgl. Abbildung 6.5. Dadurch werden diese Unstetigkeiten nur schwach verfeinert. Dem kann durch die Verwendung von Schockindikatoren, vgl. z. B. [69], oder durch Anpassung des Gitters an die Unstetigkeit, vgl. [14] begegnet werden.

Ein für die Anwendung des Verfahrens wichtigerer Aspekt der Adaption ist der nicht ausreichend geklärte Einfluss der Gittergeometrie auf die Diskretisierung. Besonders gilt dies für die Anfangsphase mit nur wenigen (100-300) Kontrollvolumen. Die adaptive Wahl der

Verfeinerungsrichtungen bietet gute Möglichkeiten, die Zahl der benötigten Kontrollvolumen zu reduzieren, z. B. durch anisotrope Verfeinerung oder durch Orientierung der Richtung an Unstetigkeiten. Dies bereitet bisher Schwierigkeiten, weil eine ungünstige Wahl der Richtungen die Konvergenz des Zeitschrittverfahrens beeinträchtigt. Ausserdem zeigt sich, dass nur ein kleiner Teil der Kontrollvolumen, die die zum Vergröbern vorgesehen sind, auch tatsächlich vergröbert wird.

Eine messbare Kontrolle über den Einfluss der Geometrie auf die Diskretisierung würde es erlauben, die Einschränkungen bei der Richtungswahl und beim Vergröbern fallen zu lassen.

6.2 Testrechnungen (2D)

Die im Folgenden vorgestellten Testrechnungen zeigen, dass sich mit den in Kapitel 3 vorgestellten Methoden der Gittererzeugung und den in Kapitel 4 beschriebenen Diskretisierungstechniken stationäre Lösungen der Euler-Gleichungen unter verschiedenen Strömungsbedingungen genau berechnen lassen.

In den Beispielen werden Geometrien unterschiedlicher Komplexität behandelt, angefangen mit einfachen Kanalgeometrien (Abschnitt 6.2.4) bis zu zweiteiligen Tragflügelprofilen. Abgesehen von der Bereitstellung des Anfangsgitters wurde in keinem Fall Einfluss auf die Geometrie des Gitters genommen. Insofern kann das Ziel der integrierten Gittergenerierung als erreicht gelten.

6.2.1 Darstellung der Lösung

Besonderes Interesse bei der Darstellung der Lösungen galt den verwendeten Rekonstruktionen. Für die graphische Darstellung der Ergebnisse werden deshalb die Werte der Rekonstruktion auf jeder Kante des Gitters an den beiden Endknoten und dem Kantenschwerpunkt ausgewertet. Aus den so ermittelten Zustandsvektoren können alle benötigten Werte ermittelt werden.

Die Berechnung von Isolinien basiert auf einer impliziten Triangulierung der Facetten. Für jede Kante einer Facette werden zwei Dreiecke aus dem Facettenschwerpunkt, dem Kantenschwerpunkt und einem Knoten der Kante gebildet. Auf diesen Dreiecken werden Isolinien durch lineare Approximation der Eckpunktswerte approximiert. Diese Vorgehensweise führt dazu, dass an allen Gitterknoten und allen Kantenmitten zwei oder mehr Zustandswerte vorhanden sind und Unstetigkeiten in der Rekonstruktion dargestellt werden können. Deshalb zeigen alle Lösungsdarstellungen, auch die Abbildungen von Isolinien Unstetigkeiten.

Ein Problem dieser Darstellungsmethode wird allerdings in den hier gezeigten Abbildungen des Totaldrucks sichtbar, dessen Gradienten nahezu verschwinden und so kleine Oszillationen, die an den Nasen der Tragflügelprofile auftreten schlecht erkennen lassen.

Für die Bewertung der Qualität der Lösung spielen sogenannte Überschießer eine besondere Rolle, die den Wertebereich der umgebenden Schwerpunkte über- oder unterschreiten. Solche Werte werden besonders in der 3D-Darstellung des Lösungsgraphen an Unstetigkeiten sichtbar. Die Vergrößerung dieser Darstellungen zeigt, dass solche Werte besonders

deutlich an den Knoten der Kanten auftreten, während die Werte an den Kantenschwerpunkten dieses Verhalten nicht zeigen. In das numerische Verfahren und die Stabilitätsbetrachtungen gehen direkt nur die Werte an den Kantenmitten ein. Diese zeigen aufgrund der Limitierung des Gradienten anhand der Werte an den Kantenschwerpunkten nur geringe Überschießer. Kleine Überschiesser werden aber trotz Limitierung sichtbar, z. B. in den Druckverläufen, weil die Limitierung im gewählten Variablensatz durchgeführt wird und nicht in den hier dargestellten Druckwerten.

6.2.2 Umströmung eines NACA-0012-Tragflügelprofils

Das NACA-0012-Profil wird häufig für Vergleichsrechnungen herangezogen. Die erste Gruppe von Testfällen zeigt subsonische, transonische und supersonische Umströmungen dieses Profils bei verschiedenen Anstellwinkeln.

Die offene Hinterkante des Profils wurde durch Fortsetzen des Profils bis an den Nullpunkt der beschreibenden Kurve geschlossen und auf Einheitslänge reskaliert, vgl. [3]. Die Tragflügelnase befindet sich bei $x = 0$, die Länge des Profils ist eins. Das Profil ist von einem Quadrat der Kantenlänge 256 mit den Eckpunkten $(-128, -128)$ und $(128, 128)$ umgeben. Die Randkurve des Profils wird durch 500 Knoten dargestellt. Abbildung 6.1 zeigt das Initialgitter mit 504 Knoten und das Profil.

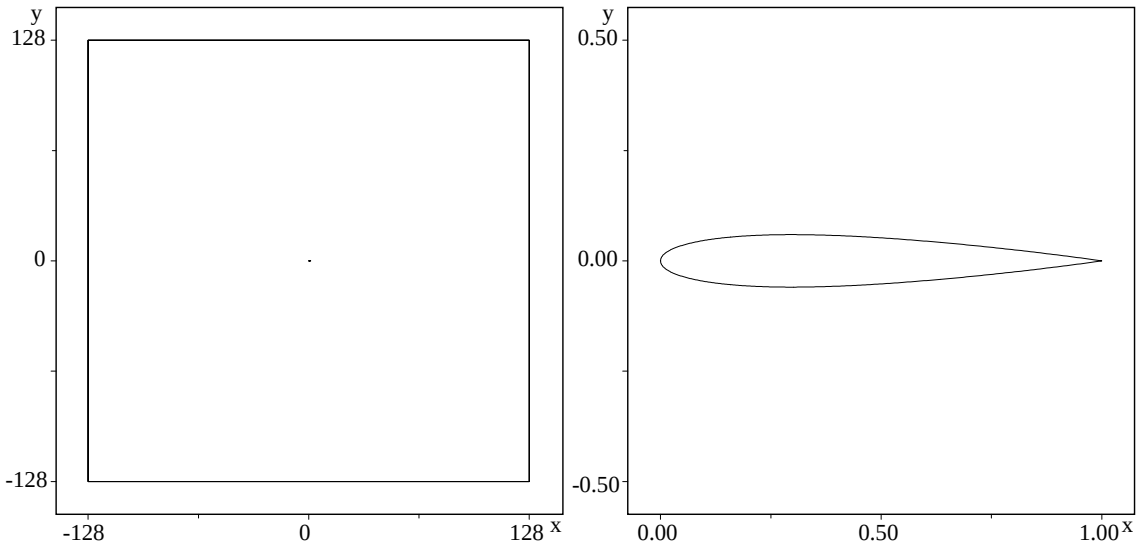


Abbildung 6.1: NACA-0012: Initialgitter und Profil

Subsonische Umströmung bei $M_\infty = 0.63$ und Anstellwinkel $\alpha = 2°$.

Die Strömung bleibt im ganzen Gebiet subsonisch, als größten Wert der Machzahl erhält man $M = 0.990$. Abbildung 6.2 zeigt die Verteilung der Machzahl. Der Druckbeiwert C_P und der Totaldruck P auf der Oberfläche des Profils sind in Abbildung 6.8 dargestellt.

Für den Auftriebsbeiwert C_L und den Wiederstandsbeiwert C_D ermittelt man $C_L = 0.3299$ und $C_D = 2.4 \cdot 10^{-4}$. Eine vergleichbare Lösung auf einem kartesischen Gitter mit 10694 Kontrollvolumen liefert die Beiwerte $C_L = 0.3289$ und $C_D = 4 \cdot 10^{-4}$ [69]. [60] ermitteln

die Beiwerte $C_L = 0.3335$ und $C_D = 3 \cdot 10^{-5}$ auf einem strukturierten Gitter mit 16705 Knoten.

Abbildung 6.9 zeigt das Gitter nach der initialen Verfeinerung mit 132 Kontrollvolumen und nach der letzten Adaption mit 10597 Kontrollvolumen.

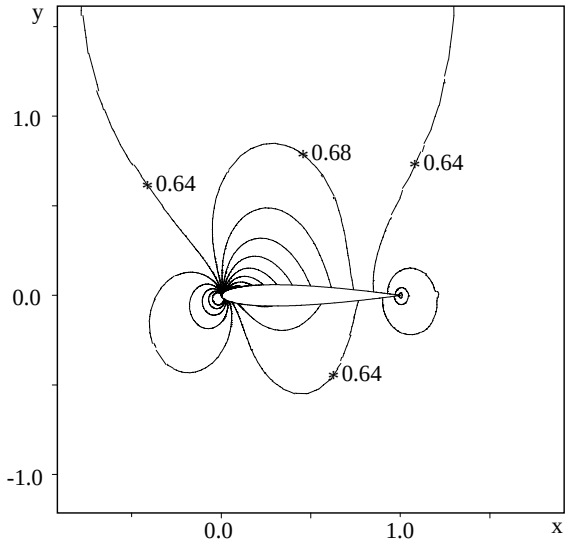


Abbildung 6.2: Mach-Isolinien (min=0.45, max=0.95, inc=0.04) für NACA-0012, $M_\infty = 0.63$, $\alpha = 2^\circ$

Transsonische Umströmung bei $M_\infty = 0.8$ und Anstellwinkel $\alpha = 1.25^\circ$

Die Lösung zeigt einen Schock auf der Oberseite des Profils bei $x \approx 0.64$ und einen schwächeren auf der Unterseite bei $x \approx 0.33$, sowie eine Scherschicht, die an der Hinterkante des Profils beginnt. Abbildung 6.3 zeigt die Verteilung der Machzahl. Der Druckbeiwert C_P und der Totaldruck P auf der Oberfläche des Profils sind in Abbildung 6.10 dargestellt und zeigen die sehr gute Auflösung des Schocks.

Für den Auftriebsbeiwert C_L und den Wiederstandsbeiwert C_D ermittelt man die Werte $C_L = 0.357$ und $C_D = 0.0230$. Die Referenzlösungen aus [3] auf strukturierten Gittern liefern die Beiwerte $C_L = 0.3632$, $C_D = 0.0230$ (20480 Knoten), $C_L = 0.3474$, $C_D = 0.0221$ (7388 Knoten) und $C_L = 0.3463$, $C_D = 0.0223$ (3634 Knoten).

Abbildung 6.11 zeigt das Gitter mit 110 Kontrollvolumen nach der initialen Verfeinerung und nach der letzten Adaption mit 10179 Kontrollvolumen.

Transsonische Umströmung bei $M_\infty = 0.85$ und Anstellwinkel $\alpha = 1^\circ$

Die Lösung zeigt Schocks vergleichbarer Stärke auf der Oberseite des Profils bei $x \approx 0.85$ und der Unterseite bei $x \approx 0.64$, sowie eine Scherschicht, die an der Hinterkante des Profils beginnt.

Abbildung 6.4 zeigt die Verteilung der Machzahl. Der Druckbeiwert C_P und der Totaldruck P auf der Oberfläche des Profils sind in Abbildung 6.13 dargestellt. Der Druckverlauf stimmt mit den in [3] und [69] dargestellten Werten sehr gut überein.

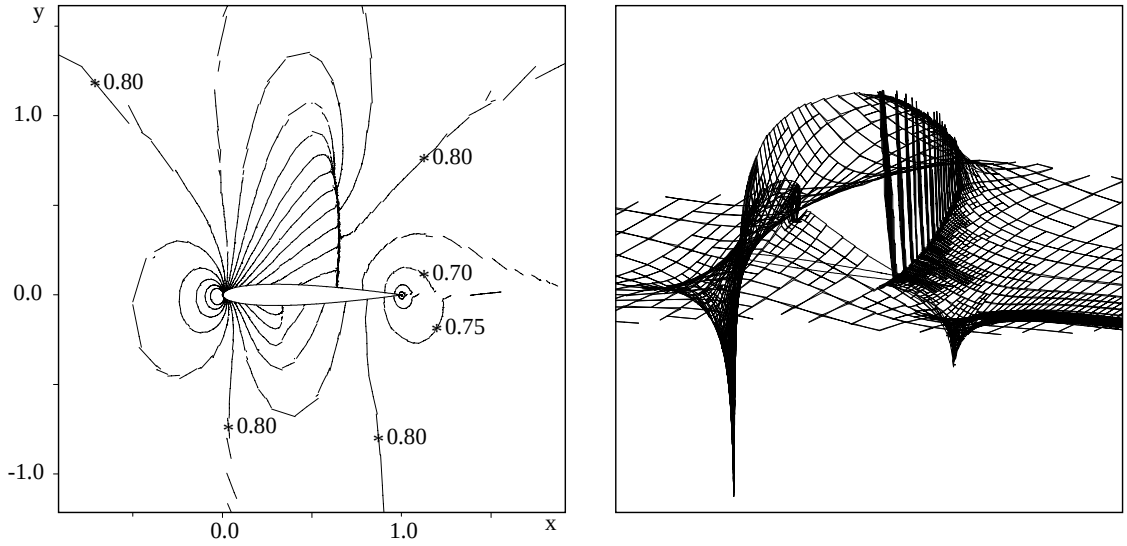


Abbildung 6.3: Mach-Isolinien (min=0.60, max=1.35, inc=0.05, links) und Mach-Graph (rechts) für NACA-0012, $M_\infty = 0.8$, $\alpha = 1.25°$

Der Totaldruck zeigt auch an der Profilnase nur geringe Ungenauigkeiten. Wie beim vorherigen Testfall treten an jedem Schock ein isolierter Wert und kleine Oszillationen an der Profilnase auf.

