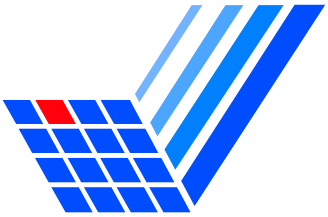
Geometrische Projektionstechniken auf Oberflächentriangulierungen zur numerischen Strömungssimulation mit hierarchischen Mehrgitterverfahren

– Diplomarbeit –

dem Fachbereich Informatik
der Universität Dortmund vorgelegt von

Dominik Göddeke



Universität Dortmund,
Institut für Angewandte Mathematik

Hiermit versichere ich, daß ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel verfaßt habe.

Dortmund, den 31. August 2004

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Einführung und Motivation

Strömungsvorgänge treten in der Natur in einer einer Vielzahl verschiedenster Situationen auf:

- meteorologische Vorgänge wie Wind- und Meeresströmungen
- Umweltverschmutzung
- Belüftungs- und Klimaanlagen
- Luft- und Benzinströmungen in Automotoren, Turbinen und Triebwerken
- Blut- und Luftströme im menschlichen Körper
- chemische Reaktionsvorgänge
- Umströmungen zur Analyse des Luft- bzw. Wasserwiderstandes komplexer Objekte wie Automobilen oder Schiffen

Typischerweise ist man in doppelter Hinsicht an der Untersuchung solcher Szenarien interessiert: Zum einen soll eine **qualitative** Analyse eine Visualisierung des Strömungsvorganges liefern, um beispielsweise Wirbelablösungen an den Tragflächen eines Flugzeugs genau lokalisieren zu können. Auf der anderen Seite sollen auch **quantitative** Aussagen möglich sein, für ein Fahrzeug soll beispielsweise der Luftwiderstand bestimmt werden, oder für ein mechanisches Bauteil der Anpreßdruck, bei dem es bricht.

1.1.1 Experiment und Simulation

Es gibt zwei grundsätzliche Vorgehensweisen, um Untersuchungen dieser Art durchzuführen: Im **Experiment** wird ein Modell der Wirklichkeit, häufig in verändertem Maßstab, konstruiert und dieses dann getestet: In der Automobilindustrie plaziert man beispielsweise während der Designphase ein verkleinertes Modelle eines Prototyps in einen Windkanal, um seinen

Luftwiderstand zu messen. In der **Simulation** geht man im Gegensatz dazu abstrakter vor, ein Simulationszyklus besteht aus vier wesentlichen Schritten:

1. Modellbildung: Ein physikalisches Phänomen wird durch eine Reihe von mathematischen Gleichungen beschrieben. Um alle Probleme mit einheitlichen Verfahren lösen zu können, werden die Modelle mit Hilfe von Differentialgleichungen formuliert.

2. Diskretisierung: Das Modell wird diskretisiert, d.h. in eine für den Computer verarbeitbare Form gebracht. Dies liefert typischerweise große (lineare) Gleichungssysteme.

3. Berechnung: Die Lösung dieser Gleichungssysteme liefert große Mengen an diskreten Rohdaten.

4. Aufbereitung: Mit Hilfe geeigneter Visualisierungstechniken werden die Rohdaten in eine für den Anwender interpretierbare Form gebracht.

Die wissenschaftliche Disziplin des CFD (engl. *computational fluid dynamics*) beschäftigt sich mit der Vorhersage von Strömungsvorgängen durch eben diese Simulationen. CFD ist ein interdisziplinäres Forschungsgebiet, es beinhaltet Aspekte aus so verschiedenen Bereichen wie den klassischen Naturwissenschaften, der angewandten Mathematik, der Informatik und den Ingenieurwissenschaften.
Die Simulation ersetzt das Experiment jedoch nicht. Simulationen sind zwar häufig billiger und schneller als Experimente, auf die Ergebnisse sollte sich jedoch nur bedingt quantitativ verlassen werden, da beispielsweise die zu erreichende Genauigkeit stark vom zugrundeliegenden Modell und von der Leistungsfähigkeit der verwendeten Computer abhängt. Eine Untersuchung des Einflusses des Modells auf die maximal erreichbare Genauigkeit ist ein Thema dieser Arbeit. Außerdem sind heutige Computer (von wenigen Ausnahmen abgesehen) noch nicht in der Lage, in vertretbarer Zeit bzw. mit den bisher existierenden mathematischen Verfahren quantitative Analysen selbst moderat komplexer Modelle durchzuführen. So ist die praktische Relevanz von numerischer Strömungssimulation offensichtlich: Sie hilft in der Entwurfs- und Designphase eines Prototyps Zeit und Geld zu sparen, indem teure Experimente an echten Modellen erst im „Feintuning" durchgeführt werden. Außerdem bietet sie die einzige Möglichkeit, wenn Experimente als Ansatz ausscheiden. So hat beispielsweise eine (natürlich vom Pentagon finanzierte) amerikanische Forschungsgruppe die Explosion einer kleinen Atombombe in Chicago simuliert und damit großes populärwissenschaftliches Interesse geweckt.

Eine Einführung in den Themenkomplex des CFD bietet das Lehrbuch [63]. Empfehlenswert sind auch die Unterlagen zur Vorlesung *CFD* am Lehrstuhl für angewandte Mathematik und Numerik, zu finden unter
`http://www.mathematik.uni-dortmund.de/lsiii/download`.

1.1.2 Nische dieser Arbeit

In dieser Arbeit werden Strömungssimulationen untersucht, die grob in das Szenario des **virtuellen Windkanals** passen. Hierbei wird ein dreidimensionales, komplexes Objekt, gegeben durch eine Beschreibung seiner Oberfläche, in einem Kanal fixiert und von einer Seite des Kanals eine Strömung eingeleitet. Um das Szenario in eine für den Computer verarbeitbare Form zu bringen, wird das Differenzvolumen zwischen Objekt und Kanal (also der Bereich,

in dem beim Experiment tatsächlich Strömungsvorgänge auftreten würden) in viele kleine hexaederförmige Elemente zerlegt. All diese Elemente überdecken zusammen das Simulationsgebiet, ihre Gesamtheit wird **Rechengitter** (engl. *grid, mesh*) genannt. Diesen Vorgang nennt man **Diskretisierung**. Um hinreichend genaue quantitative Resultate mit der CFD-Simulation erhalten zu können, ist die räumliche Auflösung sehr fein zu wählen, d.h. es werden oft 10 Millionen und mehr Hexaeder-Zellen benötigt. Dies führt zu großen Problemdimensionen, die zugrundeliegenden Berechnungen können nur von Supercomputern bzw. hochparallelen Systemen in endlicher Zeit gelöst werden. Für qualitative Resultate reichen 500000 bis 5 Millionen Zellen (je nach Komplexität der Geometrie) aus. Es sei allerdings darauf hingewiesen, daß mit CFD-Verfahren durchgeführte Simulationen – selbst wenn sich qualitativ keine Unterschiede zeigen – große Differenzen im quantitativen Bereich, also in der erreichbaren Genauigkeit, aufweisen können. Ein Beispiel hierfür ist eine kleine lokale Störung der Oberfläche (etwa eine leichte Modifikation am Kühlergrill eines Automobils), die auf den ersten Blick den qualitativen Strömungsverlauf nicht beeinflußt, bei einer quantitativen Analyse aber zu einer meßbaren Änderung im Widerstandsbeiwert (einem Maß für den Luftwiderstand eines Fahrzeugs) führt. Die im Bereich des CFD entwickelten Techniken sind prinzipiell sehr gut für eben solche quantitativen Untersuchungen und Simulationen geeignet, allerdings stehen bei sehr komplexen Szenarien praktische Limitationen im Vordergrund: Die Berechnung eines vollständigen Simulationszyklus benötigt zu viel Zeit. Man könnte sich jetzt darauf verlassen, daß sich die Leistungsfähigkeit von Computern etwa alle 18 Monate verdoppelt (Moore's Gesetz). Dies ist aber unbefriedigend. Am Ende dieser Arbeit werden daher die Grenzen der bisher verwendeten Techniken aufgezeigt und diskutiert.

Ein Ansatz zur numerischen Durchführung von Strömungssimulationen ist die Methode der Finiten Elemente. Darin spielt u.a. der sogenannte Mehrgitteralgorithmus eine wesentliche Rolle. Die Kernidee ist es hierbei (Details werden in den nächsten Kapiteln ausführlich behandelt), zur Diskretisierung Gitter in verschiedenen Auflösungen zu verwenden. Ausgehend vom sogenannten Grobgitter, einem Gitter mit wenigen Zellen, dessen Existenz zunächst vorausgesetzt wird, wird ein feineres Gitter durch reguläre Unterteilung der einzelnen Gitterzellen erzeugt. Im Zweidimensionalen entstehen beispielsweise vier neue Elemente durch die Unterteilung eines Vierecks, im Dreidimensionalen wird ein Hexaeder in acht neue zerlegt. Befinden sich die dabei neu erzeugten Punkte im Inneren des Strömungsgebiets, können sie unverändert übernommen werden. Andernfalls ist nicht sichergestellt, daß durch Hinzunahme der neuen Punkte die Geometrie besser aufgelöst ist, d.h. **daß die neuen Punkte direkt geometrisch auf dem zu umströmenden Objekt zu liegen kommen**. Die Sicherstellung dieser Eigenschaft und die Untersuchung des Einflusses der dabei verwendeten Techniken auf die der Simulation zugrundeliegenden mathematischen Verfahren ist das Kernthema dieser Arbeit. Zur Beschreibung der Oberflächen wird dabei eine Triangulierung derselben verwendet. Diese sogenannten **Oberflächentriangulierungen** sind eine sehr vielseitige Darstellungsform, mit ihnen können Objekte beliebiger Gestalt und Topologie dargestellt werden. Sie sind zwar stückweise linear und somit prinzipiell nicht geeignet, analytisch glatte Oberflächen zu beschreiben, bei hinreichender Feinheit können sie jedoch eine geforderte Glattheit beliebig gut approximieren.

1.2 Übersicht

Hinweise Für einen kurzen Überblick über die wichtigsten Ergebnisse der Arbeit sollten – neben dieser Einleitung und dem essentiellen Ergebniskapitel – die Unterkapitel 2.3 und 3.4 gelesen werden. Sie stellen die untersuchten und teilweise selbst entwickelten Gitterglättungs- und Projektionstechniken vor, die in den numerischen Experimenten zum Einsatz kommen. Bei Bedarf kann dann basierend auf den dort jeweils angegeben Querverweisen in weitere Grundlagenkapitel verzweigt werden.

Strukturierung der Arbeit Kapitel 2 führt in die mathematischen Grundlagen der numerischen Strömungssimulation ein. Die Modellbildung wird anhand typischer Beispiele erklärt. Es folgen Abschnitte über die Diskretisierung mit der Methode der Finiten Elemente, sowie Details über den Weg vom diskretisierten Modell hin zu einem großen Gleichungssystem. Zum Schluß werden verschiedene Lösungsstrategien vorgestellt. Der für das Verständnis dieser Arbeit sehr wichtige Mehrgitteralgorithmus wird in einem eigenen Abschnitt detailliert behandelt. Der letzte Abschnitt stellt einfache Ansätze zur Gitterglättung vor, d.h. Verfahren, die die Qualität eines Rechengitters verbessern.

Das nächste Kapitel beinhaltet zunächst eine kurze Übersicht über Oberflächentriangulierungen als in dieser Arbeit verwendete Repräsentation von zu umströmenden Objekten. In zwei Abschnitten werden wichtige Formeln und Zusammenhänge hergeleitet, die für die folgenden Kernalgorithmen von Bedeutung sind. Der nächste Abschnitt beschäftigt sich mit Datenstrukturen und Suchverfahren, die zur effizienten Implementierung unverzichtbar sind. Hier wird insbesondere ein hocheffizientes parametrisches Traversierungsverfahren für Octrees vorgestellt sowie Nachteile bei der Verwendung von Suchheuristiken diskutiert. Es folgt ein Kernabschnitt dieser Arbeit, in dem drei Möglichkeiten vorgestellt werden, das Problem der Randanpassung im Verfeinerungsprozeß der numerischen Strömungssimulation mit streng geometrischen Verfahren zu lösen: die Projektion mit kürzesten Abständen, die gewichtete Projektion und die Umbrella-Projektion. Ein ausführliches Beispiel wird vorgestellt. Es schließt sich ein kurzer Überblick über verschiedene Techniken zur Reparatur von Objektrepräsentationen an, ein dieser Arbeit nah verwandtes Forschungsgebiet.