Für den Auftriebsbeiwert C_L und den Wiederstandsbeiwert C_D ermittelt man $C_L = 0.3804$ und $C_D = 0.0582$. Für die Referenzlösungen in [3] auf strukturierten Gittern werden die Beiwerte $C_L = 0.3584$, $C_D = 0.0580$ (20480 Knoten) und $C_L = 0.3472$, $C_D = 0.0557$ (3634 Knoten) angegeben. Aus einer anderen Lösung auf strukturierten Gittern [60] wurden $C_L = 0.3793$ und $C_D = 0.0576$ ermittelt. Auf einem kartesischen Gitter mit 13515 Kontrollvolumen ermittelt [69] die Beiwerte $C_L = 0.3682$ und $C_D = 0.0599$.

Abbildung 6.14 zeigt das Gitter mit 137 Kontrollvolumen nach der initialen Verfeinerung und nach der letzten Adaption mit 14995 Kontrollvolumen.

Supersonische Umströmung bei $M_\infty = 0.95$ und Anstellwinkel $\alpha = 0°$

Die Lösung zeigt zwei schräge Schocks, die von der Hinterkante des Profils ausgehen, sowie einen schwachen Schock in Normalenrichtung im Nachlauf bei $x \approx 4.38$. Für die Position dieses Schocks ermittelt [69] $x = 4.41$, in [3] konnte wegen zu unterschiedlicher Ergebnisse keine Position festgestellt werden. Abbildung 6.5 zeigt die Verteilung der Machzahl und eine Ansicht der schrägen Schocks an der Profilhinterkante. Diese Ansicht zeigt, dass der schräge Schock, von der Verfeinerung unabhängig, durch drei bis fünf Kontrollvolumen aufgelöst wird. Für den Auftriebsbeiwert C_L und den Wiederstandsbeiwert C_D ermittelt man $C_L = 3.0310^{-8}$ und $C_D = 0.1084$. Die Referenzlösungen auf strukturierten Gittern in [3] mit 20480 bzw. 3840 Knoten liefern Wiederstandsbeiwerte $C_D = 0.1084$ und $C_D = 0.1085$. Abbildung 6.16 zeigt das Gitter nach der initialen Verfeinerung mit 92 Kontrollvolumen und nach der letzten Adaption mit 14299 Kontrollvolumen. Für diese Rechnung wurden niedrigerere Schwellwerte für die Adaption benutzt, weil die Position des schwachen Schocks im Nachlauf von der Verfeinerung in diesem Bereich beeinflusst

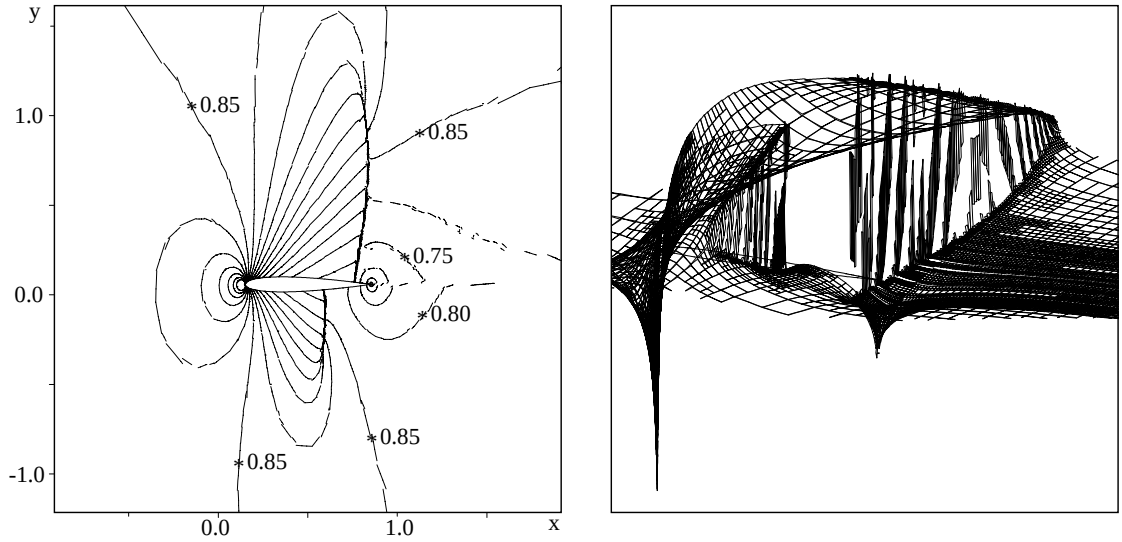


Abbildung 6.4: Mach-Isolinien (min=0.55, max=1.40, inc=0.05, links) und Mach-Graph (rechts) für NACA-0012, $M_\infty = 0.85$, $\alpha = 1^\circ$

wird. Eine Abhängigkeit von der räumlichen Auflösung vor der Profilhinterkante, vgl. [69] wurde hingegen nicht beobachtet.

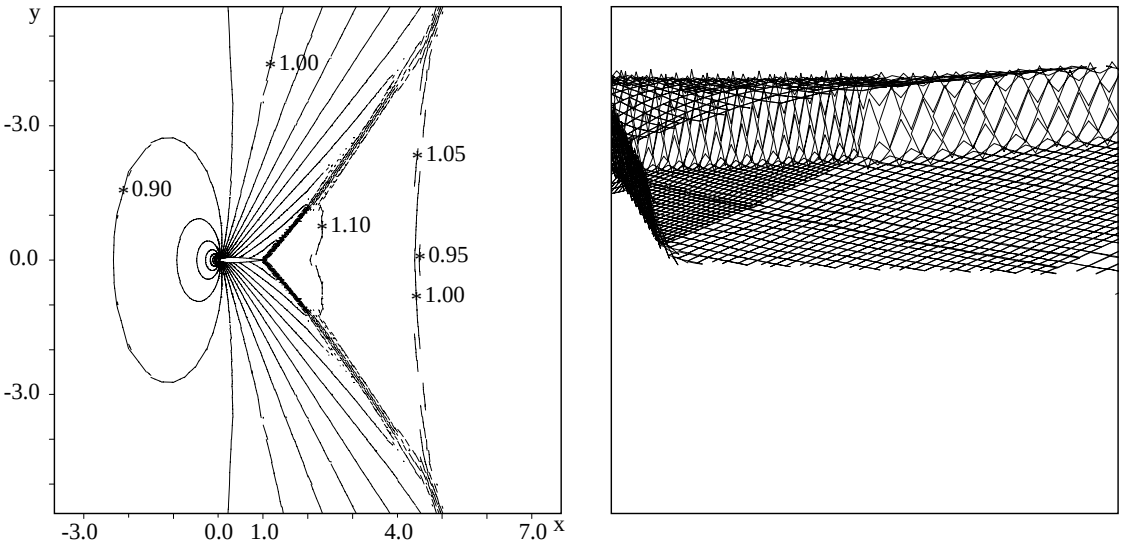


Abbildung 6.5: Mach-Isolinien (min=0.55, max=1.45, inc=0.05, links) und Mach-Graph (rechts) für NACA-0012, $M_\infty = 0.95$, $\alpha = 0^\circ$

Supersonische Umströmung bei $M_\infty = 1.2$ und Anstellwinkel $\alpha = 0^\circ$

Die Lösung zeigt einen abgelösten Bogenschock vor dem Profil bei $x = -0.47$. und einen schrägen Schock, der von der Hinterkante des Profils ausgeht. Abbildung 6.6 zeigt die Verteilung der Machzahl. Für den Auftriebsbeiwert C_L und den Wiederstandsbeiwert C_D ermittelt man $C_L = 5.2 \cdot 10^{-10}$ und $C_D = 0.0952$. Die Referenzlösungen auf strukturierten Gittern in [3] mit 20480 bzw. 3840 Knoten liefern die Wiederstandsbeiwerte $C_D = 0.0960$

bzw. $C_D = 0.0951$. Abbildung 6.18 zeigt das Gitter nach der initialen Verfeinerung mit 92 Kontrollvolumen und nach der letzten Adaption mit 12832 Kontrollvolumen.

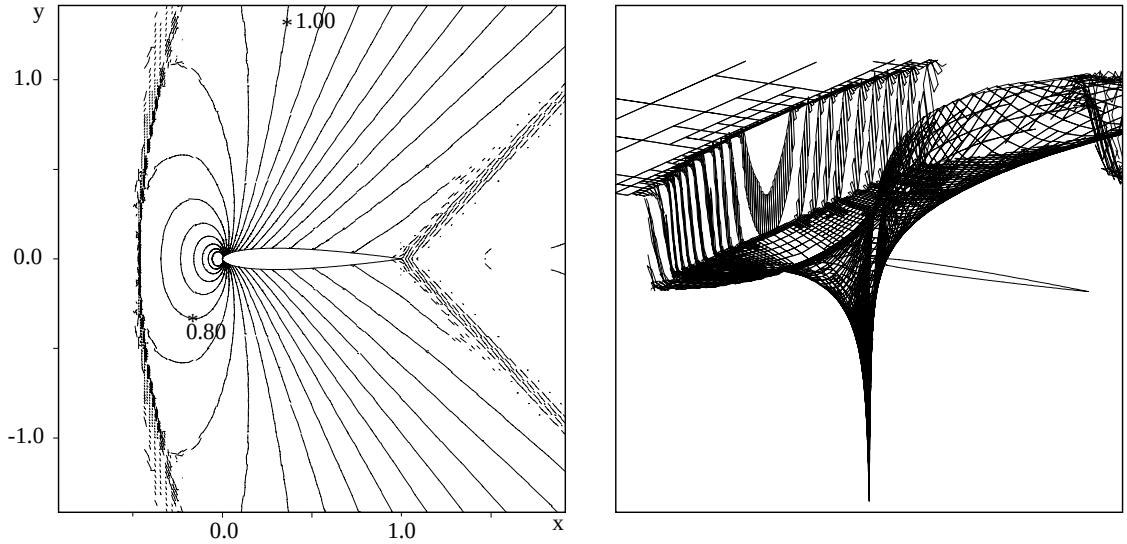


Abbildung 6.6: Mach-Isolinien (min=0.55, max=1.50, inc=0.05, links) und Mach-Graph (rechts) für NACA-0012, $M_\infty = 1.2$, $\alpha = 0°$

Supersonische Umströmung bei $M_\infty = 1.2$ und Anstellwinkel $\alpha = 7°$

Die Lösung zeigt einen Bogenschock vor dem Profil, der die x-Achse bei $x = -0.58$ schneidet. Von der Hinterkante des Profils geht ein einzelner schräger Schock und eine Scherschicht aus. Abbildung 6.7 zeigt die Verteilung der Machzahl. Für den Auftriebsbeiwert C_L und den Wiederstandsbeiwert C_D ermittelt man $C_L = 0.5251$ und $C_D = 0.1544$. Die Referenzlösungen auf strukturierten Gittern in [3] mit 10209 bzw. 11055 Knoten liefern die Auftriebsbeiwerte $C_L = 0.5138$ bzw. $C_L = 0.5280$ und die Wiederstandsbeiwerte $C_D = 0.1538$ und $C_D = 0.1536$.

Abbildung 6.20 zeigt das Gitter nach der initialen Verfeinerung mit 111 Kontrollvolumen und nach der letzten Adaption mit 9474 Kontrollvolumen.

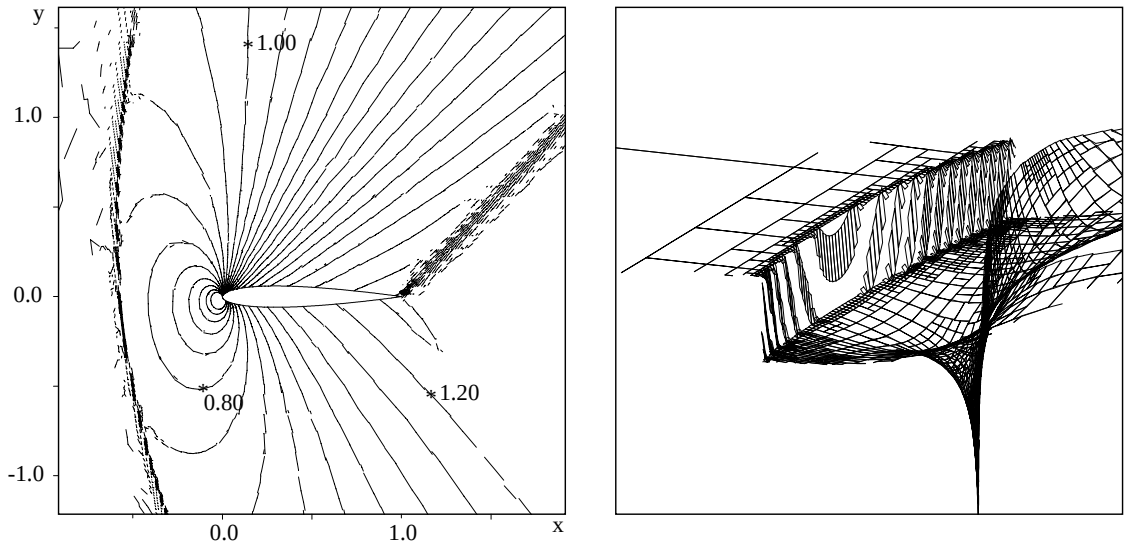


Abbildung 6.7: Mach-Isolinien (min=0.55, max=1.80, inc=0.05 links) und Mach-Graph (rechts) für NACA-0012, $M_\infty = 1.2$, $\alpha = 7°$