Kapitel 4 beschäftigt sich in der gebotenen Kürze mit der Generierung von Hexaedergittern, einem mit dieser Arbeit untrennbar verbundenen Forschungsgebiet. Zunächst werden verschiedene Ansprüche formuliert, die ein allgemeiner Gittergenerator einhalten sollte. Danach werden einige ausgewählte aktuelle Lösungsvorschläge zusammengefaßt und eine Klassifizierung existierender Ansätze vorgenommen. Es folgt die Beschreibung eines eigenen Entwurfs, der sich gewissermaßen als Nebenprodukt der im vorherigen Kapitel vorgestellten Kernalgorithmen für die Gittergenerierung ergab. Das Kapitel schließt mit der Definition allgemeiner Qualitätskriterien einer Hexaederdiskretisierung. Das gesamte Kapitel ist als Ausblick für interessierte Leser zu verstehen.

Im letzten Kapitel werden die numerisch-experimentellen Ergebnisse der Arbeit vorgestellt. Dazu werden zunächst zwei verschiedene Benchmarks zusammengefaßt, die beide Vergleiche zwischen einer reinen analytischen und einer mittels einer Oberflächentriangulierung appro-

ximierten Repräsentation einer Geometrie in einer Strömungssimulation erlauben. Einer der Benchmarks kann zudem als Standard bezeichnet werden und es existieren hochgenaue bestätigte Referenzlösungen. Zusätzlich wird eine qualitative Analyse eines Fahrzeugs in einem Windkanal durchgeführt, ein Szenario, was vorher mit der verwendeten Numeriksoftware nicht möglich war. Die Ergebnisse dieser Rechnungen lassen sich wie folgt zusammenfassen:

- Die Feinheit der Oberflächentriangulierung spielt für die erreichbare Genauigkeit eine große Rolle: Selbst wenn in einer hochauflösenden Visualisierung keine Unterschiede feststellbar sind, so kann doch eine zu grobe Oberflächentriangulierung die Genauigkeit empfindlich beeinflussen. Andererseits sind bei einer hinreichend feinen Triangulierung nur minimale Unterschiede zu einer analytischen Randbeschreibung feststellbar.

- Die Feinheit der Oberflächentriangulierung hat keinen Einfluß auf das Konvergenzverhalten des Mehrgitters. Die entwickelten Verfahren resultieren in robusten und effizienten Mehrgitter-Lösern.

- Unter der (starken und etwas „schwammigen") Voraussetzung, das Grobgitter sei bereits „hinreichend" genau an die Details der zu umströmenden Geometrie angepaßt, sind die in der Arbeit vorgestellten Projektionstechniken für die qualitative Analyse von Strömungsvorgängen um komplexe Objekte gut geeignet. Für die quantitative Auswertung komplexer Szenarien resultiert die Voraussetzung jedoch in Grobgittern mit mehreren 10000 Elementen, was die Verwendung von Mehrgittertechniken konterkariert.

Aus Interesse wird zusätzlich der Einfluß eines relativ neuen Gitterglättungs-Operators quantitativ untersucht, weil eine kurze Recherche keine Resultate zu diesem Thema lieferte. Das Kapitel endet mit einer ausführlichen kritischen Diskussion aller Resultate.

Die Arbeit schließt mit einer ausführlichen Literaturreferenz zum Thema.

Praktische Umsetzung Alle praktischen Implementierungen zur Arbeit wurden in einer modifizierten Version der Simulationssoftware FEATFLOW vorgenommen. In die nächste Version dieser Software (voraussichtlich Anfang 2005) werden die Implementierungen integriert werden und somit als *open source* unter `http://www.featflow.de` zur Verfügung stehen.

Kapitel 2

Mathematische Grundlagen der Strömungssimulation

2.1 Einführung

Dieses Kapitel beschreibt die mathematischen Grundlagen von numerischer Strömungssimulation. Es basiert auf Lehrbüchern und Vorlesungsskripten zum Thema, konkret im Bereich der Modellbildung auf [27,28], zur Theorie und Praxis in FEM und CFD auf [2,9,15,24,57,62,63] sowie allgemein zur numerischen Mathematik auf [20, 25, 49, 53, 58].

Notationen Es werden offene Gebiete $\Omega \subseteq \mathbb{R}^d$ betrachtet, für $d = 1, 2, 3$ mit dem Rand $\partial\Omega$. Der orthogonal zum äußeren Rand stehende Normalenvektor in einem Punkt $\mathbf{x} \in \partial\Omega$ ist $\mathbf{n}$, der Tangentialvektor $\mathbf{t}$ (s. Abb. 2.1).

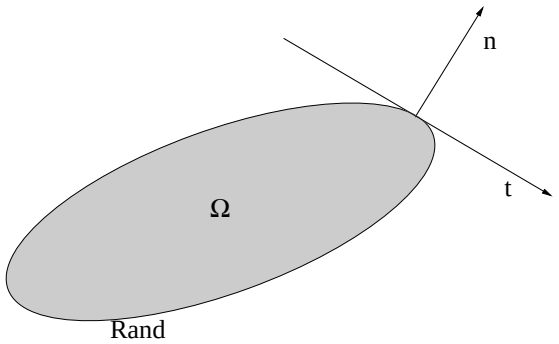


Abbildung 2.1: *Definition eines Gebiets*

Ortspunkte $\mathbf{x} \in \Omega$ werden mit $\mathbf{x} = (x_1, \ldots, x_d)^T$, Zeitpunkte mit $t \in I \subseteq \mathbb{R}$ bezeichnet. Für d-dimensionale Vektoren $\mathbf{a}, \mathbf{b}$ wird das übliche euklidische Skalarprodukt mit $\mathbf{a} \cdot \mathbf{b}$, und die

euklidische Norm mit $||\mathbf{a}||$ bezeichnet.

$$\begin{aligned} \mathbf{a} \cdot \mathbf{b} &:= \sum_{i=1}^{d} a_i b_i \\ ||\mathbf{a}|| &:= \sqrt{\mathbf{a} \cdot \mathbf{a}} \end{aligned}$$

Funktionen treten entweder skalarwertig ($u = u(\mathbf{x})$, $u = u(\mathbf{x}, t)$) oder vektorwertig

$$\mathbf{u} = \begin{pmatrix} u_1(\mathbf{x}, t) \\ \vdots \\ u_d(\mathbf{x}, t) \end{pmatrix}$$

auf. Partielle Ableitungen, d.h. Ableitungen nach einer der vorkommenden Variablen, werden wie folgt geschrieben:

$$\begin{aligned} \partial_t \mathbf{u} &:= \frac{\partial \mathbf{u}}{\partial t} \\ \mathbf{u}_{x_i} := \partial_i \mathbf{u} &:= \frac{\partial \mathbf{u}}{\partial x_i}. \end{aligned}$$

$\partial_i^p \mathbf{u}$ bezeichnet die p-te Ableitung.
Mit dem Nabla-Operator ∇ werden der Gradient (Vektor aller partiellen Ableitungen) einer skalaren sowie die Divergenz einer Vektorfunktion geschrieben als

$$\text{grad } u := \nabla u := (\partial_1 u, \ldots, \partial_d u)^T$$

$$\text{div } \mathbf{u} := \nabla \cdot \mathbf{u} := \partial_1 u_1 + \cdots + \partial_d u_d.$$

Zu einem Vektor $\mathbf{b} \in \mathbb{R}^d$ wird die Ableitung in Richtung $\mathbf{b}$ mit $\partial_b u := \mathbf{b} \cdot \nabla u$ bezeichnet. Entsprechend ist $\partial_n u := \mathbf{n} \cdot \nabla u$ die Ableitung in Richtung der äußeren Normalen auf dem Gebietsrand.
Die Kombination von Divergenz- und Gradientenoperator ergibt den *Laplace-Operator*:

DEFINITION 2.1 (LAPLACE-OPERATOR)
*Der **Laplace-Operator** Δ ist wie folgt definiert:*

$$\Delta u := \nabla \cdot (\nabla u) = \partial_1^2 u + \cdots + \partial_d^2 u. \tag{2.1}$$

2.1.1 Modellierung

Ziel der Modellierung ist es, für Naturvorgänge eine – oft stark vereinfachte – mathematische Beschreibung zu finden, d.h. Vorgänge wie Diffusion, Stofftransport oder Temperaturverteilung mit Hilfe mathematischer Gleichungen zu beschreiben. Dies ist ein höchst kreativer Akt, der viel Erfahrung erfordert und für den es kein Standardkochrezept gibt. Deshalb sollen an dieser Stelle nur vier typische Modelle exemplarisch dargestellt werden. Um alle Modelle später gleich behandeln zu können, werden sie in Form von Differentialgleichungen beschrieben.

Beispiele

1. Wellengleichung Schwingende Ausbreitungsvorgänge sind z.B. die Ausbreitung einer Wasserwelle (interpretiert als lokale Störung der Wasseroberfläche) oder eines Geräuschs (interpretiert als lokale Störung der Luftdichte). Schwingungen können mathematisch durch trigonometrische Funktionen dargestellt werden. Ein einfaches eindimensionales Beispiel ist die Funktion $u = u(x,t)$ im Ort x und der Zeit t mit $u(x,t) = \sin(x)\sin(t)$. Die Wellenbewegung wird als Sinusfunktion im Ort modelliert, zusätzlich beeinflußt durch eine Sinusfunktion in der Zeit. Diese genügt der Differentialgleichung

$$\partial_t^2 u = \partial_x^2 u, \tag{2.2}$$

wie man durch partielle Differentiation zeigen kann. Gleichungen dieser Form heißen auch *hyperbolisch*.

2. Wärmeleitungsgleichung Andere Ausbreitungsvorgänge sind dadurch gekennzeichnet, daß sich lokale Störungen nicht schwingend, sondern „diffundierend" (also sich abschwächend) fortpflanzen, z.B. die Temperaturausbreitung in einem Leiter (Wärmeleitung) oder die Verteilung einer Dichtekonzentration in einer Flüssigkeit (Stofftransport). Zur Beschreibung solcher Vorgänge dienen beispielsweise Funktionen der Form $u(x,t) = \sin(x)e^{-t}$, die Sinus-Funktion spiegelt die (im Ort ungedämpfte) Wellenbewegung wider, während die e-Funktion für eine Dämpfung in der Zeit sorgt. Diese Funktion genügt der Differentialgleichung

$$\partial_t u = \partial_x^2 u. \tag{2.3}$$

Der Beweis sowie weitere Beispiele finden sich in [27]. Dieser Gleichungstyp wird als *parabolisch* bezeichnet.

3. Auslenkung einer Membran Die Auslenkung einer Membran ist ein Anwendungsfall für das sogenannte *Poissonproblem*.
Die Membran sei am Rand eingespannt und werde durch eine vertikale Kraft $f(\mathbf{x})$ belastet. Gesucht ist die Auslenkung $u(\mathbf{x})$ unter der Einspannbedingung $u(\mathbf{x}) = 0$ auf $\partial\Omega$, mit $\mathbf{x} \in \Omega$. Zur Herleitung betrachtet man die Gesamtenergie $J(u)$ des Systems, die sich zusammensetzt aus Spannungsenergie $J_1(u)$ und potentieller Energie $J_2(u)$. Die Spannungsenergie ist proportional zur Oberflächenänderung und zusätzlich abhängig von einem Elastizitätsfaktor α. Eine ausführliche Herleitung ergibt:

$$\begin{aligned} J_1(u) &\approx \alpha/2 \int_\Omega |\nabla u|^2 \, d\Omega \\ J_2(u) &= -\int_\Omega f u \, d\Omega \end{aligned}$$

Eine Funktion $u = u(\mathbf{x})$ ist die gesuchte Auslenkung, falls sie das Prinzip der minimalen Energie erfüllt:

$$\begin{aligned} J(u) &\leq J(v) \qquad \text{für alle zulässigen Auslenkungen } v \text{ bzw.} \\ J(u) &\leq J(u + \varepsilon v) \text{ für alle } \varepsilon > 0 \text{ und alle zulässigen Auslenkungen } v \end{aligned}$$

Das Auslenkungsproblem ist also auf das folgende Minimierungsproblem zurückgeführt worden:

$$\frac{d}{d\epsilon} J(u + \varepsilon v)\Big|_{\varepsilon=0} = 0 \text{ für alle zulässigen Funktionen } v$$

Man erhält nach weiteren Umformungen die folgende äquivalente Integralschreibweise:

$$\alpha \int_\Omega \nabla u \nabla v d\Omega - \int_\Omega f d\Omega \, v = 0 \; \forall v.$$

Mit Hilfe der Greenschen Formel (s.u.) kann man nun das Integral aufteilen in ein Gebietsintegral und ein Randintegral. Da die Einspannbedingung Nullrandwerte vorschreibt, fällt das Randintegral weg. Es bleibt:

$$\int_\Omega (-\alpha \Delta u - f) v \, d\Omega = 0 \text{ für alle zulässigen } v$$

Da dies für alle Funktionen v gilt, muß der Ausdruck in der Klammer nach dem DuBois-Raymond-Lemma (s. [20]) bereits die Nullfunktion sein. Man erhält die Gleichung

$$-\alpha \Delta u = f \quad \text{in } \Omega \text{ mit Nullrandbedingungen.} \tag{2.4}$$

Dies ist das prototypische *Poissonproblem*, allgemein werden Gleichungen diesen Typs auch als *elliptisch* bezeichnet.

4. Strömungsprobleme Um die Grundgleichungen der Strömungsdynamik und der Strömungsmechanik zu beschreiben, sind zunächst zwei Betrachtungsweisen zu unterscheiden: Die **eulersche Betrachtungsweise** bezieht sich auf ein festes Bezugssystem, d.h. der Beobachter sieht beispielsweise Teilchen durch ein festes Fenster an sich vorbeiziehen. Die **lagrangesche Betrachtungsweise** bezieht sich auf ein bewegtes Bezugssystem, d.h. der Beobachter sitzt anschaulich direkt auf einer sich verbiegenden Metallplatte.

Abhängig von den Koordinatensystemen ergeben sich verschiedene Zeitableitungsbegriffe: Die materielle Zeitableitung du/dt bezeichnet die Änderungsrate eines sich bewegenden Fluidpartikels, während die lokale Zeitableitung $\partial u/\partial t$ die Änderungsrate in einem festen Punkt angibt. Mit Hilfe der Kettenregel ergibt sich der Zusammenhang

$$\frac{du}{dt} = \frac{\partial u}{\partial t} + \mathbf{v} \cdot \nabla u.$$

Die Grundgleichungen der (klassischen) Kontinuumsmechanik basieren auf dem physikalischen Grundprinzip der „Erhaltung": Zustandsgrößen wie z.B. Massedichte $\rho(\mathbf{x}, t)$, Energie bzw. Temperatur $T(\mathbf{x}, t)$, Impuls $\rho(\mathbf{x}, t)\mathbf{v}(\mathbf{x}, t)$ usw., werden als Dichtefunktionen beschrieben, deren Integrale über beliebige bewegte Kontrollvolumen sich beim Fehlen von äußeren Einflüssen nicht verändern. Änderungen finden nur bei Ein- und Ausströmvorgängen über den Rand statt.
Für die zeitliche Veränderung der Masse $m_V(t)$ eines solchen mit dem Geschwindigkeitsfeld $\mathbf{v}$ bewegten Volumens $V(t)$ mit der Oberfläche $S(t)$ gilt:

SATZ 2.2 (REYNOLDSCHES TRANSPORTTHEOREM)

$$0 = \frac{dm_V}{dt} = \frac{d}{dt}\int_{V(t)} \rho(\mathbf{x},t)\,dV = \int_{V(t)} \frac{\partial \rho(\mathbf{x},t)}{\partial t}\,dV + \int_{S(t)} \rho\mathbf{v}\cdot\mathbf{n}\,dS. \qquad (2.5)$$

Der Beweis dieser Aussage kann in [15] oder [63] nachgelesen werden.

Da dies für beliebige Volumen $V(t)$ gelten soll, ergibt sich für stetige Dichtefunktionen die folgende Erhaltungsgleichung 1. Ordnung (sog. „Kontinuitätsgleichung"):

$$\frac{\partial \rho}{\partial t} + \nabla\cdot(\rho\mathbf{v}) = 0.$$

Auf analogem Wege erhält man aus dem Erhaltungssatz für die Temperatur unter Berücksichtigung von Quelltermen q und einem Wärmediffusionsfaktor $\kappa \in \mathbb{R}$ in einem ruhenden Medium ($\mathbf{v} \equiv 0$) die folgende Erhaltungsgleichung 2. Ordnung (sog. „Wärmeleitungsgleichung"):

$$\frac{\partial T}{\partial t} + (\mathbf{v}\cdot\nabla)T = \kappa\nabla^2 T + q$$

Physikalische Anschauung erfordert, daß Lösungen zu diesen Gleichungen bei physikalisch sinnvollen Anfangs- und Randbedingungen stets positiv sind, d.h. $\rho > 0,\ T > 0$.

Die entsprechenden Erhaltungssätze für Dichte, Impuls und Energie führen unter geeigneten zusätzlichen Annahmen mit der Kontinuitätsgleichung auf die sogenannten *Navier-Stokes-Gleichungen* für inkompressible Flüssigkeiten, d.h. Flüssigkeiten, die auch unter Druck ihre Dichte nicht ändern:

DEFINITION 2.3 (NAVIER-STOKES-GLEICHUNGEN)

$$\partial_t\mathbf{v} - \nu\Delta\mathbf{v} + \mathbf{v}\cdot\nabla\mathbf{v} + \nabla p = \mathbf{f}, \quad \nabla\cdot\mathbf{v} = 0. \qquad (2.6)$$

Eine ausführliche Herleitung bieten [15, 63].

2.1.2 Diskretisierung

Nachdem ein mathematisches Modell in Form von Differentialgleichungen vorliegt, muß aus ihm eine Lösung gewonnen werden, die eine Vorhersage für den zugrundeliegenden Naturvorgang erlaubt. Bei einfachen Gleichungen ist es vielleicht mit viel Erfahrung, Geduld und Intuition möglich, direkt durch scharfes „Daraufschauen" eine analytische Lösung zu bestimmen. Bei komplexeren Systemen wie den Navier-Stokes-Gleichungen ist dies nicht mehr möglich, daher ist man in diesem Fall auf ein numerisches Lösungsverfahren angewiesen.
Ein Beispiel ist das Differenzenverfahren, bei dem die Differentialoperatoren durch Differenzenquotienten ersetzt werden. Eine Approximation niedriger Ordnung der ersten Ableitung ist beispielsweise gegeben durch:

$$\partial_x f(x) \approx \frac{f(x+h) - f(x)}{h} \text{ für kleines } h.$$

Ähnliche Formeln existieren auch mit besseren Ordnungen bzw. für höhere Ableitungen. Untersuchungen und Beweise der jeweiligen Approximationsordnungen für verschiedene Differenzenverfahren finden sich in [53].

Das Verfahren, um das es im folgenden etwas konkreter gehen wird, heißt **Finite-Elemente-Verfahren** oder **Methode der Finiten Elemente**, kurz FEM.
Anders als beim Differenzenverfahren, bei dem die punktweise Gültigkeit der Gleichung gefordert wird, wird eine unter gewissen Zusatzforderungen äquivalente Variationsgleichung oder schwache Formulierung zugrunde gelegt, die eine Übereinstimmung nur in einem gewissen Mittel fordert. Eine aktuelle Übersicht, insbesondere in praktischer Hinsicht, findet sich in [15] sowie in [9]. Dort kann auch die ausführliche Herleitung nachgeschlagen werden, die hier lediglich beispielhaft zusammengefaßt werden soll.
Für das Beispiel der Poisson-Gleichung

$$-\Delta u = f \text{ mit } u|_{\partial\Omega} = 0$$

läßt sich über die Minimierung eines Energiefunktionals als variationelle Formulierung herleiten:

$$\int_\Omega -\Delta u v \, d\Omega = \int_\Omega f v \, d\Omega \text{ für ein } v \in V$$

V ist hierbei ein Funktionenraum, der wenigstens so gewählt sein muß, daß obige Integration definiert ist. V heißt auch *Ansatzraum*. Diese Formulierung ist äquivalent, wenn f als stetig vorausgesetzt wird und wenn eine Lösung tatsächlich existiert. Durch partielle Integration erhält man die sogenannte Greensche Formel (s. [20])

$$\int \Delta u v = \int_{\partial\Omega} \partial_n u v \, dS - \int_\Omega \nabla u \nabla v \, d\Omega.$$

Der Δ-Operator wird quasi auf beide Funktionen „verteilt", wobei zusätzlich das Randintegral entsteht. Geht man von Nullrandbedingungen aus, so verschwindet dieses aber und es bleibt:

$$\int_\Omega \Delta u v d\Omega = -\int_\Omega \nabla u \nabla v d\Omega.$$

Daraus ergibt sich folgende Formulierung:

$$\int_\Omega \nabla u \nabla v \, d\Omega = \int_\Omega f v \, d\Omega,$$

Um das Problem lösen zu können, betrachtet man eine näherungsweise Lösung. Anstelle des zugrundeliegenden (im Fall der Anwendung auf Differentialgleichungen stets unendlichdimensionalen) Funktionenraums V, aus dem die Funktionen v stammen, wird ein Teilraum $V_h \subset V$ mit $N := \dim V_h < \infty$ gewählt und die Lösung dort gesucht. Beispielsweise kann man nur alle Polynome bis zu einem bestimmten Grad betrachten. Die schwache Formulierung lautet dann

$$\int_\Omega \nabla u_h \nabla v_h \, d\Omega = \int_\Omega f_h v_h \, d\Omega. \tag{2.7}$$

Da die Dimension von V_h endlich ist, gibt es eine endliche Basis $\{\phi_i \in V_h, i = 1, \ldots, N\}$, die den Teilraum V_h aufspannen, d.h.

$$V_h = \{v_h : v_h(x) = \sum_{i=1}^{N} \xi_i \phi_i(x), \quad \xi_i \in \mathbb{R}\}.$$

Mit der Linearität der verwendeten Operatoren und f ist Gleichung (2.7) äquivalent zu:

$$\int_\Omega \nabla u_h \nabla \phi_i \, d\Omega = \int_\Omega f_h \phi_i \, d\Omega, \quad i = 1, \ldots, N.$$

Da die Lösung u_h in V_h liegen soll, besitzt sie ebenfalls eine Darstellung der Form

$$u_h(x) = \sum_{j=1}^{N} \xi_j \phi_j(x).$$

Einsetzen und Herausziehen der Summe ergibt:

$$\sum_{j=1}^{N} \xi_j \int_\Omega \nabla \phi_j \nabla \phi_i \, d\Omega = \int_\Omega f_h \phi_i \, d\Omega, \quad i = 1, \ldots, N. \tag{2.8}$$

Die Funktionen $\{\phi_i\}$ heißen *Ansatz- bzw. Testfunktionen*. Man spricht von *Galerkin-Verfahren*. Setzt man als Testfunktionen v die N Basisfunktionen ϕ_j ein, entsteht ein lineares Gleichungssystem der Form $A\mathbf{x} = \mathbf{b}$ mit den Unbekannten $\mathbf{x} = (\xi_1, ..., \xi_N)$, den Matrixeinträgen $a_{ij} = \int_\Omega \nabla \phi_j \nabla \phi_i \, d\Omega$ und der rechten Seite $\mathbf{b} = (\int_\Omega f_h \phi_i \, d\Omega)_{i=1...N}$. Folgende Fragen bleiben zu klären:

- Wahl der Funktionen ϕ_i und Zerlegung des Gebiets Ω,
- Berechnung der Integrale und
- Lösung des Gleichungssystems.

2.1.3 Wahl der Basisfunktionen und Zerlegung des Gebiets

Die Wahl der Basisfunktionen ist für die Realisierbarkeit und Effizienz des Verfahrens von großer Bedeutung. Im Unterschied zu klassischen Ansätzen mittels global einheitlich definierter Funktionen werden bei der Methode der Finiten Elemente stückweise definierte Funktionen – in der Regel Polynome – zugrunde gelegt. Die dabei erzeugten diskreten Probleme besitzen eine spezielle Struktur, und es existieren angepaßte Verfahren zu ihrer numerischen Behandlung.

Die Methode der Finiten Elemente besitzt die folgenden drei typischen Merkmale:

1. Das Grundgebiet wird in geometrisch einfache Teilgebiete, z.B. Dreiecke und Rechtecke bei Problemen in der Ebene oder Tetraeder und Hexader bei Problemen im Raum, zerlegt. Ein solches Teilgebiet heißt **Zelle** oder **Element**, ihre Gesamtheit wird als **Gitter** (engl. *grid, mesh*) bezeichnet.
2. Über diesen Teilgebieten werden lokale Ansatz- und Testfunktionen definiert.

3. Die Ansatzfunktionen werden so angepaßt, daß sie Übergangsbedingungen zwischen Elementen zur Sicherung gewünschter globaler Eigenschaften wie z.B. der Stetigkeit einhalten. Beispielsweise kann ein stetiger Übergang zwischen den Knoten oder den Kantenmittelpunkten benachbarter Elemente gefordert werden.

Zerlegung des Gebiets Ω Die unterschiedlichen Typen von Gittern lassen sich in folgende Klassen einteilen:

1. Uniform strukturierte (kartesische) Gitter (engl. *cartesian grids*) bestehen aus achsenparallelen Zellen gleichen Typs und gleicher Größe.

2. Rechteckig strukturierte Gitter (engl. *rectilinear grids*) bestehen aus Rechteckzellen, die aber in Breite und Höhe variieren können.