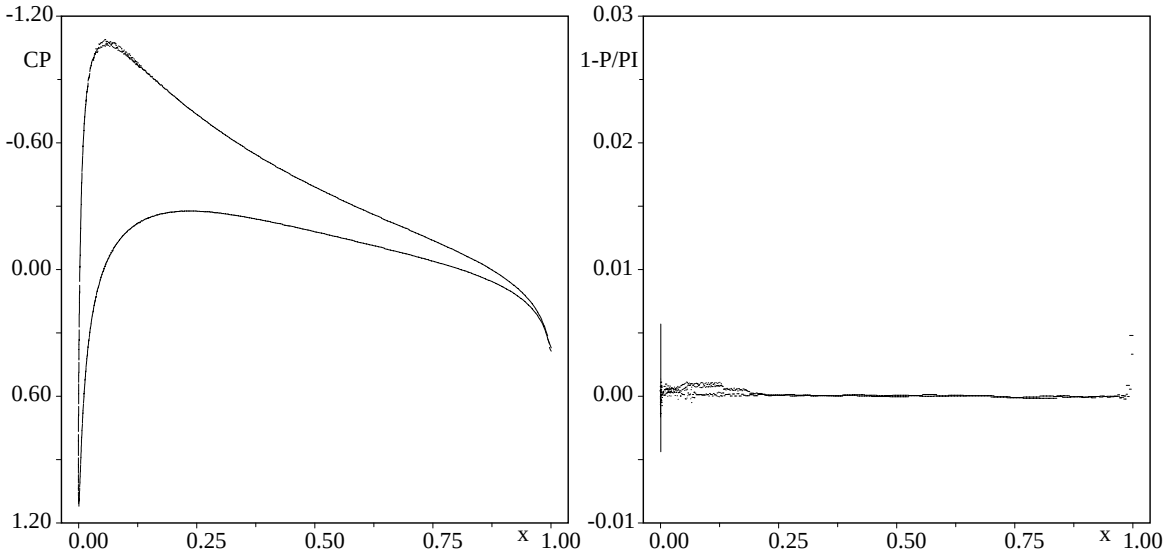


Abbildung 6.8: Druckbeiwert C_P und Totaldruck $1 - P/P\infty$ für NACA-0012, $M_\infty = 0.63$, $\alpha = 2^\circ$

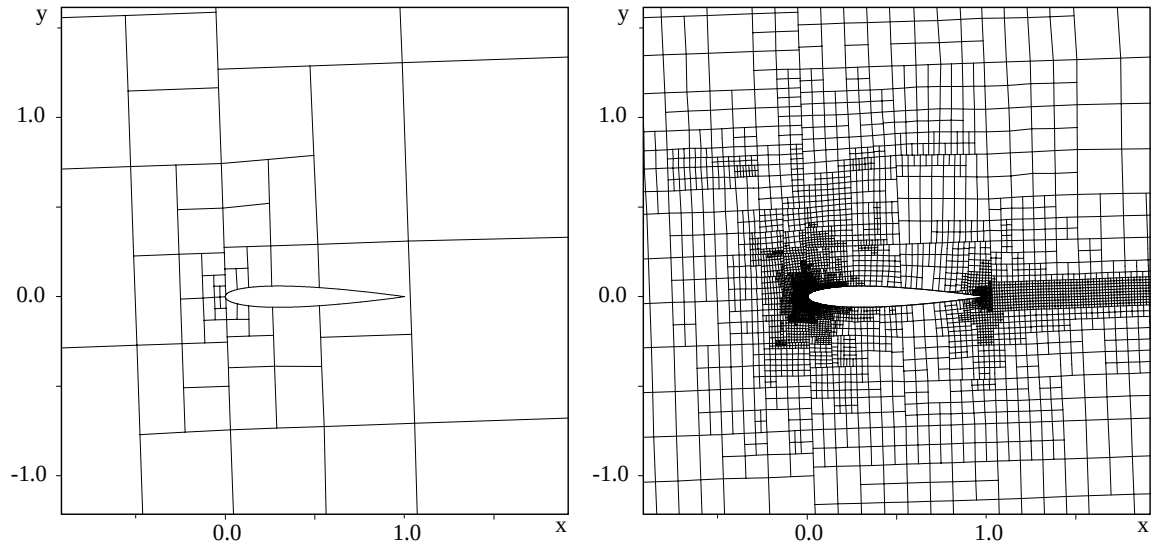


Abbildung 6.9: Gitterausschnitte vor und nach Adaption für NACA-0012, $M_\infty = 0.63$, $\alpha = 2^\circ$

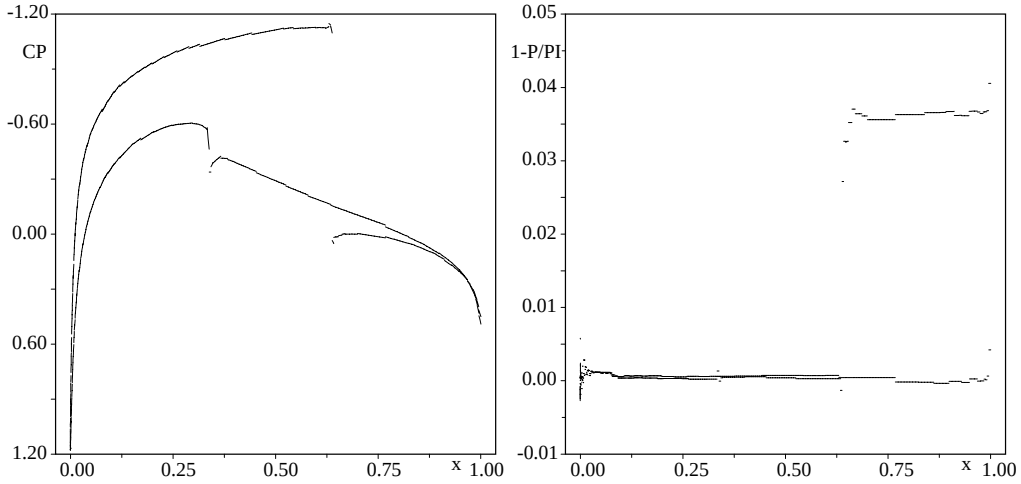


Abbildung 6.10: Druckbeiwert C_P und Totaldruck $1 - P/P\infty$, $M_\infty = 0.8$, $\alpha = 1.25°$

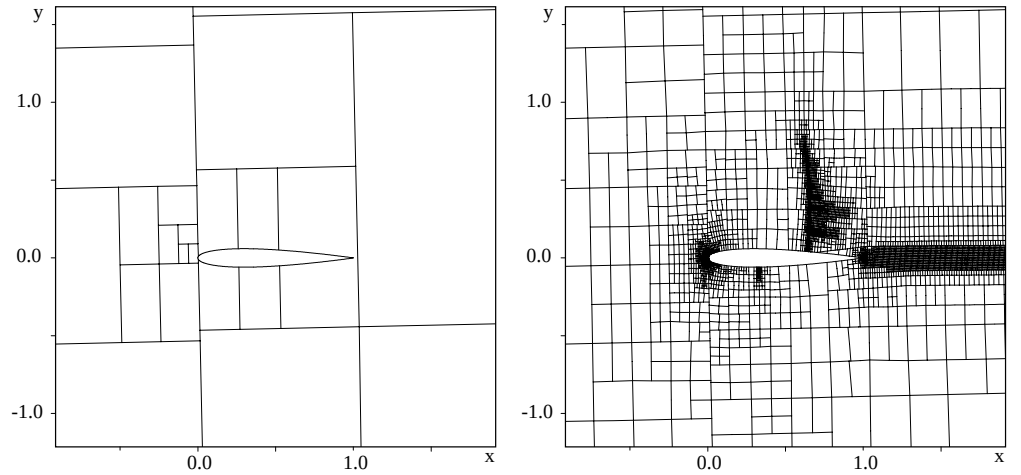


Abbildung 6.11: Gitterausschnitte vor und nach Adaption, $M_\infty = 0.8$, $\alpha = 1.25°$

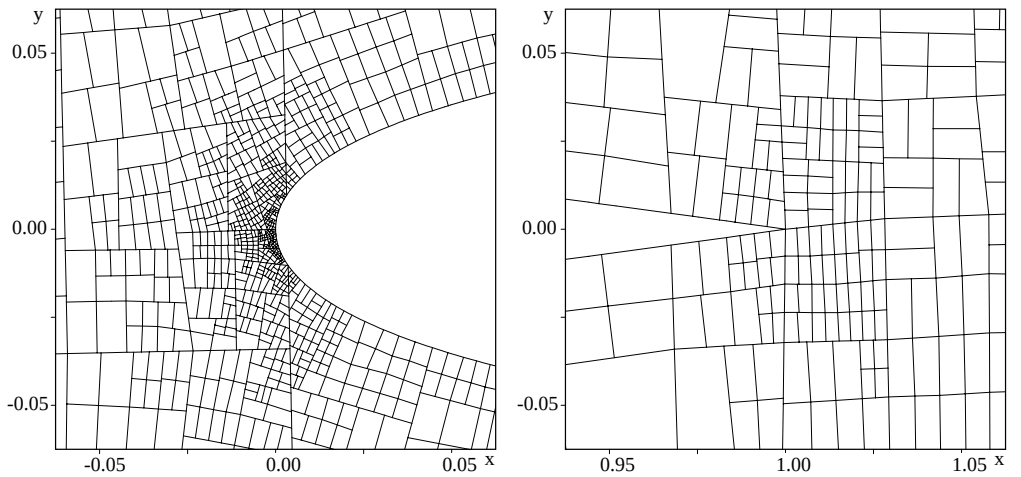


Abbildung 6.12: Gitterausschnitte der Profilenden für NACA-0012, $M_\infty = 0.8$, $\alpha = 1.25°$

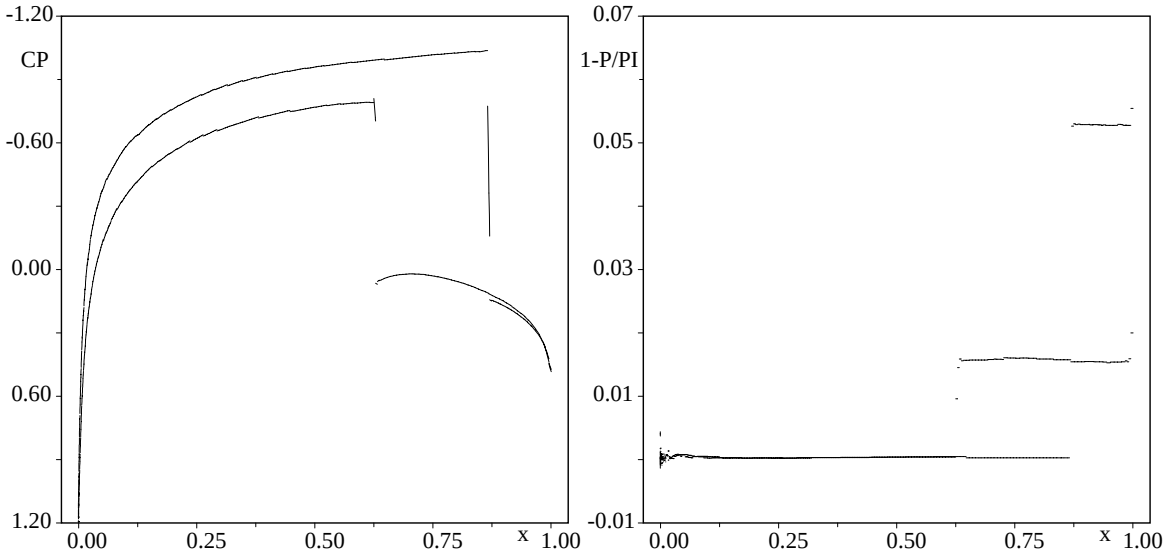


Abbildung 6.13: Druckbeiwert C_P und Totaldruck $1-P/P\infty$ für NACA-0012, $M_\infty = 0.85$, $\alpha = 1^\circ$

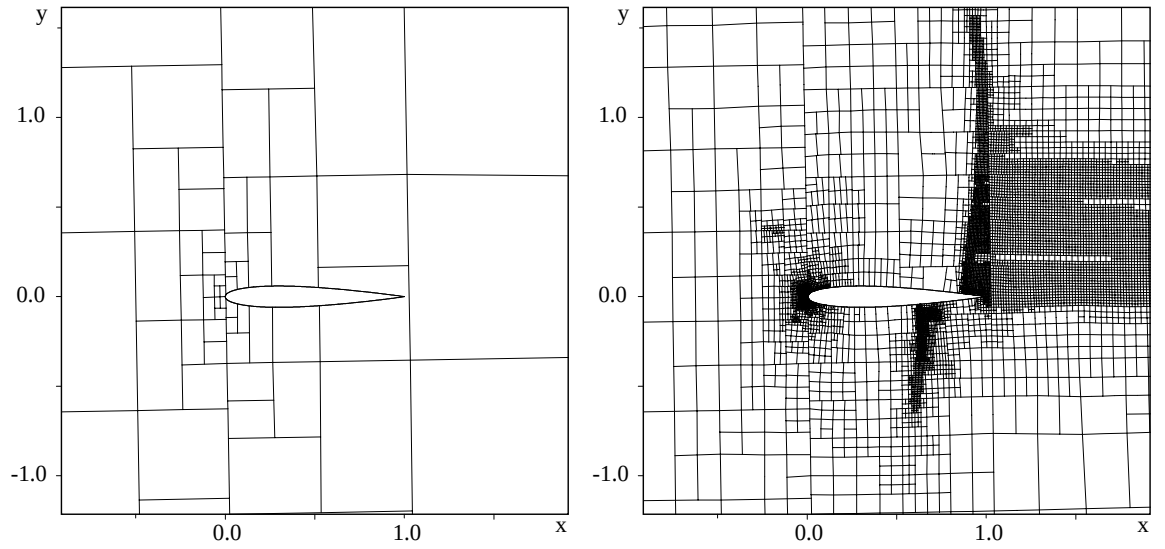


Abbildung 6.14: Gitterausschnitte vor und nach Adaption für NACA-0012, $M_\infty = 0.85$, $\alpha = 1^\circ$