3. Strukturierte Gitter (engl. *stuctured, curvilinear grids*) bestehen aus Zellen gleichen Typs und gleicher Konnektivität, die aber in Größe und Ausrichtung variabel sind.

4. Allgemein strukturierte Gitter (engl. *general structured grids*) sind Gitter, die auf irgendeine Art und Weise strukturiert sind, beispielsweise durch Polygone oder Kreise etc.

5. Unstrukturierte Gitter (engl. *unstructured grids*) verwenden verschiedene Zelltypen in einem Gitter oder haben variable Konnektivität.

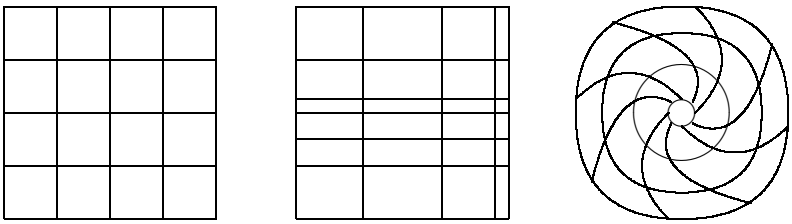


Abbildung 2.2: Beispiele für Gittertypen nach [43]: kartesisch, rechteckig bzw. strukturiert, allgemein

Definition eines Finiten Elements Die konkreten Finiten Elemente werden charakterisiert durch die Geometrie des Teilgebiets (s.o.) sowie die Anzahl, die Lage und die Art der Vorgaben für die Ansatzfunktionen ϕ_i.
Die unabhängig vorgebbaren Informationen werden in Anlehnung an die Mechanik „Freiheitsgrade" genannt. Der Einfluß der einzelnen Freiheitsgrade wird durch entsprechende Formfunktionen erfaßt.
Als Beispiel aus dem „Zoo" von Finiten Elementen soll an dieser Stelle ein einfaches bilineares Viereckselement erläutert werden (s. Abbildung 2.3).

Abbildung 2.3: *Rechteckiges Finites Element*

Die Freiheitsgrade für die Funktionswerte liegen in den Ecken, d.h. man kann auf einem Referenzelement bilineare Ansatzfunktionen verwenden, die in einer Ecke den Wert 1 annehmen und in den anderen 0. Auf dem Einheitsquadrat (dem Standard-Referenzelement), d.h. $\xi, \eta \in [0, 1]$, können die Ansatzfunktionen folgendermaßen aussehen:

$$\begin{aligned} \phi_1 &= (1-\xi)(1-\eta), \\ \phi_2 &= \xi(1-\eta), \\ \phi_3 &= \xi\eta, \\ \phi_4 &= (1-\xi)\eta. \end{aligned}$$

Die i-te Funktion ist dabei im i-ten Eckpunkt gleich eins, in den anderen Eckpunkten gleich null und dazwischen hat sie einen linearen Verlauf. Fordert man stetige Übergänge zwischen benachbarten Elementen in den Eckpunkten (und damit wegen der Linearität auf einer ganzen Kante), so nennt man die Menge dieser Elemente Q_1, wird nur Stetigkeit an den Kantenmittelpunkten gefordert, spricht man von $\tilde{Q}_1$. Die Koeffizienten der Gleichung „ziehen" sozusagen an den Ansatzfunktionen und bilden so durch unterschiedliche Werte die Lösung ab. Weitere Beispiele für die Definition von Ansatzfunktionen und ihre analytische Untersuchung finden sich in jedem Lehrbuch über Finite Elemente, also insbesondere in [9, 15].

2.1.4 Diskretisierung in der Zeit

Bisher sind die vorkommenden Gleichungen nur im Ort diskretisiert worden. Viele Differentialgleichungen wie beispielsweise die Navier-Stokes-Gleichungen sind jedoch auch zeitabhängig.

Typischerweise wird zunächst in der Zeit diskretisiert, und zwar beispielsweise mit dem *Crank-Nicolson*-Verfahren der Ordnung 2. Dieses und weitere Verfahren finden sich in [20, 53]. Die halbdiskrete Gleichung wird dann mit der Methode der Finiten Elemente im Ort diskretisiert.

Alternativ kann natürlich auch erst in der im Ort und dann in der Zeit diskretisiert werden. Dies führt in der halbdiskreten Version zu gewöhnlichen Differentialgleichungen, in die die kontinuierliche Zeit als Anfangswert eingeht. Details finden sich wieder in fast jedem der angegebenen Lehrbücher.

2.1.5 Berechnung der Integrale

Eine wichtige Teilaufgabe ist die Berechnung der in der schwachen Formulierung auftretenden Integrale $\int_\Omega \phi_i \phi_j \, d\Omega$. Da diese Integrale allgemein nicht analytisch gelöst werden können, ist

man hier auch wieder auf numerische Verfahren, genannt Quadraturverfahren, angewiesen. Ein allgemeines Quadraturverfahren hat die folgende Gestalt:

$$\int_\Omega f(x)\, d\Omega \approx \sum_{i=1}^{n} w_i f(\mathbf{x}_i) \text{ mit } \mathbf{x}_i \in \Omega$$

Die $\mathbf{x}_i$ heißen Stützstellen, die w_i Gewichte der Quadraturformel. Das Verhalten und die Qualität der Quadraturformel wird von der Wahl dieser Werte maßgeblich beeinflußt. Zwei Verfahren sollen im folgenden vorgestellt werden, Details und Beweise finden sich in jedem guten Lehrbuch über numerische Mathematik (also beispielsweise in [53]).
Das erste Verfahren ist das Trapezverfahren. Das Intervall, über das zu integrieren ist, wird äquidistant aufgeteilt, und die so entstandenen Trapezflächen werden aufsummiert. Abbildung 2.4 verdeutlicht das Verfahren.

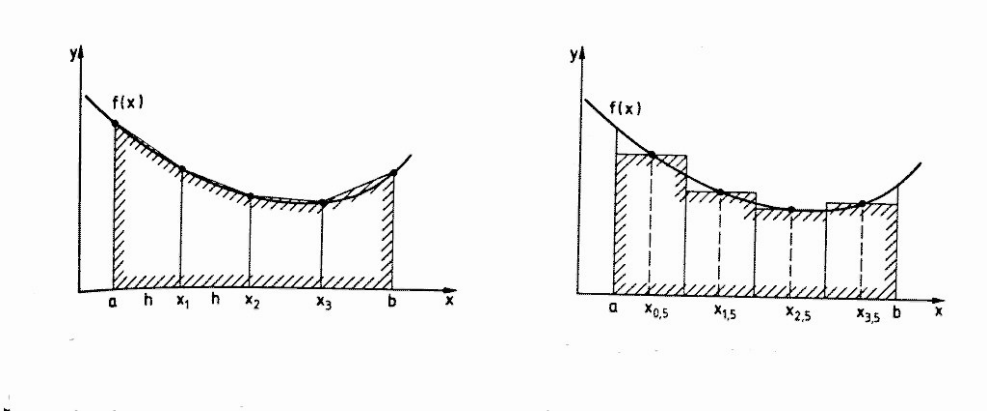


Abbildung 2.4: *Trapez- und Mittelpunktverfahren nach [53]*

Eine Variation benutzt nicht die Funktionswerte an den Teilintervallgrenzen, sondern in den Mittelpunkten. Dieses Verfahren heißt Mittelpunktverfahren.

Das nächste betrachtete Verfahren gehört zu der Klasse der sogenannten Interpolationsquadraturformeln. Die grundlegende Idee hinter diesen Verfahren ist es, die zu integrierende Funktion durch ein Interpolationspolynom der Ordnung n zu ersetzen und dieses Polynom dann exakt (d.h. analytisch) zu integrieren. Ein Vertreter dieser Klasse ist das Gauß-Verfahren. Es verwendet als Stückstellen die Nullstellen der Lagrange-Polynome $L_i^{(n)}$ zu den (äquidistant gewählten) Interpolationsstellen $\mathbf{x}_j$. Details finden sich in jedem Lehrbuch über Interpolation oder numerische Mathematik (a.a.O.). Die Gauß-Integration ist unter allen Quadraturformeln dadurch ausgezeichnet, daß sie bei n gegebenen Stützstellen noch Polynome vom Grad $2n-1$ exakt integrieren kann.

2.1.6 Lösung des Gleichungssystems

Es gibt zwei große Klassen an Verfahren zur Lösung von linearen Gleichungssystemen, die direkten und die iterativen Verfahren. Die direkten Verfahren liefern die Lösung nach einem

Schritt, sind dafür aber sehr rechenaufwendig. Die iterativen Verfahren sind pro Schritt weniger rechenintensiv, benötigen aber mehrere, unter Umständen sehr viele Schritte, um die Lösung zu liefern. Dabei tritt jedoch das Problem auf, daß nicht alle Verfahren in allen Situationen auch konvergieren, d.h. im Grenzwert die Lösung bestimmen. Bieten diese Verfahren aber eine Möglichkeit zur Fehlerschätzung, so kann in jedem Schritt beurteilt werden, wie gut die Approximation bereits ist, was in praktischen Szenarien vollkommen ausreicht. Die Darstellung in diesem Kapitel basiert größtenteils auf den Büchern [45] und [23], beide stellen neben den Grundlagen und Konvergenzbeweisen auch ausführliche spezialisierte Verfahren für den (in der FEM-Praxis relevanten) Fall dünn besetzter Matrizen einer festen Struktur vor. Darüber hinaus enthält [45] auch die Ergebnisse experimenteller Analysen des Konvergenzverhaltens iterativer Verfahren für typische Modellprobleme. Für Details wird auf diese Quelle verwiesen.

Direkte Verfahren

Das **Gauß-Verfahren** ist Stoff der Schulmathematik. Es benötigt kubische Laufzeit, was es für praktische Anwendung disqualifiziert.
Eine Variante des Gauss-Verfahrens ist der **Thomas-Algorithmus** zur Lösung tridiagonaler Gleichungssysteme in linearer Zeit ([53] stellt ihn vor, ohne allerdings diesen Namen zu verwenden).
In der Praxis ist die Programmbibliothek `UMFPACK` hochperformant, zu finden unter

`http://www.cise.ufl.edu/research/sparse/umfpack/`.

Sie bietet hochoptimierte Implementierungen von verschiedenen Lösungsverfahren. Hauptgrund für die Effizienzsteigerung ist die Verwendung intelligenter Umordnungsstrategien, so daß die Rechnung mit möglichst wenig Transfers zwischen Cache und Speicher auskommt. Eine Beschreibung aller Algorithmen bietet [14]. Experimente zeigen, daß `UMFPACK` für kleine Problemgrößen (ca. 10000 Unbekannte) auf handelsüblichen PCs fast unschlagbar schnell ist.

Iterative Verfahren

Die zweite Klasse bilden die iterativen Verfahren, die das endgültige Resultat nicht schon nach einem Schritt ausgeben, sondern erst nach mehreren. Dafür sind die Einzelschritte deutlich billiger. Das Kriterium zur Bewertung der Geschwindigkeit eines Iterationsverfahrens heißt Konvergenzrate ϱ und ist folgendermaßen definiert (dabei bezeichne $\mathbf{x}^{(k)}$ das Ergebnis nach k Iteration und $\mathbf{x}^{(0)}$ den Startwert):

DEFINITION 2.4 (KONVERGENZRATE)
*Die **numerische Konvergenzrate** ϱ ist definiert als*

$$\varrho := \left(\frac{||A\mathbf{x}^{(k)} - \mathbf{b}||}{||A\mathbf{x}^{(0)} - \mathbf{b}||} \right)^{1/k} \quad \in \quad]0;1[,$$

*also als Verhältnis der Residuen im k-ten und ersten Schritt der Iteration. Die **analytische Konvergenzrate** wird in einigen Beweisen benötigt und ist der limes superior dieser Größe für $k \to \infty$.*

Die Konvergenzrate gibt an, wie viel Genauigkeit man pro Iterationsschritt gewinnt. Konvergenzraten größer 0.5 sind „schlecht”, kleiner als 0.1 sind „sehr gut”.