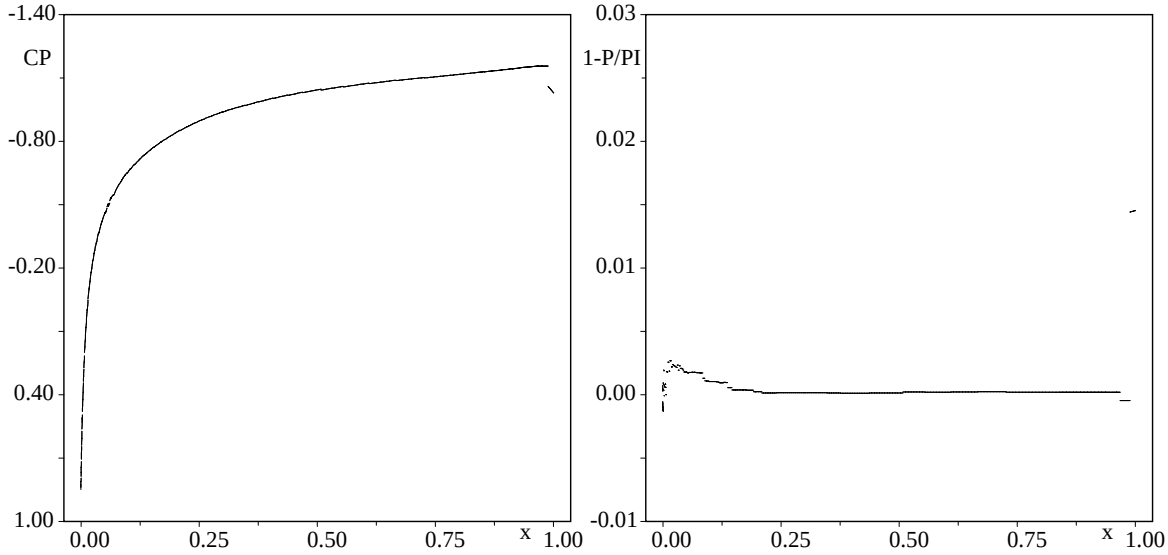


Abbildung 6.15: Druckbeiwert C_P und Totaldruck $1-P/P\infty$ für NACA-0012, $M_\infty = 0.95$, $\alpha = 0^\circ$

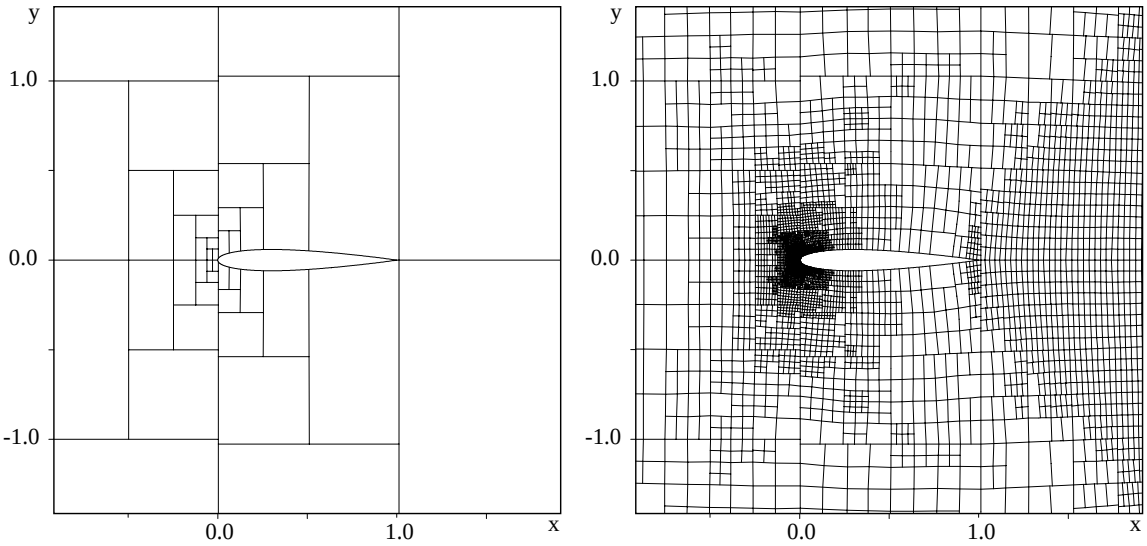


Abbildung 6.16: Gitterausschnitte vor und nach Adaption für NACA-0012, $M_\infty = 0.95$, $\alpha = 0^\circ$

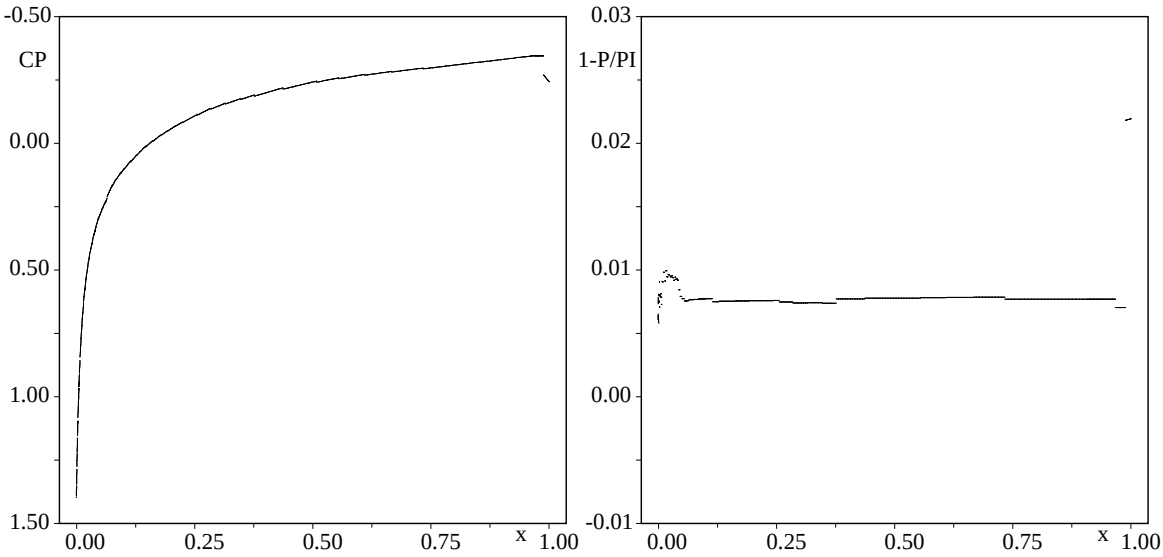


Abbildung 6.17: Druckbeiwert C_P und Totaldruck $1 - P/P\infty$ für NACA-0012, $M_\infty = 1.2$, $\alpha = 0°$

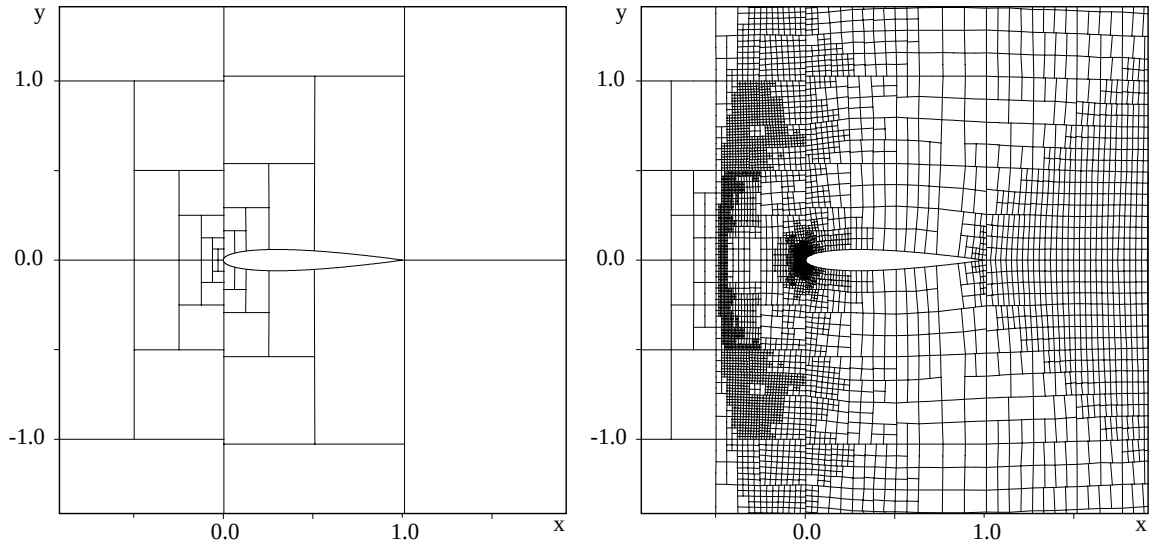


Abbildung 6.18: Gitterausschnitte vor und nach Adaption für NACA-0012, $M_\infty = 1.2$, $\alpha = 0°$

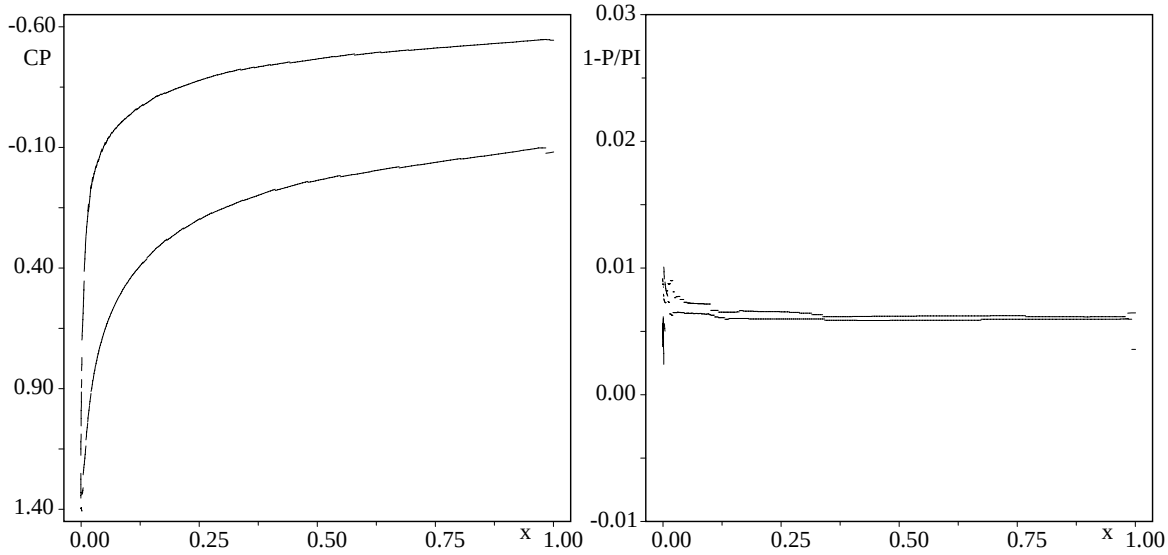


Abbildung 6.19: Druckbeiwert C_P und Totaldruck $1 - P/P\infty$ für NACA-0012, $M_\infty = 1.2$, $\alpha = 7^\circ$

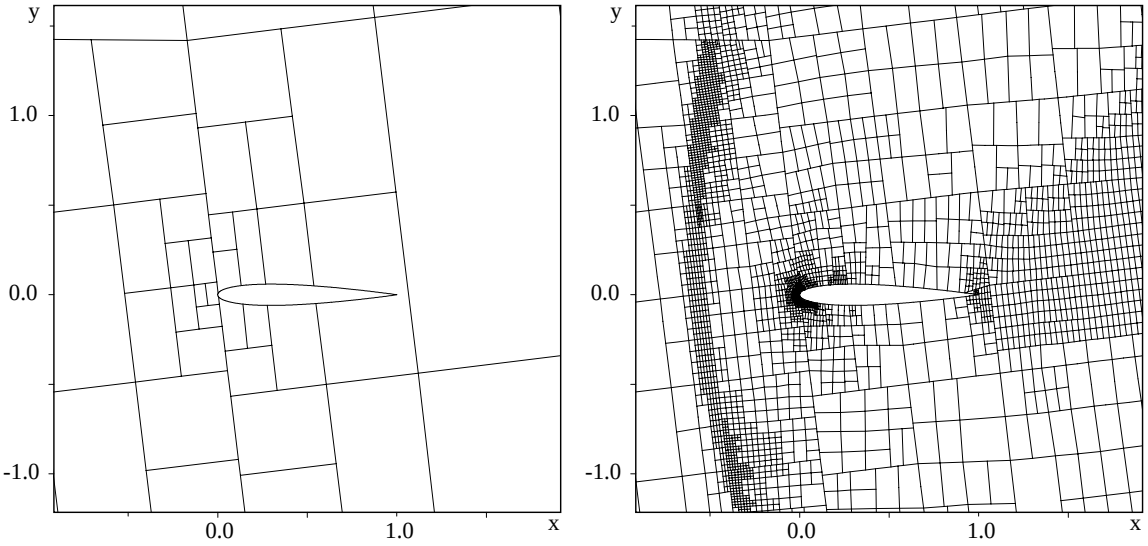


Abbildung 6.20: Gitterausschnitte vor und nach Adaption für NACA-0012, $M_\infty = 1.2$, $\alpha = 7^\circ$

6.2.3 Umströmung eines NLR-7301 Tragflügelprofils

Dieser Testfall zeigt die Umströmung eines NLR-7301-Profils mit Klappe. Sowohl das Hauptprofil (424 Knoten) als auch die Klappe (216 Knoten) wurden bis zum Schnittpunkt der letzten beiden Kanten verlängert, aber nicht reskaliert. Die Tragflügelnase (des Hauptprofils) ist bei $x = 0$, die Länge des (Haupt-)Profils ist 0.9534. Das Rechengebiet wird von einem Quadrat der Kantenlänge 256 mit den Eckpunkten $(-128, -128)$ und $(128, 128)$ begrenzt. Abbildung 6.21 zeigt die Konfiguration und die Isolinien der Machzahl, Abbildung 6.22 zeigt das Gitter am Anfang der adaptiven Phase mit 240 Kontrollvolumen und nach der letzten Adaption mit 10437 Kontrollvolumen. Für diesen Fall standen keine Vergleichsdaten zur Verfügung.

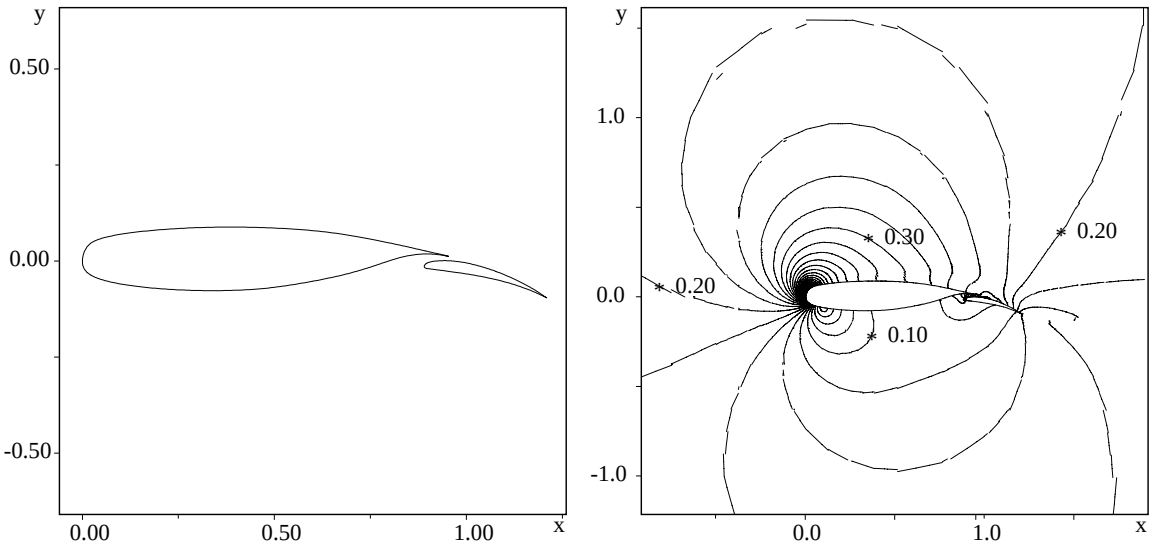


Abbildung 6.21: Konfiguration und Mach-Isolinien (min=0.02, max=0.84, inc=0.02) für NLR-7301, $M_\infty = 0.185$, $\alpha = 13.1°$

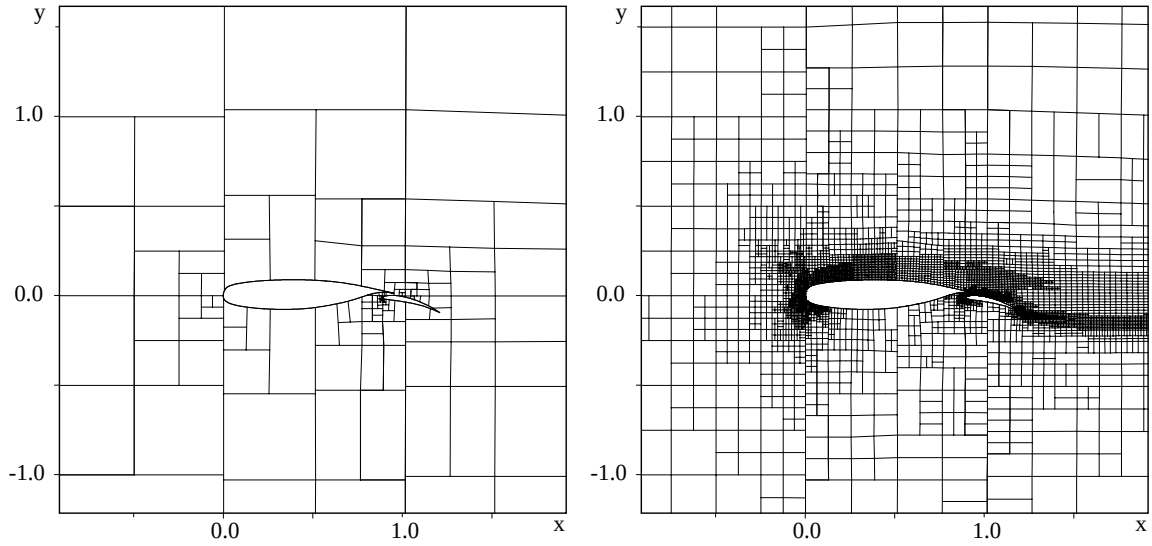


Abbildung 6.22: Gitterausschnitt für NLR-7301, $M_\infty = 0.185$, $\alpha = 13.1°$

6.2.4 Kanal mit 15° Rampe

In diesem Testfall wird die Strömung in einem Kanal über eine Rampe berechnet. Die Rampe hat eine Steigung von 15°, die Anströmmachzahl ist $M_\infty = 2.0$. Am Fußpunkt der Rampe entsteht ein Schock, der an der gegenüberliegenden Wand reflektiert wird und von dem Expansionsfächer beeinflusst wird , der sich am Ende der Rampe ausbildet. Dabei bildet sich am oberen Reflexionspunkt ein kleiner Machstamm aus, am Tripelpunkt bildet sich eine Scherschicht. Der Schock wird an der unteren Wand ein weiteres Mal reflektiert, bevor er das Rechengebiet verlässt.

Abbildung 6.23 zeigt den Verlauf der Machzahl im Rechengebiet, der Machstamm und die von ihm ausgehende Scherschicht sind gut aufgelöst. Abbildung 6.24 zeigt den Verlauf

des Drucks, der von der Scherschicht nur leicht beeinflusst wird. Abbildung 6.25 zeigt das zugehörige Gitter mit 15424 Kontrollvolumen. Die Abbildungen zeigen zusätzlich eine Ausschnittsvergrößerung in der Gegend des Tripelpunktes.

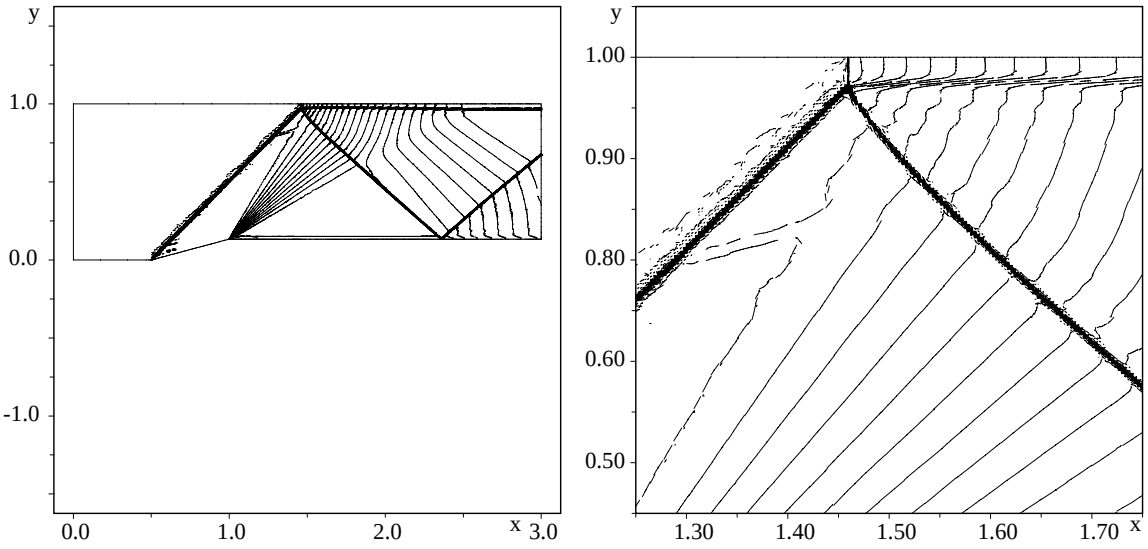


Abbildung 6.23: Mach-Isolinien für Kanal mit 15° Rampe, $M_\infty = 2.0$

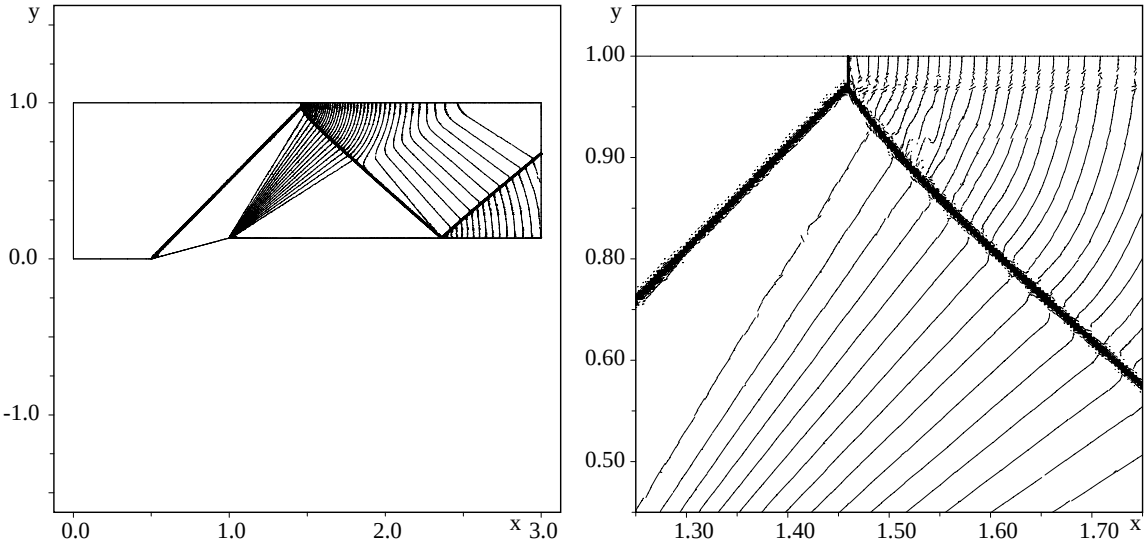


Abbildung 6.24: Druck-Isolinien für Kanal mit 15° Rampe, $M_\infty = 2.0$

6.3 Einfache Tests in 3D

Die folgenden drei Abbildungen zeigen die ersten Versuche zur Behandlung von dreidimensionalen Rechengebieten. Da die Grafikwerkzeuge noch nicht angepasst sind, lassen sich nur relativ einfache Oberflächenbilder und Gitteransichten darstellen. Diese werden auf Grund der großen Maschenzahl aber schnell unübersichtlich.

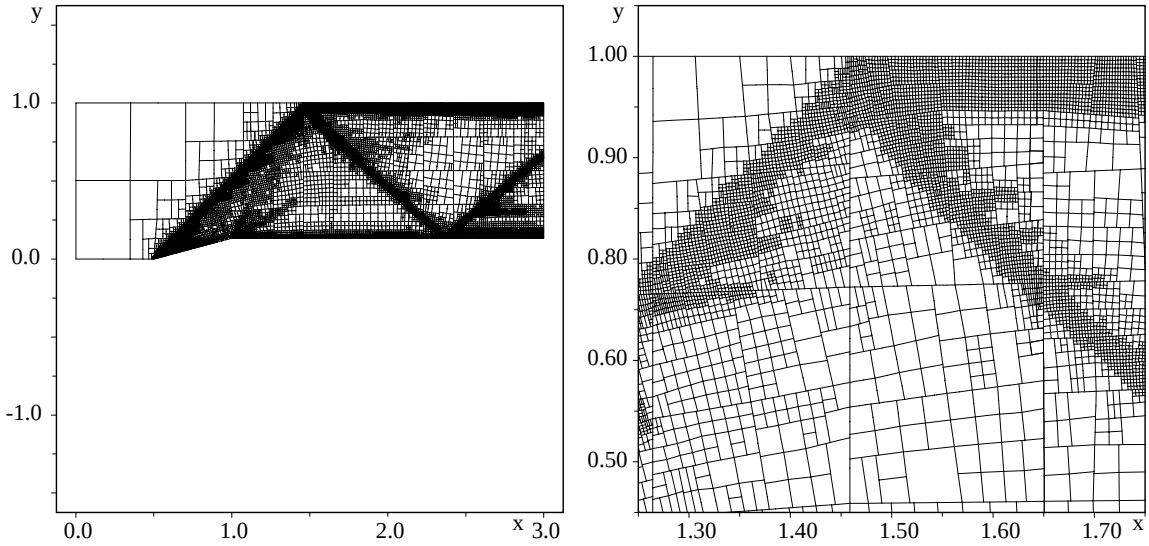


Abbildung 6.25: Gitterausschnitt für Kanal mit 15° Rampe, $M_\infty = 2.0$

6.3.1 Kanal mit zwei Keilen

Das erste Bild zeigt einen Kanal, der sich an zwei zueinander senkrechten Seiten durch eine 15°-Rampe verjüngt. Das Oberflächengitter zeigt die Spur der ersten Adaptionen an den Kompressions- und Expansionskanten der Keile. Da der Kanal im wesentlichen quaderförmig ist, treten bei der Gitterteilung nur einfach zu behandelnde Fälle auf.

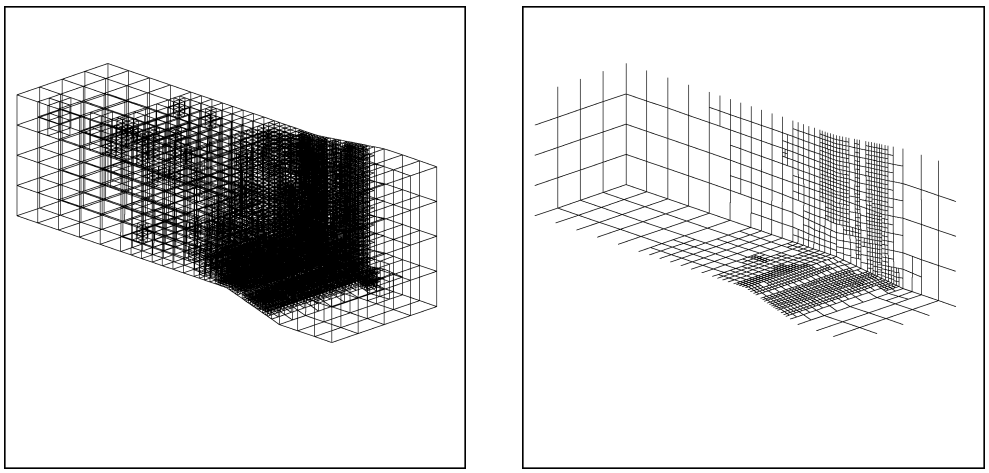


Abbildung 6.26: Gitterausschnitt für Kanal zwei 15° Rampen, $M_\infty = 3.0$

6.3.2 DLR-F5 Tragflügel im Windkanal

Abbildung 6.27 zeigt einen DLR-F5 Tragflügel, der an die Wand eines Windkanals montiert ist. Das Gitterbild zeigt die im entstehen begriffene Verfeinerung in der Nähe der Tragflügel- und Windkanaloberflächen.