CG-Verfahren Das CG-Verfahren setzt symmetrische und positiv definite Koeffizientenmatrizen A voraus, d.h.

$$\begin{aligned} A &= A^T \\ \mathbf{x} \cdot A\mathbf{x} &> 0 \quad \forall \mathbf{x} \in \mathbb{R}^N \setminus \{\mathbf{0}\} \\ \mathbf{x} \cdot A\mathbf{x} &= 0 \quad \Leftrightarrow \mathbf{x} = \mathbf{0}. \end{aligned}$$

Der zur Auflösung des Gleichungssystems erforderliche Aufwand läßt sich durch die Verwendung einer der Aufgabe angepaßten Basis $\{\mathbf{p}_j\}$ des $\mathbb{R}^N$ gezielt reduzieren. Basisvektoren mit der Eigenschaft

$$A\mathbf{p}_i \cdot \mathbf{p}_j = \delta_{ij},$$

mit dem Kronecker-Symbol δ_{ij} heißen konjugiert oder A-orthogonal. Stellt man die gesuchte Lösung $\mathbf{x}$ des Gleichungssystem über der Basis $\{\mathbf{p}_j\}$ dar, d.h.

$$\mathbf{x} = \sum_{j=1}^{N} \eta_j \mathbf{p}_j,$$

dann lassen sich die zugehörigen Koeffizienten $\eta_j, j = 1, \ldots, N$ wegen der A-Orthogonalität von $\{\mathbf{p}_j\}$ explizit darstellen durch

$$\eta_j = \frac{\mathbf{b} \cdot \mathbf{p}_j}{A\mathbf{p}_j \cdot \mathbf{p}_j}, \quad j = 1, \ldots, N.$$

Konjugierte Richtungen sind in der Regel nicht a-priori bekannt. Der Grundgedanke der CG-Verfahren besteht darin, diese Richtungen mit Hilfe eines Orthogonalisierungsverfahrens aus den verbleibenden Defekten $\mathbf{d}^{(k+1)} := \mathbf{b} - A\mathbf{x}^{(k+1)}$ in den Iterationspunkten $\mathbf{x}^{(k+1)}$ rekursiv zu erzeugen. Man stellt also die neue Richtung $\mathbf{p}^{(k+1)}$ in der Form

$$\mathbf{p}^{(k+1)} = \mathbf{d}^{(k+1)} + \sum_{j=1}^{k} \beta_{kj} \mathbf{p}^{(j)}$$

dar, und bestimmt die Koeffizienten $\beta_{kj} \in \mathbb{R}$ aus der verallgemeinerten Orthogonalitätsbedingung $A\mathbf{p}^{(k+1)} \cdot \mathbf{p}_j = 0, \quad j = 1, \ldots, k$.
Die Richtungen $\mathbf{d}^{(k+1)}$, die auch Anti-Gradienten der dem linearen Gleichungssystem zugeordneten Funktion

$$F(\mathbf{u}) = \frac{1}{2}(A\mathbf{u}, \mathbf{u}) - (\mathbf{b}, \mathbf{u})$$

in den Punkten $\mathbf{u}^{(k+1)}$ sind, gaben dem CG-Verfahren (engl. *conjugate gradients*) seinen Namen. In [23] und in [45] wird gezeigt, daß das Verfahren theoretisch nach N Iterationen die Lösung erreicht, es konvergiert also immer in linear vielen Schritten. Leider kann dieser Vorteil aus Gründen der numerischen Instabilität nicht immer in der Praxis auch genutzt werden.

BiCGStab-Verfahren Um den numerischen Nachteil des CG-Verfahrens auszugleichen, wurden zwei Modifikationen vorgestellt: **BiCG** und **CGS**. Jedes dieser Verfahren ist noch nicht zufriedenstellend, es können trotzdem noch Oszillationen auftreten. Kombiniert man jedoch beide Verfahren zum sogenannten **BiCGStab**-Algorithmus (s. [61]), so ergibt sich eine stabile und effiziente Iteration. Details finden sich wieder in [45]. In beiden angegebenen Quellen wird gezeigt, daß der Algorithmus parameterfrei ist (d.h. es entfällt die oft manuelle Suche nach der günstigsten Setzung eines Parameters). Außerdem arbeitet **BiCGStab** auch auf nichtsymmetrischen Matrizen. Praktische Experimente deuten auf hohe Konvergenzraten hin, aus theoretischer Sicht ist es bisher allerdings nicht gelungen, die Konvergenz auch nachzuweisen. Aus der Sicht des Mehrgitter-Algorithmus (s. Kap. 2.2) ist noch bemerkenswert, daß sich diese Iteration sehr gut als Glätter verwenden läßt, weil die Kombination mit einer Vorkonditionierungsmatrix möglich ist.

Defektkorrekturverfahren Diese Klasse von Verfahren arbeitet so, daß zum Iterationsvektor ein Korrekturwert hinzuaddiert wird, um so näher an die Lösung zu gelangen. Das zu lösende lineare Gleichungssystem wird folgendermaßen umgeschrieben:

$$\begin{aligned} \mathbf{x} &= A^{-1}\mathbf{b} \\ &= \mathbf{x} - \mathbf{x} + A^{-1}\mathbf{b} \\ &= \mathbf{x} - A^{-1}A\mathbf{x} + A^{-1}\mathbf{b} \\ &= \mathbf{x} + A^{-1}(\mathbf{b} - A\mathbf{x}). \end{aligned}$$

In dieser Form ist dies noch ein direktes Verfahren, die Lösung steht nach einem Schritt zur Verfügung. Zur Vorbereitung der weiteren Schritte wird obige Gleichung formuliert als Iterationsverfahren:

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

Der Ausdruck in der Klammer ist gerade der Defekt im k-ten Schritt.
Für die sogenannte Vorkonditionierungsmatrix werden nun, um Defektkorrekturverfahren herzuleiten, nicht die kompletten Einträge von A verwendet, sondern nur Teile davon, die einfacher zu invertieren sind (Matrix C in der folgenden Gleichung). Nimmt man z.B. nur die Diagonaleinträge von A, sind zur Invertierung nur deren Kehrwerte zu bilden, und man spricht vom Jacobi-Verfahren. Nimmt man die untere Dreiecksmatrix hinzu, die sich durch Rückwärtseinsetzen relativ leicht invertieren läßt, erhält man das sogenannte Gauß-Seidel-Verfahren. Allerdings sind die Konvergenzraten bei diesen Verfahren so schlecht, daß sie als eigenständige Löser in der Praxis nicht verwendet werden. In [45] wird beispielsweise gezeigt, daß die Abschätzung der Konvergenzrate nur mit der Minimalschranke „< 1" gelingt, d.h. die Verfahren konvergieren, jedoch sehr langsam. Sie spielen aber eine wichtige Rolle im noch folgenden Mehrgitterverfahren. Auch hier ist [23] eine gute Empfehlung, um weitere Details zu erfahren. Ferner ist ein sogenannter Relaxationsparameter $\omega < 1$ gebräuchlich, um den Defektkorrekturwert zu dämpfen und somit bessere Konvergenzraten zu erzielen. Die vollständige Vorschrift schreibt sich schließlich:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \omega C^{-1}(\mathbf{b} - A\mathbf{x}^{(k)}). \tag{2.9}$$

Es ist auch möglich, ω in Abhängigkeit von der Iterationszahl k zu wählen.

Alle bisher betrachteten Verfahren sind in praktischen Anwendungen oft (zu) langsam und haben den Nachteil, daß ihre Konvergenzrate von der Feinheit des Gitters abhängt. Das im nächsten Abschnitt vorgestellte Mehrgitterverfahren stellt eine wichtige Alternative dar.

2.2 Der Mehrgitter-Algorithmus

Der Mehrgitteralgorithmus wurde in den Jahren 1981 bis 1985 u.a. von Hackbusch entwickelt. Große Resonanz rief seine Monographie [24] aus dem Jahr 1985 (aktualisierte Ausgabe 2003) hervor. In diesem Buch und beispielsweise in [11] wird zur Motivation des Mehrgitterverfahrens zunächst analytisch untersucht, wie gut obige Defektkorrekturverfahren Fehler in der Lösung dämpfen. Es wird gezeigt, daß die hochfrequenten Anteile des Fehlers gut gedämpft werden, während bei den niederfrequenten Anteilen Schwierigkeiten auftreten. Dies verursacht die langsame Konvergenz. Die Grundidee des Mehrgitterverfahrens besteht nun darin, eine Sequenz oder Hierarchie von Gittern unterschiedlicher Feinheit zu verwenden. Die einzelnen Hierarchiestufen werden häufig mit dem englischen Begriff *level* bezeichnet. Ein Fehler, der auf einem Gitter noch niederfrequent ist, ist auf einem anderen Gitter hochfrequent und kann deshalb dort gut geglättet werden. Der Defekt wird also zwischen Defektkorrekturverfahren, die auf Gittern verschiedener Feinheit arbeiten, hin- und hertransferiert. Somit kann der gesamte Mehrgitteralgorithmus wieder als Defektkorrekturverfahren aufgefaßt werden.

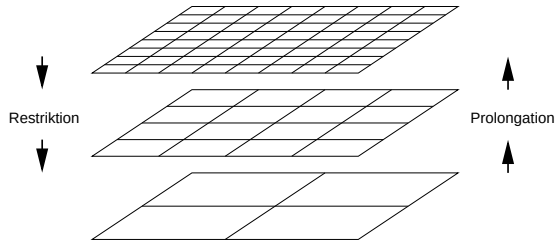


Abbildung 2.5: *Mehrgitteroperationen auf drei verschieden feinen Gittern*

In den folgenden Abschnitten werden zunächst vorbereitend einzelne Komponenten des Mehrgitteralgorithmus analysiert, bevor das Verfahren am Ende des Kapitels im Pseudocode vorgestellt wird und einige theoretische und praktische Resultate insbesondere zum Konvergenzverhalten zusammengestellt werden.

2.2.1 Erstellen der Gitterhierarchie

Dieser Abschnitt beschreibt Möglichkeiten, wie eine für den Mehrgitteralgorithmus nötige Gitterhierarchie erstellt werden kann. Adaptive Varianten des Algorithmus, die eine feinere Gitterebene nur dann erzeugen, wenn die erzeugte Näherung nicht genau genug ist, sollen nicht betrachtet werden. Desweiteren werden nur solche Verfahren betrachtet, die alle Zellen regulär unterteilen, d.h. die keine „hängenden" Knoten erzeugen. Für eine exakte Definition der Regularität sowie für eine ausführliche Beschreibung alternativer Unterteilungstechniken wird

auf [11] verwiesen.

Die Kernunterscheidung aller Ansätze ist die Richtung, nach der sie vorgehen: *top down*-Zugänge beginnen bei einem **Grobgitter** (engl. *coarse grid*), und verfeinern dieses sukzessive, *bottom up*-Zugänge arbeiten genau anders herum. Für beide Ansätze gibt es starke Argumente, Teilaspekte, die in die eine Richtung trivial zu lösen sind, führen beim alternativen Zugang zu komplexen Algorithmen. Die Kernprobleme aus der Sicht eines Informatikers sind:

- Bei Verfeinerungsstrategien muß für neu erzeugte Gitterpunkte und Elemente (s.u.) sichergestellt werden, daß sie die Geometrie der Simulation besser auflösen. Konkret müssen diese neuen Knoten klassifiziert werden in innere Knoten und Randknoten. Diese Entscheidung ist nichttrivial. Desweiteren darf die Qualität (s. Kap. 4.4) der neuen Elemente nicht schlechter sein als die der Elemente auf dem jeweils gröberen Gitter, um die numerische Stabilität des Mehrgitteralgorithmus nicht zu gefährden.

- Bei Vergröberungsstrategien müssen mehrere Zellen zu einer neuen zusammengefaßt werden, und zwar so, daß die Knoten des neuen „großen" Elements mit Knoten der Ausgangselemente übereinstimmen. Dies stellt sehr starke Ansprüche an die Anzahl der Elemente eines Gitters. Andernfalls muß bei der Restriktion der Lösung auf das gröbere Gitter interpoliert werden, was starke Auswirkungen auf das Konvergenzverhalten und die Effizienz des gesamten Mehrgitterverfahrens haben kann. Dies ist auch der Hauptgrund, warum in der Praxis *top down*-Ansätze häufiger anzutreffen sind.

Da die in dieser Arbeit verwendete Numeriksoftware FEATFLOW (v1.2) (s. [59, 60] sowie `http://www.featflow.de`) streng *top down* strukturiert ist, wird hier nur der Unterteilungsprozeß, nicht aber der Vergröberungsprozeß beschrieben.

Geometrische Unterteilung

Die Unterteilung ist zunächst ein rein geometrischer Prozeß. Aus einem Viereckselement werden vier neue Zellen erzeugt, indem neue Knoten an den Mittelpunkten der Kanten und im geometrischen Mittelpunkt des Elements erzeugt und diese dann durch Kanten verbunden werden. Analog werden bei einer Hexaederzelle alle sechs Seitenflächen unterteilt und zusätzlich der Mittelpunkt des Elements hinzugenommen. So entstehen acht neue Hexaeder.

Iteriert man diesen Prozeß über alle m Zellen eines Gitters, entsteht ein feineres Hexaedergitter mit $8m$ Zellen. Wichtig ist, daß die Punkte der (groben) Gitterebene k in allen so entstehenden Hierarchiestufen $k + j, j \geq 1$ erhalten bleiben. In einer praktischen Implementierung wird man natürlich nicht alle Kanten und Seitenflächen, die zu mehreren Zellen gehören, für jedes Element separat unterteilen.