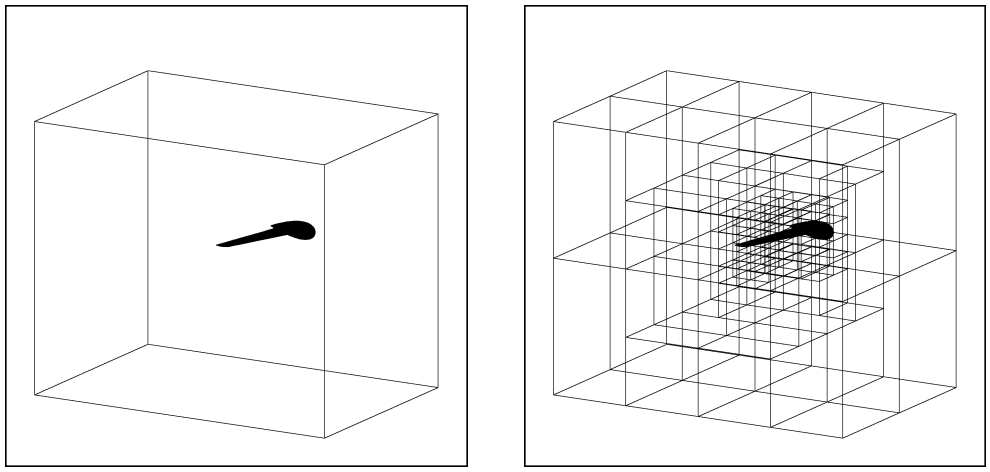


Abbildung 6.27: Anfangsgitter und Detail der Oberflächenverfeinerung für DLR-F5 Tragflügel in Windkanal

6.3.3 Flugzeugkonfiguration

Abbildung 6.28 zeigt eine Konfiguration aus Rumpf, Tragflügel, Triebwerksaufhängung und -gondel und die durch die Oberflächenverfeinerung eingefügten Gitterkanten.

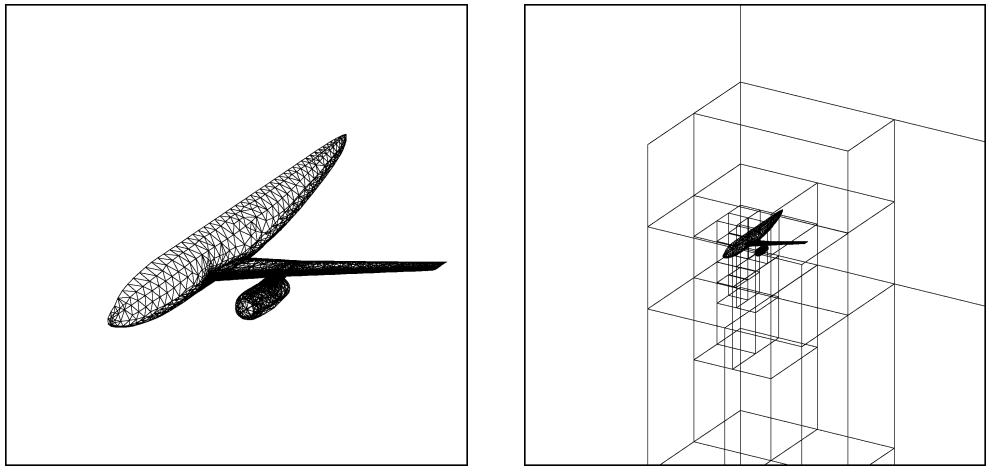


Abbildung 6.28: Anfangsgitter und Detail der Oberflächenverfeinerung für Flugzeugkonfiguration

Kapitel 7

Zusammenfassung und Ausblick

Zusammenfassung

Es wurde ein Finite-Volumen-Verfahren zur numerischen Simulation reibungsfreier Strömungen auf polygonal (bzw. polyhedral) berandeten Maschen vorgestellt.

Das Verfahren zeichnet sich dadurch aus, dass die Gittergenerierung in den Lösungsprozess integriert ist. Die zur Simulation notwendigen Gitter werden aus der Oberflächenbeschreibung des Rechengebietes in selbst-adaptiver Weise, ohne Eingriff des Anwenders, zusammen mit der Lösung erzeugt.

Die Integration beruht auf einer höchst flexiblen Gitterstruktur, die polygonale bzw. polyhedrale Maschen mit beliebig vielen Oberflächen unterstützt. So kann beispielsweise das Untersuchungsgebiet nach Diskretierung seiner gekrümmten Oberflächen direkt als Masche repräsentiert werden. Es wurde ein Algorithmus entwickelt, der Maschen durch Schnittebenen verfeinert, ohne jedoch beliebig kleine Teile zu erzeugen. Benachbarte Maschen können durch Entfernen gemeinsamer Seitenflächen vereinigt werden.

Mit Hilfe neuer, problemangepasster Methoden für die Rekonstruktion der Lösung aus den Zellmittelwerten können auch auf automatisch erzeugten Gittern genaue Lösungen berechnet werden. Dazu wird die Rekonstruktion in transformierten Variablen berechnet, die unter allen Umständen physikalisch sinnvolle Werte liefert. Die diskreten Ableitungen werden so modifiziert, dass die Rekonstruktion die stationäre Differentialgleichung erfüllt. Das Verfahren wurde für 2D Referenzfälle intensiv getestet.

Das numerische Verfahren und die Gitteradaption wurden auf Rechnern mit verteiltem Speicher implementiert.

Ausblick

Die vorgestellte Methode kann auch auf 3D Geometrien angewendet werden. Erste Schritte dazu wurden bereits unternommen, die Implementierung der Gitterteilung für 3D-Maschen ist aber noch nicht abgeschlossen. Die Versuche zeigten, dass eine ausgefeilte Grafikunterstützung und der Speicherbedarf für die Anwendung des Verfahrens in 3D eine wesentlich höhere Bedeutung hat als in 2D.

Eine andere wichtige Entwicklungsrichtung stellt die Anwendung des Verfahrens auf reibungsbehaftete Strömungen dar. Im Rahmen einer Studienarbeit [20] wurden die Diskretisierung der Konvektions-Diffusions-Gleichung auf polygonalen Maschen untersucht. Die Anwendung des Verfahrens auf die Navier-Stokes-Gleichungen und die Implementierung von Turbulenzmodellen stehen aber noch aus.

Eine wichtige Herausforderung stellt die Verkürzung der Rechenzeiten dar. In einer weiteren Dissertation wird gegenwärtig die Anwendung von Mehrgittermethoden untersucht [62]. Auch die Optimierung des numerischen Verfahrens für Vektorrechner kann dazu einen Beitrag liefern.

Für die Robustheit der Methode spielen zuverlässige Adaptionsindikatoren eine wichtige Rolle. Insbesondere in der Anfangsphase auf sehr groben Gittern sind die bisher angewendeten Standardmethoden nicht sehr zuverlässig. Die Ermittlung notwendiger Bedingungen an die Maschengeometrie ist deshalb von besonderem Interesse.

Anhang A

Integrationsformeln

Für ein Finite Volumen Verfahren höherer Ordnung werden verschiedene Integrationsformeln benötigt. Aufgrund der geometrischen Komplexität der vorgeschlagenen Kontrollvolumen ist die Berechnung der notwendigen Größen in $2D$ und $3D$ nicht trivial und soll im Folgenden erläutert werden. Die Darstellung beschränkt sich auf den Fall eben-berandeter Kontrollvolumen in 2D und 3D, wie er für 2D-Geometrien typisch ist. Der weitaus schwierigere Fall nicht-ebener Oberflächen, wie sie in 3D auftreten können, wird in [31] behandelt.

Für das explizite Zeitschrittverfahren Glg. 4.9

$$\bar{\varphi}_j^{n+1} = \bar{\varphi}_j^n - \frac{\triangle t}{\lambda_j} \cdot \sum_{k \in N_j} \int_{\partial\Omega_{jk}} F^*(\varphi_j^n(\vec{x}), \varphi_k^n(\vec{x}), \vec{s}_{jk})\, \mathbf{d}\lambda$$

wird die Größe des Kontrollvolumens λ_j benötigt, zur Flußberechnung der nach außen gerichtete Oberflächennormalenvektor $\vec{s}_{jk}$ sowie eine Quadraturformel für das Flußintegral $\int_{\partial\Omega_k} F^*\, \mathbf{d}\lambda$.

In die Formeln zur Ermittlung einer polynomialen Rekonstruktion p-ter Ordnung gehen alle Momente bis zur Ordnung p

$$\begin{aligned} m^{(0)} &= |\Omega| = \int_\Omega 1\, \mathbf{d}\lambda \in R \\ m^{(1)} &= \frac{1}{|\Omega|} \int_\Omega \vec{x}\, \mathbf{d}\lambda \in R^d \\ &\dots \end{aligned}$$

ein. Siehe dazu auch die Darstellung im Abschnitt 4.3.

Eine Strategie zur Berechnung der gewünschten Größen besteht darin, die gesuchten Integrale mit Hilfe der Integralsätze von Stokes und Gauß durch Integrale auf den Oberflächen des Kontrollvolumens darzustellen.

A.1 Integralnotation

Sei $\Omega \subset R^d$ eine offene, beschränkte und zusammenhängende Teilmenge des Raumes der Dimension d, deren Rand $\partial\Omega$ durch eine endliche Zahl stetig differenzierbarer, $d-1$-dimensionaler Mannigfaltigkeiten beschrieben ist.

Die Schreibweise

$$\int_\Omega \varphi \, \mathbf{d}\lambda = \int_\Omega \varphi(\vec{x}) \, \mathbf{d}\lambda(\vec{x}) \tag{A.1}$$

bezeichnet das Integral einer reellwertigen Funktion $\varphi : \bar{\Omega} \to R$, $\vec{x} \mapsto \varphi(\vec{x})$ über die Menge Ω, mit den Momenten

$$\begin{aligned} m^{(0)} &= |\Gamma| = \int_\Gamma 1 \, \mathbf{d}\mu \in R \\ m^{(1)} &= \frac{1}{|\Gamma|} \int_\Gamma \vec{x} \, \mathbf{d}\mu \in R^d \\ &\dots \end{aligned}$$

Das Integral $\int_\Gamma \cdot dy$ über ein stetig differenzierbares Teilstück $\Gamma \subset \partial\Omega$ des Randes ist durch

$$\int_\Gamma \varphi(\vec{x}) \, \mathbf{d}\mu := \int_{\Psi = \kappa^{-1}(\Gamma)} \varphi(\kappa(\vec{y})) \sqrt{\det(D\kappa^t(\vec{y}) \cdot D\kappa(\vec{y}))} \, \mathbf{d}\lambda(\vec{y}) \tag{A.2}$$

gegeben. Dabei beschreibt die bijektive, stetig differenzierbare Abbildung $\kappa : \Psi \to \Gamma$, $\vec{y} \mapsto \kappa(\vec{y}) = \vec{x}$ ($\Psi := T^{-1}(\Gamma) \subset R^{d-1}$) die Fläche Γ. Ist Γ eine ebene Hyperfläche, so ist κ eine affine Abbildung, d. h. $D\kappa = const.$ In diesem Fall ist $\sqrt{\det(D\kappa^t(\vec{y}) \cdot D\kappa(\vec{y}))}$ konstant und beschreibt die Größe (Fläche, Länge) der Hyperfläche Γ.

Mit $\int_\Gamma \cdot \, \mathbf{d}\sigma$ wird das Integral einer vektorwertigen Funktion $\Phi : \bar{\Omega} \to R^d$ mit nach aussen gerichteter Oberflächennormale $\sigma(\vec{x})$,

$$\int_\Gamma \Phi \, \mathbf{d}\sigma := \int_\Gamma \langle \Phi(\vec{x}), \sigma(\vec{x}) \rangle \, \mathbf{d}\mu \tag{A.3}$$

bezeichnet.

A.2 Bestimmung des Oberflächennormalenvektors

Aus dem Stokeschen Integralsatzes für eine ebene, polygonal berandete Fläche $f \in R^3$ und ihren Rand ∂f

$$\int_f \mathrm{rot}(\vec{v}) \, \mathbf{d}\sigma = \int_{\partial f} \vec{v} \, \mathbf{d}\mu \tag{A.4}$$

kann unter Verwendung geeigneter Vektorfelder $\vec{v}$ eine Formel für den Oberflächennormalenvektor abgeleitet werden. Für die Vektorfelder $\vec{v}_i$, $(i = 1, 2, 3)$

$$\vec{v}_i(\vec{x}) = \langle \vec{x}, \vec{e}_{i+1} \rangle \cdot \vec{e}_{i+2} \tag{A.5}$$

erhält man

$$\mathrm{rot}(\vec{v}_i) = \vec{e}_i.$$

Dabei bezeichnet $\vec{e}_i$ den i-ten Einheitsvektor, für die Indizes $i > 3$ sei $\vec{e}_i = \vec{e}_{i-3}$ gesetzt.

Mit Hilfe der Vektorfelder aus Glg. A.5 läßt sich aus Glg. A.4 die mittlere Oberflächennormale $\vec{s}(f)$ bestimmen, welche im Fall der ebenen Fläche f mit der Oberflächennormale

übereinstimmt. Ist die Fläche polygonal berandet, so kann das Kurvenintegral auf die polygonalen Randstücke ∂f_k aufgeteilt werden.

$$\begin{aligned} \vec{s}_i(f) &= \int_f \mathrm{rot}(\vec{v}_i)\,\mathbf{d}\sigma = \sum_{k\in S(f)} \int_{\partial f_k} \vec{v}_i\,\mathbf{d}\mu \\ &= \sum_{k\in S(f)} m^{(1)}_{i+1}(\partial f_k)\cdot \vec{s}_{i+2}(\partial f_k) \end{aligned} \tag{A.6}$$

Dabei bezeichnet $S(f)$ die Indexmenge aller Oberflächenobjekte der Facette f, $\vec{s}(\partial f_k)$ den orientierten Tangentialvektor der Kante f_k mit der Länge $m^{(0)}(\partial f_k)$ und es gilt $||\vec{s}(f)|| = m^{(0)}(f)$.

A.3 Darstellung eines Monoms als Divergenz

Eine Strategie zur Berechnung der benötigten Integrale besteht darin, die gesuchten Integrale mit Hilfe der Integralsätze von Stokes und Gauß und einer "Stammfunktion" des Integranden zu ermitteln.

[Lemma: Darstellung eines Monoms als Divergenz.] Sei $\alpha = (\alpha_1, ..., \alpha_d) \in N_0^d$ ein Multi-Index ($d \in N$) und $|\alpha| = \sum_{i=1...d} \alpha_i$ sein Betrag. Das Polynom p_α

$$p_\alpha : R^d \to R, \quad \vec{x} \mapsto \vec{x}^\alpha = \prod_{i=1..d} x_i^{\alpha_i}$$

ist das zu α gehörige Monom. Das Monom p_α besitzt die Darstellung

$$p_\alpha(\vec{x}) = \frac{1}{|\alpha| + d}\mathrm{div}\,(p_\alpha(\vec{x})\cdot\vec{x}) \tag{A.7}$$

Die behauptete Darstellung läßt sich, ausgehend von der rechten Seite der Identität, durch einfaches differenzieren verifizieren. Mit Hilfe der Produktregel erhält man

$$\mathrm{div}\,(p_\alpha(\vec{x})\cdot\vec{x}) = \sum_{i=1..d} \partial_i\,(\vec{x}^\alpha \cdot x_i) = \sum_{i=1..d} \partial_i\,(\vec{x}^\alpha)\cdot x_i + \sum_{i=1..d} \vec{x}^\alpha \cdot \partial_i\,(x_i)$$

Der erste Term ergibt nach wiederholter Anwendung der Produktregel

$$\partial_i\,(\vec{x}^\alpha)\,x_i = \partial_i\left(\prod_{r=1..d} x_r^{\alpha_r}\right)\cdot x_i = \partial_i\,(x_i^{\alpha_i})\cdot x_i \prod_{r\neq i} x_r^{\alpha_r} = \alpha_i x_i^{\alpha_i}\prod_{r\neq i} x_r^{\alpha_r} = \alpha_i \vec{x}^\alpha \tag{A.8}$$

Damit ergibt sich die behauptete Identität

$$\mathrm{div}\,(p_\alpha(\vec{x})\cdot\vec{x}) = \sum_{i=1..d} \alpha_i\vec{x}^\alpha + \sum_{i=1..d} \vec{x}^\alpha = (|\alpha| + d)\cdot\vec{x}^\alpha$$

A.4 Volumen- und Schwerpunktsformeln

Die Momente eines Kontrollvolumens lassen sich unter Anwendung des Gauß'schen Integralsatzes

$$\int_\Omega \operatorname{div}(\vec{v})\ \mathbf{d}\lambda = \int_{\partial\Omega} \vec{v}\,\mathbf{d}\sigma.$$

und der Darstellung eines Monoms als Divergenz, Glg. A.7 ermitteln. Für das Volumen $m^{(0)}$ eines eben-berandeten Kontrollvolumens Ω gilt:

$$m^{(0)}(\Omega) = \int_\Omega 1\,\mathbf{d}\lambda = \frac{1}{d}\int_{\partial\Omega} \vec{x}\,\mathbf{d}\sigma$$

Das Oberflächenintegral läßt sich auf die Oberflächenstücke $\partial\Omega_k$ des Kontrollvolumens aufteilen. Da diese als eben angenommen wurden, ist die nach außen gerichtete Oberflächennormale $\mathbf{d}\sigma(\vec{x})$

$$\mathbf{d}\sigma(\vec{x}) = \frac{1}{|\partial\Omega_k|}\vec{s}(\partial\Omega_k), \quad \forall \vec{x} \in \partial\Omega_k$$

auf $\partial\Omega_k$ konstant. Darüberhinaus variiert $\vec{x} \in \partial\Omega_k$ nur senkrecht zur Normalen so, dass das Skalarprodukt ebenfalls konstant ist und an einem beliebigen Punkt $\vec{x} \in \partial\Omega_k$ ausgewertet werden darf. Bezeichnet man die Indexmenge aller Seiten des Kontrollvolumens Ω mit $S(\Omega)$, so erhält man für das Volumen

$$m^{(0)}(\Omega) = \frac{1}{d}\sum_{k\in S(\Omega)} \langle m^{(1)}(\partial\Omega_k), \vec{s}(\partial\Omega_k)\rangle \tag{A.9}$$

Analog läßt sich für den Schwerpunkt des Kontrollvolumens

$$\begin{aligned}(d+1)|\Omega|\cdot m_i^{(1)}(\Omega) &= (d+1)\cdot\textstyle\int_\Omega \vec{x}_i\,\mathbf{d}\lambda = \int_\Omega \operatorname{div}(\vec{x}_i\cdot\vec{x})\,\mathbf{d}\lambda \\ &= \textstyle\int_{\partial\Omega}(\vec{x}_i\cdot\vec{x})\,\mathbf{d}\sigma = \sum_{k\in S(\Omega)} \langle m^{(1)}(\partial\Omega_k), \vec{s}(\partial\Omega_k)\rangle \cdot \frac{1}{|\partial\Omega_k|}\int_{\partial\Omega_k}\vec{x}_i\,\mathbf{d}\mu\end{aligned}$$

ermitteln und es ergibt sich

$$m^{(1)}(\Omega) = \frac{1}{(d+1)|\Omega|}\sum_{k\in S(\Omega)} \langle m^{(1)}(\partial\Omega_k), \vec{s}(\partial\Omega_k)\rangle \cdot m^{(1)}(\partial\Omega_k) \tag{A.10}$$

A.5 Integrale für die Flussquadratur

Mit Hilfe der Integrale über Monome

$$\int_{\partial\Omega} (p_\alpha(\vec{x})\cdot\vec{e}_i)\ \mathbf{d}\sigma$$

sollen die Überlegungen zur Konsistenz der Flussintegration in Abschnitt 4.2.4 vervollständigt werden. Wie in Glg. A.8 ermittelt man für $\alpha_i = 0$

$$\begin{aligned}0 &= \int_\Omega \operatorname{div}(\vec{x}^\alpha\,\vec{e}_i)\ \mathbf{d}\lambda = \int_{\partial\Omega}(\vec{x}^\alpha\,\vec{e}_i)\ \mathbf{d}\sigma \\ &= \sum_{k\in S(\Omega)} s_{k,i}\cdot\frac{1}{|\partial\Omega_k|}\int_{\partial\Omega_k}\vec{x}^\alpha\,\mathbf{d}\mu\end{aligned}$$

So ergibt sich für die Monome $p_\alpha = 1$, $p_\alpha = x_n$ und $p_\alpha = x_n x_m$ $(n, m \neq i)$

$$\begin{aligned} 0 &= \sum_{k \in S(\Omega)} s_{k,i} \cdot 1 \\ 0 &= \sum_{k \in S(\Omega)} s_{k,i} \cdot m_n^{(1)}(\partial\Omega_k) \\ 0 &= \sum_{k \in S(\Omega)} s_{k,i} \cdot m_{nm}^{(2)}(\partial\Omega_k) \end{aligned}$$

Außerdem erhält man

$$\begin{aligned} (1+\alpha_i) \int_\Omega \vec{x}^\alpha \, \mathbf{d}\lambda &= \int_\Omega \operatorname{div}(\vec{x}^\alpha \, x_i \, \vec{e}_i) \, \mathbf{d}\lambda = \int_{\partial\Omega} (\vec{x}^\alpha \, x_i \, \vec{e}_i) \, \mathbf{d}\sigma \\ &= \sum_{k \in S(\Omega)} \langle \vec{e}_i, \vec{s}_k \rangle \frac{1}{|\partial\Omega_k|} \int_{\partial\Omega_k} \vec{x}^\alpha \, x_i \, \mathbf{d}\mu \\ &= \sum_{k \in S(\Omega)} m_i^{(1)}(\partial\Omega_k) s_{k,i} \frac{1}{|\partial\Omega_k|} \int_{\partial\Omega_k} p_\alpha(\vec{x}) \, \mathbf{d}\lambda \end{aligned}$$

und es ergibt sich für die Monome $p_\alpha = 1$, $p_\alpha = x_n$ $(n \neq i)$ und $p_\alpha = x_i$

$$\begin{aligned} |\Omega| &= \sum_{k \in S(\Omega)} s_{k,i} \cdot m_i^{(1)}(\partial\Omega_k) \\ |\Omega| \, m_n^{(1)}(\Omega) &= \sum_{k \in S(\Omega)} s_{k,i} \cdot m_{in}^{(2)}(\partial\Omega_k) \\ |\Omega| \, m_i^{(1)}(\Omega) &= \frac{1}{2} \sum_{k \in S(\Omega)} s_{k,i} \cdot m_{ii}^{(1)}(\partial\Omega_k) \end{aligned}$$

Zusammen ergibt sich also

$$0 = \sum_{k \in S(\Omega)} s_{k,i} \tag{A.11}$$

$$|\Omega| \, \vec{e}_i = \sum_{k \in S(\Omega)} s_{k,i} \, m^{(1)}(\partial\Omega_k) \tag{A.12}$$

$$|\Omega| \left(m^{(1)}(\Omega) \, \vec{e}_i^{\,t} + \vec{e}_i \, m^{(1)}(\Omega)^t \right) = \sum_{k \in S(\Omega)} s_{k,i} \, m^{(2)}(\partial\Omega_k) \tag{A.13}$$

Anders als bei der Herleitung der Volumen- und Schwerpunktsformeln (Glgen. A.9, A.10) erniedrigt hier die Multiplikation mit den Komponenten der Seitennormale den Exponenten des Integranden *nicht*. Diese Bemerkung führt zu den Identitäten $(i = 1..d)$

$$\begin{aligned} \frac{1}{d} \sum_{k \in S(\Omega)} \langle m(\partial\Omega_k), \vec{s}(\partial\Omega_k) \rangle &= \sum_{k \in S(\Omega)} m_i^{(1)}(\partial\Omega_k) \vec{s}_i(\partial\Omega_k) \\ \frac{2}{d+1} \sum_{k \in S(\Omega)} \langle m^{(1)}(\partial\Omega_k), \vec{s}_k \rangle m_i^{(1)}(\partial\Omega_k) &= \sum_{k \in S(\Omega)} m_{ii}^{(2)}(\partial\Omega_k) \vec{s}_i(\partial\Omega_k) \end{aligned}$$

Dabei bezeichnet $m(\partial\Omega_k)$ einen beliebigen Punkt auf der Oberfläche der Seite $\partial\Omega_k$.

Anhang B

Nomenklatur

d	Raumdimension
$\vec{e}_i$	i-ter Einheitsvektor
i	Index, der die Raumdimensionen $1 \ldots d$ durchläuft.
Id	Einheitsmatrix
j	Index, der das betrachtete Kontrollvolumen identifiziert.
k	Index, der die zu j benachbarten Kontrollvolumen durchläuft.
$\kappa : \Psi \to \Gamma$	beschreibende Abbildung der Hyperfläche $\Gamma \subset R^d$, mit Parametermenge $\Psi \subset R^{d-1}$
m	Dimension des Zustandsraumes
$m^{(0)}(\Omega)$	nulltes Moment (Größe) des Kontrollvolumens $\Omega \subset R^d$
$m^{(0)}(\Gamma)$	nulltes Moment (Größe) der Hyperfläche $\Gamma \subset R^d$
$m^{(1)}(\Omega)$	erstes Moment (Schwerpunkt) des Kontrollvolumens $\Omega \subset R^d$
$m^{(1)}(\Gamma)$	erstes Moment (Schwerpunkt) der ebenen Hyperfläche $\Gamma \subset R^d$
n	Anzahl der Integrationspunkte der Seitenquadraturformel
N_j	Indexmenge der Kontrollvolumen, die an das Kontrollvolumen j angrenzen.
$\bar{N}_j$	$N_j \cup \{j\}$
$\Omega, \partial\Omega$	Kontrollvolumen und seine Oberfläche, $(\Omega, \partial\Omega \subset R^d)$
Ω_j	Kontrollvolumen
$\partial\Omega_{jk}$	Seite zwischen Ω_j und Ω_k
$\vec{s}_{jk}$	Oberflächennormalenvektor der Seite $\partial\Omega_{jk}$, bzgl. Ω_j auswärts gerichtet, $\|\|\vec{s}_{jk}\|\| = \|\partial\Omega_{jk}\|$
$S(o)$	(Index-) Menge aller Oberflächenobjekte des Gitterobjektes o

Literaturverzeichnis

[1] R. Abgrall and F.C. Lafon. ENO schemes on unstructured meshes. In *Lecture Series: Computational Fluid Dynamics, 1993-04*, Rhode St. Genese, Belgium, March 15-19 1993. Van Karman Institute for Fluid Dynamics.

[2] R. Abgrall, T. Sonar, O. Friedrich, and G. Billet. High order approximations for compressible fluid dynamics on unstructured and cartesian meshes. In *High Order Methods for Computational Physics*, Lecture Notes in Computational Science and Engineering. Springer Verlag, 1999.

[3] Test cases for inviscid flow field methods. Technical Report AR-211, AGARD, May 1985.

[4] G.D. Van Albada, B. Van Leer, and W.W. Roberts. A comparativve study of computational methods in cosmic gas dynamics. *Astronomy and Astrophysics*, 108:76–84, 1982.

[5] T.J. Barth. On unstructured grids and solvers. In *Lecture Series: Computational Fluid Dynamics, 1990-03*, Rhode St. Genese, Belgium, March 5-9 1990. Van Karman Institute for Fluid Dynamics.

[6] T.J. Barth. Aspects of unstructured grids and finite-volume solvers for the Euler and Navier-Stokes equations. In *Special Course on Unstructured Grid Methods for Advection Dominated Flows*, AGARD Report R-787, Neuilly Sur Seine, France, May 1992.