Eine wichtige Modifikation dieser Unterteilungstechnik ist die anisotrope Verfeinerung, bei der eine Unterteilungsrichtung stärkeres Gewicht bekommt. [36] stellt dazu das ScaRC-Konzept vor. So kann die Unterteilung besser an zu erwartende Strömungsvorgänge angepaßt werden. Die Parallelisierung eines Mehrgittercodes spielt in der Praxis auch eine wesentliche Rolle. Adaptive Unterteilungstechniken finden sich in 3D bisher kaum.

Randpunkte

Für alle so neu entstehenden inneren Knoten ist der Unterteilungsvorgang (mit Ausnahme einer eventuell durchzuführenden Gitterglättung, s. Kapitel 2.3) damit abgeschlossen. Neu entstandende Punkte, die als Randpunkte (s.u.) erkannt werden, müssen jedoch nicht notwendig bereits geometrisch auf dem Rand liegen. Ein Beispiel zeigt Abbildung 2.6.

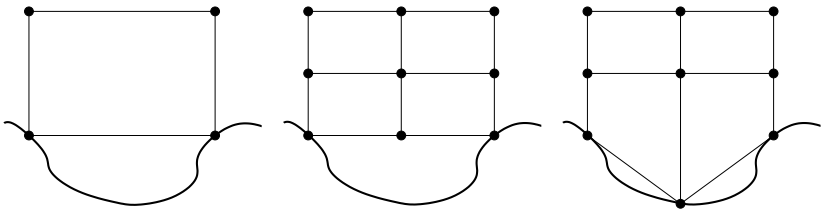


Abbildung 2.6: *Ein Beispiel für Randanpassung. links: vor der Unterteilung, Mitte: Unterteilung durch lineare Interpolation der Koordinaten, ohne Randanpassung, rechts: mit Randanpassung*

Wird die Position dieser Punkte nicht korrigiert, d.h. wird keine **Randanpassung** vorgenommen, so löst das feinere Gitter die gegebene Geometrie nicht besser auf als das grobe. Und da gerade bei den Navier-Stokes-Gleichungen physikalisch relevante Effekte durch den Rand hervorgerufen werden, ist wenig gewonnen.

Zur Randanpassung stehen zwei grundverschiedene Techniken zur Verfügung, ein parametrischer und ein geometrischer Zugang. Bei parametrischen Ansätzen wird die Position neuer Knoten nicht geometrisch bestimmt, sondern auf der Basis einer Parametrisierung des Randes: Jedem Randknoten wird bijektiv ein Parameterwert zugeordnet. Über die Parametrisierung lassen sich dann konkrete Koordinaten des Punktes berechnen. Ist der Rand etwa als B-Spline-Kurve gegeben, so wird diese Aufgabe gerade vom **de Boor**-Algorithmus gelöst (s. [1]). Um am Rand zu unterteilen, wird bei Randkanten der Parameterwert des neuen Knotens als Mittelwert der Parameterwerte der alten Knoten gewählt, bei Randflächen analog als Mittel der vier Eckpunkte (wobei Flächen im Raum typischerweise mit zwei Parameterwerten parametrisiert werden). In der Simulationssoftware FEATFLOW ist in 2D eine Randparametrisierung vorgesehen, in 3D existiert sie nicht.

Geometrische Zugänge unterteilen Randkanten und Randflächen genauso wie innere Flächen und unterscheiden sich in der gewählten Form der Randanpassung. Projektionsbasierte Techniken sind ein Schwerpunkt dieser Arbeit und werden mit ihren theoretischen Grundlagen in Kapitel 3 vorgestellt. Mögliche Parametrisierungstechniken in 3D werden im letzten Kapitel, das die Ergebnisse der Arbeit zusammenfaßt, diskutiert.

Unabhängig von der Vorgehensweise kann während der Randanpassung eine einfache Modifikation vorgenommen werden, um die Gitterqualität lokal zu verbessern: Die Knoten, die aus Kantenmittelpunkten entstehen, bleiben unverändert. Die Knoten, die aus Flächenmittelpunkten entstehen, werden nicht vor der Randanpassung aus diesen berechnet, sondern erst danach. Dies führt heuristisch immer zu einer Gitterverbesserung, ohne daß zusätzliche Berechnungen

vorgenommen werden müssen.

Klassifikation neuer Knoten

Innere Knoten Innere Knoten sind einfach zu klassifizieren: Ein Knoten, der durch Kantenunterteilung entsteht, ist innerer Knoten, wenn mindestens eine der beiden Ecken der Kante innerer Knoten ist. Ein Knoten, der als Mittelpunkt einer Seitenfläche entsteht, ist innerer Knoten, wenn die vier Ecken bereits innere Knoten sind. Ein Knoten, der als Mittelpunkt eines Hexaeders entsteht, ist immer innerer Knoten.

Randknoten Leider gibt es schon in 2D einfache Beispiele, die zeigen, daß die logische Umkehrung der Bedingungen für innere Knoten keine Bedingung zur Klassifikation von Randknoten ergibt. Ein Beispiel zeigt Abbildung 2.7.

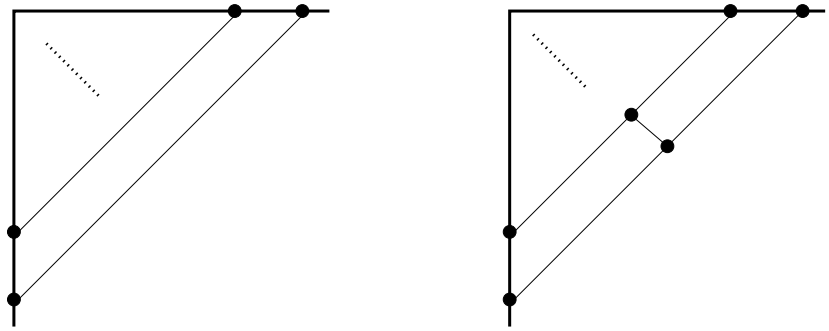


Abbildung 2.7: *Beispiel für Knoten, der durch Bisektion von Randkanten entstehen und nicht als Randknoten klassifiziert werden dürfen. Links: vor, rechts: nach der Unterteilung*

In 3D gibt es gerade bei unstrukturierten Gittern mit hängenden Knoten und variabler Konnektivität noch viel mehr mögliche Konstellationen. Hinzu kommt, daß nach der Klassifizierung eines Knotens noch der Zielrand bestimmt werden muß, sobald es mehr als eine Randkomponente gibt.
Für dieses Problem gibt es in der Literatur keine wirklich zufriedenstellenden Lösungsvorschläge. Daher wird pragmatisch vorgegangen. Wenn zur Diskretisierung ein reguläres Gitter aus Hexaederzellen gewählt wird, so daß Elemente immer mit einer kompletten Seitenfläche und nicht nur mit einer Kante auf dem Rand liegen, kann die Klassifizierung einfach über die Nachbarschaftsbeziehungen zwischen Elementen erreicht werden.

2.2.2 Glättung und Gittertransfer

Bevor der Mehrgitteralgorithmus detailliert vorgestellt werden kann, sind einige weitere Vorbemerkungen nötig. Die vorkommenden Bezeichner sind im vorherigen Abschnitt definiert worden.

DEFINITION 2.5 (GLÄTTER)
*Ein Operator S_h auf dem Ansatzraum V_h heißt **Glätter**, wenn er für die Lösung u_h des Gleichungssystems (2.8) die Fixpunkteigenschaft erfüllt, d.h.*

$$S_h u_h = u_h$$

und stabil ist, d.h.

$$||u_h - S_h v_h|| \leq c||u_h - v_h||$$

für eine Konstante $c > 0$ und eine beliebige Testfunktion $v_h \in V_h$. Die Konstante c darf von der Schrittweite h abhängen.

Man beachte, daß ein Glättungsoperator im Idealfall nicht von der Schrittweite abhängt. Glätter werden häufig nicht direkt, sondern über eine Iterationsvorschrift definiert (Fixpunktiteration, Defektkorrektur). Ausgangspunkt ist die im vorherigen Abschnitt aufgestellte allgemeine Defektkorrektur-Vorschrift (s. Formel 2.9):

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

Um diese Vorschrift zu verschiedenen Verfahren auszubauen, wird eine Zerlegung der Matrix A betrachtet,

$$A = L + D + U,$$

wobei D (engl. *diagonal*) nur die Einträge der Hauptdiagonale, L (engl. *lower triangular*) die linke untere Dreiecksmatrix und U (engl. *upper triangular*) die rechte obere Dreiecksmatrix als Teilmatrix von A enthält. Je nach Wahl der Korrekturmatrix C ergeben sich nun drei einfache Glättungsvorschriften:

Jacobi-Glätter Wähle C^{-1} als Inverse der Diagonalmatrix, $C = D$.

Gauss-Seidel-Glätter Wähle C^{-1} als Inverse der Diagonal- und unteren Dreiecksmatrix, $C = (L + D)$.

SOR-Glätter Relaxiere mit einem weiteren Parameter $\tilde{\omega}$ zusätzlich L: $C = (D + \tilde{\omega}L)$.

Man beachte, daß die Invertierung der Matrix C jeweils einfach und in linearer Zeit möglich ist. Diese Verfahren lassen sich effizient implementieren, insbesondere ist es nicht nötig, die Defektkorrekturmatrix explizit aufzustellen. Man bezeichnet sie daher auch als *matrixfrei*.

Ein anderer Ansatz verwendet die LU-Zerlegung der Matrix A. Mit Hilfe des Gauß-Algorithmus läßt sich jede positiv definite diagonaldominante Matrix schreiben als Produkt einer unteren und einer oberen Dreiecksmatrix:

$$A = LU$$

Mit Hilfe dieser Zerlegung kann man die Lösung eines linearen Gleichungssystems auf die Lösung zweier gekoppelter Systeme zurückführen:

$$A\mathbf{x} = \mathbf{b} \Leftrightarrow LU\mathbf{x} = \mathbf{b} \Leftrightarrow L\mathbf{y} = \mathbf{b} \text{ und } U\mathbf{x} = \mathbf{y}$$

Wie bei den einfachen Defektkorrekturverfahren ist so noch nichts gewonnen: Die Berechnung der LU-Zerlegung ist zu aufwendig, und es läßt sich sogar zeigen, daß die Zerlegung dünn

besetzter Matrizen (wie sie in den betrachteten Anwendungsfällen der Lösung der Navier-Stokes-Gleichungen mittels Finiter Elemente vorkommen) leider nicht mehr dünn besetzt ist und damit Speicherprobleme bei der Lösung durch den Computer hervorruft. Stattdessen geht man zur unvollständigen Zerlegung (**ILU-Zerlegung**) über und läßt in den Matrizen $\tilde{L}$ und $\tilde{U}$ nur die Einträge zu, die schon in A von Null verschieden sind. Untersuchungen, welche Auswahl von Einträgen zu welchem Konvergenzverhalten führt, werden in [31, 32] vorgestellt. Dieses Verfahren läßt sich auch wieder als Fixpunktiteration formulieren:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \omega(\tilde{L}\tilde{U})^{-1}(\mathbf{b} - A\mathbf{x}^{(k)})$$

Weiter werden zur Formulierung des allgemeinen Mehrgitterverfahrens in seiner hier relevanten diskreten Version (s.o.) noch sogenannte *Gittertransferoperatoren* benötigt. Sie dienen dazu, Defektkorrekturen von einem Gitter auf ein anderes zu übertragen. Für eine detaillierte funktionalanalytische Beschreibung wird auf [9, 11] verwiesen.

DEFINITION 2.6 (GITTERTRANSFEROPERATOREN)
*Der Schritt, einen Koeffizientenvektor bzw. einen Defektkorrekturterm von einem feinen Gitter k auf das nächstgröbere Gitter $k-1$ zu übertragen, heißt **Restriktion**, geschrieben R_k^{k-1}, der umgekehrte **Prolongation**, geschrieben P_{k-1}^k.*

2.2.3 Diskrete Formulierung des Mehrgitteralgorithmus

Nach diesen Vorarbeiten kann nun der Mehrgitteralgorithmus formuliert werden. Er wird in seiner „natürlichen" rekursiven Form dargestellt (böse Zungen behaupten, daß nur eine nichtrekursive Form entwickelt wurde, weil die Programmiersprache FORTRAN 77 als erste und beliebteste Implementierungssprache keine Rekursion unterstützt). Für seine Beschreibung sind folgende Bezeichner nötig:

k	aktueller Level, $k = 0$ bezeichne das Grobgitter, $k = k_{max}$ das feinste Gitter
$\mathbf{u}_k^j, \mathbf{v}_k^j$	(näherungsweise) Lösung nach j Glättungsschritten
$\mathbf{u}_k^*$	(näherungsweise) Lösung
$\mathbf{d}_k$	Defekt

Man beachte, daß im Gegensatz zur bisherigen Formulierung die Probleme abhängig von dem Level formuliert sind.

Übergabeparameter $k, \mathbf{u}_k, \mathbf{b}_k$

Rückgabewert $\mathbf{u}_k^*$

Aufruf `MEHRGITTER`$(k_{max}, 0, \mathbf{b}_{k_{max}})$

1. Rekursionsabbruch: Falls das aktuelle Gitter das Grobgitter ist ($k = 0$), löse das *Grobgitterproblem* exakt und gebe die Lösung zurück.

$$\mathbf{u}_0^* = A_0^{-1}\mathbf{b}_0$$

2. Sonst führe die folgenden Schritte durch:

   (a) Wende einen **Vorglätter** S_k μ Mal an:

$$\mathbf{u}_k^{(0)} = u_k^*$$

$$\mathbf{u}_k^{(i)} = S_k\mathbf{u}_k^{(i-1)} \text{ für } i = 1, \ldots, \mu$$

   (b) Bilde den Defekt:

$$\mathbf{d}_k = \mathbf{b}_k - A_k\mathbf{u}_k^{(\mu)}$$

   (c) Wende den Restriktionsoperator zur Transferierung auf das nächstgröbere Gitter an:

$$\mathbf{d}_{k-1} = R_k^{k-1}\mathbf{d}_k$$

   (d) Führe die sogenannte **Grobgitterkorrektur** durch, d.h. löse das Problem approximativ durch rekursiven Aufruf mit p Mehrgitterschritten.

$$\mathbf{v}_k^{(0)} = 0$$

$$\mathbf{v}_{k-1}^{(j)} = \texttt{MEHRGITTER}(k-1, \mathbf{v}_k^{(p-1)}\mathbf{d}_{k-1}) \text{ für } j = 1, \ldots, p$$

   (e) Prolongiere das Ergebnis zurück auf Level k und führe die Defektkorrektur durch:

$$\mathbf{u}_k^{\mu+1} = \mathbf{u}_k^{\mu} + P_{k-1}^k\mathbf{v}_{k-1}^{(p)}$$

   (f) Führe nochmals ν Glättungsschritte auf diesem Vektor durch und liefere ihn als Rückgabewert.

$$\mathbf{u}_k^{(\mu+1+j)} = S_k\mathbf{u}_k^{(\mu+1+j-1)} \text{ für } j = 1, \ldots, \nu$$

$$\text{return } \mathbf{u}_k^{(\mu+\nu+1)}$$

Algorithmus 2.1: Diskrete Formulierung des Mehrgitter-Algorithmus `MEHRGITTER`

Der Algorithmus verwendet also als freie Parameter die Wahl des Glätters und die Anzahl der jeweils durchgeführten Vor- und Nachglättungsschritte. Der Parameter p steuert die Form der sogenannten *Mehrgitterzyklen*. Es sind drei Zyklen gebräuchlich, V ($p = 1$), W ($p = 2$) und F. Der F-Zyklus folgt dem V-Zyklus beim Abstieg und dem W-Zyklus beim Aufstieg.

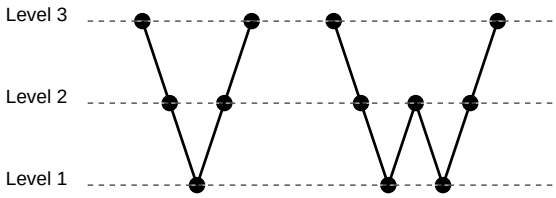


Abbildung 2.8: *Mehrgitter-Zyklen*

Zu den Glättungsoperatoren ist zu betonen, daß sie nicht die Lösung glätten, sondern den Defekt. Außerdem werden sie in Abhängigkeit der Parameter μ, ν nicht bis zur Konvergenz iteriert, sondern nur eine feste Anzahl Schritte durchgeführt.

Detaillierte Konvergenzbeweise für die verschiedenen Zyklen können in [9, 24] nachgeschlagen werden. [65] liefert eine gute Übersicht über das Verhalten in Abhängigkeit von den freien Parametern. Zusammenfassend liefern diese Quellen die folgenden Resultate: Die Konvergenzraten sind im allgemeinen sehr gut und insbesondere unabhängig von der Gitterweite. Die Laufzeit ist (bei fester Anzahl Glättungsschritte) linear in der Anzahl der Knoten bzw. Elemente auf dem feinsten Gitter, sofern der Aufwand zur (exakten) Lösung des Grobgitterproblems vernachlässigt werden kann. Somit ist der Mehrgitteralgorithmus ein sehr mächtiges Werkzeug zur Lösung von Strömungssimulationen mittels Finiter Elemente.

2.3 Gitterglättung

Gitterglättung (nicht zu verwechseln mit der Glättung einer Lösung bzw. eines Defekts während einer Mehrgitter-Iteration) bezeichnet den Vorgang, ein existierendes Gitter allein durch (lokale) Koordinatenverschiebung von Knoten so zu verändern, daß die einzelnen Elemente im Mittel eine höhere Qualität erreichen. Die Konnektivität eines Gitters ist davon nicht betroffen. In diesem Abschnitt werden verschiedene Glättungsverfahren vorgestellt.

2.3.1 Knotenbasierte Glättungsalgorithmen

Knotenbasierte Glättungsalgorithmen, wie sie hier betrachtet werden, lassen sich auffassen als (beliebige) Kombination einfacher Komponenten:

Zielknoten Es muß unterschieden werden zwischen der Glättung von inneren Knoten (bei der keine Positionen außerhalb des Gebiets generiert werden dürfen), und der Glättung von Randknoten, bei denen eine Rückprojektion der neuen Position auf den Rand nötig ist.

Globaler oder lokaler Zugang Einfache Algorithmen verwenden Punkte aus der lokalen Nachbarschaft zur Berechnung neuer Positionen. Globale Ansätze minimieren Energien (s.u.), die über dem gesamten Gitter definiert sind.

Lokale Gewichtung, Träger Bei lokalen Ansätzen existieren sehr viele Freiheitsgrade in der Wahl der Gewichtung der einzelnen Knoten der Nachbarschaft, die zur Berechnung der neuen Position kombiniert werden.

Jacobi oder Gauss-Seidel Die Unterscheidung zwischen Jacobi-artigen Glättern und Glättern vom Gauss-Seidel-Typ besteht darin, daß erstere erst alle neuen Positionen berechnen, und dann alle Koordinaten im Netz tatsächlich ändern. Letztere verwenden zur Berechnung neuer Positionen bereits bekannte neue Koordinaten für Punkte in der Nachbarschaft.

In den folgenden drei Abschnitten werden zunächst einige Ansätze vorgestellt, ihre Eigenschaften analysiert und Vor- und Nachteile gegenübergestellt. Im letzten Kapitel der Arbeit (s. Kap. 5.3) werden einige Operatoren experimentell analysiert.

Der Laplace-Operator Der Laplace-Operator berechnet die neuen Koordinaten $\mathbf{p}_{neu}$ eines Knotens $\mathbf{p}$ nach folgender Vorschrift:

$$\mathbf{p}_{neu} := (1-\alpha)\cdot\mathbf{p} \;+\; \frac{\alpha}{|Adj(\mathbf{p})|}\sum_{\mathbf{q}_j\in Adj(\mathbf{p})}\mathbf{q}_j \tag{2.10}$$

Dabei ist $\alpha \in [0;1]$ ein Gewichtsfaktor, der die lineare Interpolation zwischen alter und neuer Position steuert. Die Menge $Adj(\mathbf{p})$ enthalte für jeden Knoten $\mathbf{p}$ die Knoten, die mit $\mathbf{p}$ durch eine Kante verbunden sind.

Die Position jedes Knotens wird also auf den Mittelwert aller durch eine Kante benachbarten Knoten gesetzt. Dieses Verfahren arbeitet rein heuristisch, hat keinerlei Kontrolle auf die Qualität des Gitters und neigt dazu, an konkaven Rändern invertierte Elemente zu erzeugen. Abbildung 2.9 zeigt ein Beispiel für dieses Verhalten.

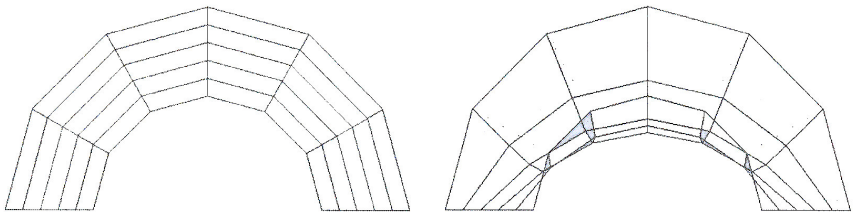


Abbildung 2.9: *Beispiel für Elementinvertierungen beim Laplace-Operator*

Es wurden mehrere Techniken vorgeschlagen, mit der die Laplace-Glättung stabilisiert werden kann. Die Kernidee ist jeweils, parallel zur Knotenverschiebung Qualitätstests (s. Kap. 4.4) durchzuführen und so Elementinvertierungen zu vermeiden. In der Literatur werden diese Ansätze unter dem Begriff *„smart laplace"* zusammengefaßt. Eine Übersicht bietet beispielsweise [46]. Ein vollkommen anderer Ansatz wird in [38, 39] verfolgt: Der Autor läßt zunächst Elementinvertierungen zu und verwendet algebraische Methoden, um Hexaedergitter zu „entwirren" (engl. *mesh untangling*). Er schreibt jedoch selbst, daß diese Idee zwar vielversprechend, jedoch noch nicht in allgemeinen Fällen anwendbar ist.

In konvexen Geometrien und als Komponente spezialisierter Algorithmen spielt die Laplace-Glättung trotz ihrer Nachteile eine wichtige Rolle. Insbesondere eignet sich der Laplace-Operator

zur Glättung von Gittern, die durch Elementunterteilungen aus bereits existierenden „guten" Gittern entstehen, da in diesem Fall das Risiko, invertierte und deformierte Elemente zu produzieren, geringer ist.

Der Umbrella-Operator Im Bereich des *multiresolution modelling* wird die Frage untersucht, wie ein geometrisches Objekt, beispielsweise gegeben als Oberflächentriangulierung, in verschiedenen Auflösungen dargestellt werden kann. Der englische Fachbegriff für diese Aufgabe lautet *remeshing*. In seiner Dissertation (s. [40]) hat Kobbelt den sogenannten **Umbrella-Operator** hergeleitet und analysiert. Nach meinem Kenntnisstand wird dieser Operator bisher allein zur Qualitätsverbesserung von einzelnen Oberflächennetzen innerhalb einer *multiresolution*-Darstellung angewendet. In dieser Arbeit wurde daher seine Eignung zur Glättung von Finite-Elemente-Gittern experimentell untersucht.

Zur Herleitung des Operators betrachtet man die Spannungsenergie E_M und die Biegeenergie E_{TP} (engl. *membrane and thin plate energy*) einer Oberfläche:

$$E_M(f) := \int f_u^2 + f_v^2 \qquad E_{TP}(f) := \int f_{uu}^2 + 2f_{uv}^2 + f_{vv}^2$$

Gesucht ist eine Funktion (d.h. eine parametrische Darstellung der Oberfläche über einem 2D-Gebiet $I \times J$)

$$f(u,v), f : I \times J \to \mathbb{R}^3,$$

die diese Energien minimiert. Mit Variationsrechnung läßt sich das Problem umformulieren zu den Euler-Langrange-Gleichungen

$$\Delta f = f_{uu} + f_{vv} = 0 \tag{2.11}$$

bzw.

$$\Delta^2 f = f_{uuuu} + 2f_{uuvv} + f_{vvvv} = 0 \tag{2.12}$$

Schon einfache Oberflächen sind in der Regel nicht geschlossen isoparametrisch (d.h. unter Beibehaltung der Längen- und Winkelverhältnisse) parametrisierbar, man betrachte als Beispiel die Aufgabe, die Erdkugel auf die 2D-Fläche einer Landkarte zu projizieren. Daher wird zur Diskretisierung dieser Gleichungen mit dividierten Differenzen eine lokale Parametrisierung in der Umgebung eines Punktes $\mathbf{p}$ gesucht. Kobbelt zeigte, daß unter Verwendung der Parametrisierung (n sei die Valenz von $\mathbf{p}$)

$$(u_i, v_i) := \left(\cos(2\pi\frac{i}{n}), \sin(2\pi\frac{i}{n})\right), \quad i = 1, \ldots, n-1 \tag{2.13}$$

die diskrete Version des Laplace-Operators Δf aus Gleichung 2.11 gerade der Umbrella-Operator

$$U(\mathbf{p}) = \frac{1}{n} \sum_{\mathbf{q}_j \in Adj(\mathbf{p})} \mathbf{q}_j - \mathbf{p} \tag{2.14}$$

ist. Rekursive Anwendung liefert eine Diskretisierung für Gleichung 2.12.