[7] T.J. Barth. Aspects of unstructured grids and finite-volume solvers for the Euler and Navier-Stokes equations. In *Lecture Series: Computational Fluid Dynamics, 1994-05*, Rhode St. Genese, Belgium, March 21-25 1994. Van Karman Institute for Fluid Dynamics.

[8] P. Bastian. *Parallele adaptive Mehrgitterverfahren.* PhD thesis, Universität Heidelberg, 1994.

[9] K. Birken. *Ein Modell zur effizienten Parallelisierung von Algorithmen auf komplexen, dynamischen Datenstrukturen.* PhD thesis, Fachbereich Energietechnik, Universität Stuttgart, Deutschland, 1998.

[10] G. Booch. *Object-Oriented Design With Applications, 2nd Edition.* The Benjamin/Cummings Publishing Company, 1994.

[11] J.E. Castillo and M. Shashkov. Grids consistent with finite difference schemes. In N.P. Weatherill, P.R. Eiseman, J. Häuser, and J.F. Thompson, editors, *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, pages 73–84, Swansea, UK, April 1994. Pineridge Press.

[12] J.O. Coplien. *Advanced C++ Programming Styles and Idioms.* Addison–Wesley, 1992.

[13] R. Courant and D. Hilbert. *Methods Of Mathematical Physics, Volume II.* Wiley & Sons, 1989.

[14] A. Dauth and U. Küster. Gitterglättung und Limitierung auf lösungsabhängigen, allgemeinen Zellen. Technical report, Rechenzentrum, Universität Stuttgart, März 1999.

[15] F. Deister and E.H. Hirschel. Adaptive cartesian/prism grid generation and solutions for arbitrary geometries. *AIAA-Paper*, 99-0782, 1999.

[16] F. Deister, D. Rocher, E.H. Hirschel, and F. Monnoyer. Three-dimensional adaptively refined cartesian grid generation and euler flow solutions for arbitrary geomtries. In *Fourth European Computational Fluid Dynamics Conference*, pages 96–101. John Wiley and Sons, 1998.

[17] P.R. Eiseman, Z. Cheng, and J. Häuser. Applications of multiblock grid generation with automatic zoning. In N.P. Weatherill, P.R. Eiseman, J. Häuser, and J.F. Thompson, editors, *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, pages 123–134, Swansea, UK, April 1994. Pineridge Press.

[18] P.R. Eiseman, N. Lu, M. Jiang, and J.F. Thompson. Algebraic-elliptic grid generation. In N.P. Weatherill, P.R. Eiseman, J. Häuser, and J.F. Thompson, editors, *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, pages 37–44, Swansea, UK, April 1994. Pineridge Press.

[19] W.K. Liu et al. Overview and applications of the reproducing kernel particle methods. In *Archives of Computational Methods in Engineering, State of the Art Reviews*, 1995.

[20] M. Fischer. Erweiterung eines Finite-Volumen-Verfahrens um Diffusionsterme für die skalare Konvektionsgleichung auf polygonalen Kontrollvolumen. Technical report, Rechenzentrum, Universität Stuttgart, 1996.

[21] M. Galle. *Ein Verfahren zur numerischen Simulation kompressibler, reibungsbehafteter Strömungen auf hybriden Netzen.* PhD thesis, Fachbereich Energietechnik, Universität Stuttgart, 1999.

[22] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns – Elements of Reusable Object–Oriented Software.* Professional Computing Series. Addison–Wesley, 1995.

[23] A. Harten, S. Engquist, S. Osher, and S. Chakravarthy. Uniformly high-order accurate essentially non-oscillatory shock-capturing schemes III. *Journal of Computational Physics*, 71:231–303, 1987.

[24] C. Helf and U. Küster. A finite volume method with arbitrary polygonal control volumes and high order reconstruction for the Euler equations. In S. Wagner, E.H. Hirschel, J. Périaux, and R. Piva, editors, *Proceedings of the Second European Computational Fluid Dynamics Conference*, Stuttgart, Germany, 1994. Wiley & Sons.

[25] C. Hirsch. *Numerical Computation of Internal and External Flows – Volume 2: Computational Methods for Inviscid and Viscous Flows.* Wiley & Sons, 1990.

[26] C. Hirsch. *Numerical Computation of Internal and External Flows – Volume 1: Fundamentals of Numerical Discretization.* Wiley & Sons, 1991.

[27] J.D. Hoffmann. High resolution schemes for hyperbolic conservation laws. *Journal of Computational Physics*, 46:469–474, 1982.

[28] O.-P. Jacquotte and G. Coussement. Structured grid variational adaption: Reaching the limit ? In *First European Computational Fluid Dynamics Conference*, pages 1077–1087. Elsevier, 1992.

[29] Y. Kallinderis, A. Kahawaja, and H. McMorris. Hybrid grid generation for complex 3-D geometries. In *Proc. 4th. European Computational Fluid Dynamics Conference*, pages 923–928. John Wiley and Sons, 1998.

[30] G. Krause. Automatic mesh generation based on CAD data. In N.P. Weatherill, P.R. Eiseman, J. Häuser, and J.F. Thompson, editors, *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, pages 329–340, Swansea, UK, April 1994. Pineridge Press.

[31] U. Küster. Interpolierende Funktionen in mehrdimensionalen Räumen. manuscript, RUS, Stuttgart, 1999.

[32] U. Küster and Manuela Zürn. Influence of Fortran90 features on performance on Cray vector computer systems. In *High Performance Computing and Networking*, volume 797 of *Lecture Notes in Computer Science*. Springer Verlag, ?

[33] M.H. Lallemand, H. Steve, and A. Dervieux. Unstructured multigridding by volume agglomeration: current status. *Comput. Fluids*, 21:397–433, 1992.

[34] B. Van Leer. Towards the ultimate conservative difference scheme, V: A second order sequel to Godunov's method. *Journal of Computational Physics*, 32:101–136, 1979.

[35] R. Löhner. Finite element methods in CFD: Grid generation, adaptivity and parallelization. In *Special Course on Unstructured Grid Methods for Advection Dominated Flows*, AGARD Report R-787, Neuilly Sur Seine, France, May 1992.

[36] R. Löhner and J.D. Baum. Adaptive h-refinement on unstructured grids for transient problems. *International Journal for Numerical MEthods in Fluids*, 14:1407–1419, 1992.

[37] Y. Maday. Introduction to spectral methods. In *Lecture Series: Computational Fluid Dynamics, 1993-04*, Rhode St. Genese, Belgium, March 15-19 1993. Van Karman Institute for Fluid Dynamics.

[38] D.J. Mavriplis. Unstructured mesh generation and adaptivity. ICASE Report 95-26, Institute for Computer Applcations in Science and Engineering, NASA Langley Research Center, Hampton, VA 23681-0001, USA, April 1995.

[39] D.J. Mavriplis and V. Venkatakrishnan. Agglomeration multigrid for viscous turbulent flows. ICASE Report 94-62, Institute for Computer Applcations in Science and Engineering, NASA Langley Research Center, Hampton, VA 23681-0001, USA, July 1994.

[40] N.P. Weatherill M.J. Marchant. Unstructured grid generation for viscous flow simulations. In N.P. Weatherill, P.R. Eiseman, J. Häuser, and J.F. Thompson, editors, *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, pages 151–162, Swansea, UK, April 1994. Pineridge Press.

[41] K. Morinishi. An implicit gridless type solver for the Navier–Stokes equations. In *7th Symposium on Computational Fluid Dynamics*. to appear, 1999.

[42] E. Oñate and S. Idelsohn. A mesh-free finite point method for advective-diffusive transport and fluid flow problems. *Computational Mechanics*, 21:283–292, 1998.

[43] M. Pandolfi and D. D'Ambrosio. Upwind methods and carbuncle phenomenon. In *Fourth European Computational Fluid Dynamics Conference*, pages 126–131. John Wiley & Sons, 1998.

[44] B. Perthame. An introduction to kinetic schemes for gas dynamics. In *An Introduction to Recent Developments in Theory and Numerics for Conservation Laws*, Lecture Notes in Computational Science and Engineering. Springer Verlag, 1999.

[45] J.J. Quirk. *An Adaptive Grid Algorithm For Computational Shock Hydrodynamics*. PhD thesis, College of Aeronautics, Cranfield Institute of Technology, January 1991.

[46] A.W. Reichert. *Strömungssimulationen zur optimalen Gestaltung von Turbomaschinenkomponenten*. PhD thesis, Fachbereich Maschinenbau, Gerhard-Mercator-Universität Gesamthochschule, Düsseldorf, 1994.

[47] M.G. Remotigue and the NGP Team. The national grid project: Making dreams into reality. In N.P. Weatherill, P.R. Eiseman, J. Häuser, and J.F. Thompson, editors, *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, pages 429–439, Swansea, UK, April 1994. Pineridge Press.

[48] P.L. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43:357–372, 1981.

[49] J. Rokicki. Chimera method for compressible flows in complex geometries. In *7th Symposium on Computational Fluid Dynamics*. to appear, 1999.

[50] M. Rudgyard. Cell vertex methods for steady inviscid flow. In *Lecture Series: Computational Fluid Dynamics, 1993-04*, Rhode St. Genese, Belgium, March 15-19 1993. Van Karman Institute for Fluid Dynamics.

[51] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Objektorientiertes Modellieren und Entwerfen*. Prentice-Hall, 1991.

[52] R. Sanders. High order non-oscillatory schemes for the compressible Euler equations. In *Lecture Series: Computational Fluid Dynamics, 1993-04*, Rhode St. Genese, Belgium, March 15-19 1993. Van Karman Institute for Fluid Dynamics.

[53] J.A. Shaw, J.M. Georgala, and P.N. Childs. General procedures employed in the generation of three - dimensional hybrid structured/unstructured meshes. In N.P. Weatherill, P.R. Eiseman, J. Häuser, and J.F. Thompson, editors, *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, pages 297–303, Swansea, UK, April 1994. Pineridge Press.

[54] A.A. Shostko, R. Löhner, and W.C. Sandberg. Surface triangulation over intersection geometries. *International Journal for Numerical Methods in Engineering*, 44:1359–1376, 1999.

[55] D.T. Sihling. Adaptive reproducing kernel particle method for computational fluid dynamics. Technical report, Northwestern University, August 1995.

[56] T. Sonar and E. Sühli. A dual graph-norm refinement indicator for finite-volume approximations of the Euler equations. *Numerische Mathematik*, 78:619–658, 1998.

[57] B.K. Soni. Grid generation: Algebraic and partial differential equations techniques revisited. In *First European Computational Fluid Dynamics Conference*, pages 929–936. Elsevier, 1992.

[58] B. Stroustrup. *The C++ Programming Language*. Addison–Wesley, 1991.

[59] E. Sühli. A posteriori analysis and adaptivity for finite element approximations of hyperbolic systems. In *An Introduction to Recent Developments in Theory and Numerics for Conservation Laws*, Lecture Notes in Computational Science and Engineering. Springer Verlag, 1999.

[60] J.L. Thomas, R.W. Walters, B. Van Leer, and W.K. Anderson. Implicit finite-volume algorithms for the flux-split Euler equations. In *Proc. GAMM Workshop on Numerical Simulation of Compressible Euler Flows*. Vieweg, 1988.

[61] P. Vankeirsbilck. *Algorithmic Developments for the Solution of Hyperbolic Conservation Laws on Adaptive Unstructured Grids (Application to the Euler Equations)*. PhD thesis, Van Karman Institute, University Of Brussels, Brussels, Belgium, 1993.

[62] K. Warendorf. *Mehrgittermethoden für konvektiv dominierte partielle Differentialgleichungen auf vollständig unstrukturierten Gittern*. PhD thesis, Fachbereich Energietechnik, Universität Stuttgart, to be published 2000.

[63] N.P. Weatherill, P.R. Eiseman, J. Häuser, and J.F. Thompson, editors. *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, Swansea, UK, April 1994. Pineridge Press. Proceedings of the 4th International Conference.

[64] M. Wierse. *Higher Order Upwind Schemes on Unstructured Grids for the Compressible Euler Equations in Timedependent Geometries in 3D*. PhD thesis, Albert-Ludwigs-Universität, Freiburg i. Br., Germany, 1994.

[65] E. Yourdon. *Object-Oriented Systems Design: An Integrated Approach*. Yourdon Press, Prentice-Hall, 1994.

[66] H. Yserentant. A new class of particle methods. *Numerische Mathematik*, 76:87–109, 1997.

[67] H. Yserentant. Particles of variable size. *Numerische Mathematik*, 82:143–159, 1999.

[68] T.-Y. Yu and B.K. Soni. Geometry transformer and NURBS in grid generation. In N.P. Weatherill, P.R. Eiseman, J. Häuser, and J.F. Thompson, editors, *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, pages 353–364, Swansea, UK, April 1994. Pineridge Press.

[69] D.L. De Zeeuw. *A Quadtree-Based Adaptively Refined Cartesion Grid Algorithm For The Solution Of The Euler Equations.* PhD thesis, Aerospace Engineering and Scientific Computing, University Of Michigan, Michigan, USA, 1992.

Lebenslauf

29. Mai 1964	Clemens Helf, geboren in Erlangen, verheiratet, 2 Kinder.
1970 bis 1974	Grundschule in Erlangen.
1974 bis 1983	Besuch des Emmy-Noether-Gymnasiums in Erlangen.
Mai 1983	Abitur.
Juli 1983 – Oktober 1984	Wehrdienst in Cham und Amberg.
November 1984 – November 1989	Diplom-Studiengang Mathematik mit Nebenfach Informatik an der Friedrich-Alexander-Universität Erlangen.
November 1989	Diplom in Mathematik.
September 1990 – Januar 1992	Wissenschaftlicher Mitarbeiter an der Staatlichen Materialprüfungsanstalt (MPA) der Universität Stuttgart.
seit Februar 1992	Wissenschaftlicher Mitarbeiter am Rechenzentrum (RUS) der Universität Stuttgart / Höchstleistungsrechenzentrum Stuttgart (HLRS).

Homepage:	`http://www.hlrs.de/people/helf`
e-mail:	`helf@rus.uni-stuttgart.de` `helf@hlrs.de`