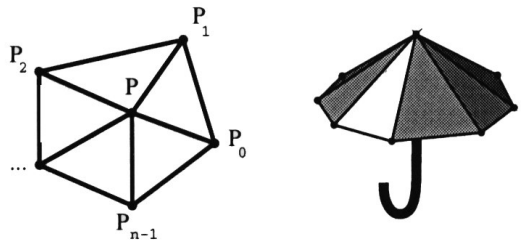


Abbildung 2.10: *Der Umbrella-Operator*

Eine modifizierte Formulierung des Operators aus [41] wird auch als *density weighted umbrella operator* bezeichnet. Dabei bezeichnet d_j die mittlere Länge aller Kanten, die von $\mathbf{q}_j$ ausgehen:

$$\mathbf{p}_{neu} := (1-\alpha)\mathbf{p} + \frac{\alpha}{\sum d_j} \sum_{\mathbf{q}_j \in Adj(\mathbf{p})} d_j \mathbf{q_j} \tag{2.15}$$

Im wesentlichen handelt es sich um einen Laplace-Operator, der durch geeignete Wahl der Gewichte das Verhältnis der Kantenlängen in einem Punkt annähernd erhält. So wird eine Häufung von Punkten verhindert. Der Dämpfungsfaktor α steuert das Gewicht der ursprünglichen Position, eine initiale Wahl von $\alpha = 0.5$ bietet sich an.

In [41] wird die Wirkungsweise dieses Operators auf das *remeshing* polygonaler Oberflächen untersucht. Im Rahmen dieser Arbeit wurden numerische Experimente zur Wirkungsweise des Operators zur Glättung von Finite-Elemente-Diskretisierungen vorgenommen. Die Ergebnisse finden sich in Kapitel 5.3.

Beispiele Die Abbildungen 2.11, 2.12 und 2.13 verdeutlichen die Unterschiede zwischen dem Laplace-Operator und dem Umbrella-Operator:

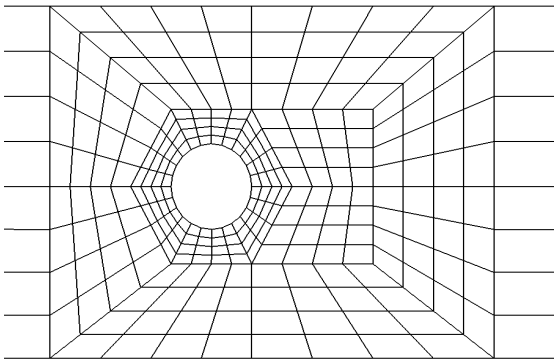


Abbildung 2.11: *Ausgangssituation: ungeglättetes Gitter*

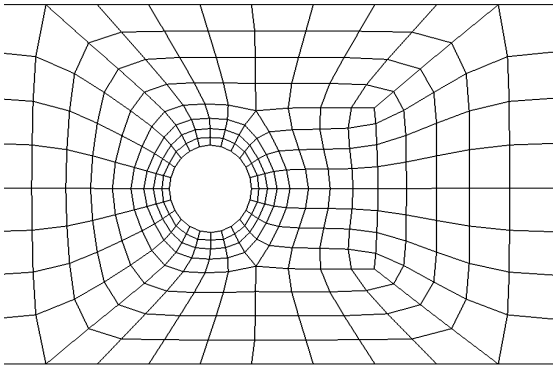


Abbildung 2.12: *Dreimalige Anwendung des Laplace-Operators auf alle inneren Knoten*

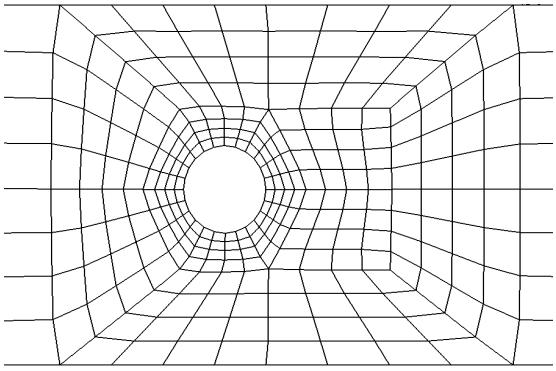


Abbildung 2.13: *Dreimalige Anwendung des Umbrella-Operators auf alle inneren Knoten*

2.3.2 Randglättung

Randglättung bezeichnet den Vorgang, die Menge der Seitenflächen von Hexaedern, die mit ihren vier Knoten auf einem Rand liegen, zu glätten. Hierzu können offenbar alle bisher vorgestellten Techniken verwendet werden. Die einzige nötige Modifikation besteht in der geänderten Definition des Trägers: Anstatt als Nachbarknoten die Knoten zu verwenden, die mit einer Kante verbunden sind, werden die Knoten verwendet, die mit einer Kante verbunden sind und auf demselben Rand liegen.

Dieses Vorgehen stellt allerdings nicht sicher, daß die neu berechneten Knotenpositionen geometrisch auf dem Rand liegen. Nach der Anwendung einer Randglättung muß also eine Projektion (s. Kap. 3.4) der Punkte zurück auf den Rand erfolgen.

Die Randglättung wird in dieser Arbeit bei der Definition eines robusten Projektionsalgorithmus (s. Kap. 3.4.3) eine wichtige Rolle spielen.

2.3.3 Weiterführende Ansätze

Eine Möglichkeit besteht darin, die Glättung als Optimierungsaufgabe zu formulieren. Minimiert wird eine globale Gitterfunktion, beispielsweise die Winkelabweichung oder jede andere zur Verfügung stehende Qualitätsmetrik (s. Kap. 4.4). Beispiele finden sich in [37, 38].

Physikalische Modelle interpretieren beispielsweise die Kanten als Federn zwischen Knoten und minimieren dann die Spannungsenergie all dieser Federn, gesucht wird also ein Gleichgewichtszustand für das Gitter (s. [42]).

Eine weitere Idee besteht darin, Elemente auf Referenzelemente in der Ebene oder im Raum zu transformieren, die nicht verzerrt sind. Es wird also in zwei Koordinatensystemen gearbeitet, die über die Jacobi-Transformation ineinander umgerechnet werden können. Integriert man nun Teile dieser Jacobi-Matrix (engl. *deformation gradient*) in den Glätter, so ergeben sich Glätter auf der Basis von partiellen Differentialgleichungen. Der Prototyp all dieser Techniken wurde schon 1967 von Winslow vorgeschlagen (s. [64]), neuere Veröffentlichungen wie [26] zeigen die unverminderte Aktualität dieser Ansätze.

Alle weiterführenden Ideen erfordern im Vergleich zu den einfachen koordinatenbasierten Glättungsoperatoren einen deutlich höheren Aufwand und sind somit rechenzeitintensiver.

Kapitel 3

Oberflächentriangulierungen und Projektionstechniken

3.1 Oberflächentriangulierungen: Grundlagen

3.1.1 Definitionen

Dieser Abschnitt listet in gebotener Kürze notwendige Definitionen und Zusammenhänge auf. Die Darstellung basiert auf den Lehrbüchern [1, 16, 17, 19, 30].

DEFINITION 3.1 (DREIECK)
*Ein **Dreieck** T ist die konvexe Hülle dreier beliebiger nicht kollinearer Punkte $\mathbf{p}_1$,$\mathbf{p}_2$,$\mathbf{p}_3 \in \mathbb{R}^n$. Die Punkte heißen auch **Ecken** oder **Knoten**. Die Strecken $\overline{\mathbf{p}_i\mathbf{p}_j}$ $(i, j \in \{1, 2, 3\}, i \neq j)$ werden als **Kanten** bezeichnet. Kanten sind somit konvexe Hüllen zweier verschiedener Knoten.*

DEFINITION 3.2 (OBERFLÄCHENTRIANGULIERUNG)
*Eine Menge $T = \{T_1, \ldots, T_m\}$ von Dreiecken heißt **Triangulierung** (engl.* triangulation*), wenn für $i \neq j$, $i, j \in \{1, \ldots, m\}$ einer der drei folgenden Fälle gilt:*

(i) $T_i \cap T_j = \varnothing$

(ii) $T_i \cap T_j$ *ist ein Knoten*

(iii) $T_i \cap T_j$ *ist eine Kante*

*Triangulierungen werden auch als **Dreiecksnetze** bezeichnet. Eine Triangulierung heißt **geschlossen**, wenn jede Kante im Schnitt von genau zwei Dreiecken liegt. Eine Triangulierung heißt **Oberflächentriangulierung** eines Volumens (engl.* surface triangulation*), wenn sie geschlossen ist und die Oberfläche des Volumens vollständig abdeckt.*

Eine Oberflächentriangulierung ist somit eine stückweise lineare, global stetige Beschreibung einer geschlossenen Fläche im Raum. Höhere Differenzierbarkeit ist im Kontrast zu beispielsweise Spline- oder Unterteilungsflächen nicht möglich. Andererseits kann eine eine glatte Fläche bzw. eine gegebene Krümmung beliebig gut approximiert werden, wählt man als Kriterium zur Approximationsgüte beispielsweise die Winkelabweichung zweier benachbarter Punkte auf der Fläche. Oberflächentriangulierungen sind die vielseitigste Repräsentationsform zur Beschreibung geometrischer Objekte und werden in der Praxis häufig als Approximation einer gegebenen Fläche höherer Ordnung verwendet, moderne Grafikkarten stellen beispielsweise dreidimensionale Szenen nur als Menge von Oberflächentriangulierungen dar.

3.1.2 Datenstrukturen

Dreiecksnetze sind im wesentlichen nichts anderes als (simpliziale) Zellzerlegungen, somit stehen alle für diese Strukturen aus der Computergrafik bekannten Repräsentationen zur Verfügung. Hier sollen nur einige Ansätze vorgestellt werden, tiefergehende Informationen können jedem der genannten Lehrbücher über Computergrafik und geometrisches Modellieren entnommen werden.

In der **Listendarstellung** werden alle Dreiecke durch Angabe der kartesischen Koordinaten ihrer drei Eckpunkte in einer linearen Liste gespeichert. Diese Datenstruktur benötigt viel Speicherplatz für redundante Informationen, da alle Eckpunkte mehrfach explizit aufgelistet werden. Trotzdem wird sie beispielsweise im `DXF`-Dateiformat [3] verwendet.

Als Standardformat durchgesetzt hat sich die sog. **shared vertex** Darstellung. Hier werden die Koordinaten der Eckpunkte in einer Liste gehalten und die Dreiecke in einer anderen, wobei zu jedem Dreieck nur noch Verweise auf die drei Punkte in der anderen Liste gespeichert werden. Dieses Format enthält keine Redundanz mehr.

Der Vorteil beider Strukturen besteht in der dynamischen Modifizierbarkeit, der Hauptnachteil besteht darin, daß zur Bestimmung von Nachbarschaftsbeziehungen lineare Zeit pro Dreieck nötig ist. Ein Ausweg ist die (leider statische) **directed edges** Datenstruktur (s. [10]), einer Modifikation der shared vertex Darstellung, die mit nur geringem zusätzlichen Speicherplatzbedarf den Zugriff auf Nachbarschaftsinformationen in konstanter Zeit bietet. Andere Alternativen, beispielsweise Zellinzidenzgraphen auf der Basis doppelt verketteter Listen oder die **winged edge** Datenstruktur (s. [1]) benötigen im Vergleich dazu deutlich mehr Speicherplatz, um Nachbarschaftsanfragen effizient beantworten zu können.

3.2 Oberflächentriangulierungen: Wichtige Hilfsmittel

Dieses Kapitel listet einige Formeln und Zusammenhänge auf, die für Operationen auf Dreiecksnetzen in den folgenden Kapiteln benötigt werden. Die elementaren Beweise finden sich in jedem guten Lehrbuch über analytische Geometrie.

3.2.1 Schnittprobleme

DEFINITION 3.3 (BARYZENTRISCHE KOORDINATEN)
Die Koeffizienten $(\lambda_1, \ldots, \lambda_m)$ *einer Linearkombination*

$$\mathbf{v} = \sum_{i=1}^{m} \lambda_i \mathbf{v}_i \tag{3.1}$$

mit einem beliebigen Erzeugendensystem $\{\mathbf{v}_1, \ldots, \mathbf{v}_m\} \subset \mathbb{R}^n$ *heißen* ***baryzentrische Koordinaten*** *des Punktes* $\mathbf{v}$ *bezogen auf die aufspannenden Vektoren* $\mathbf{v}_1, \ldots, \mathbf{v}_m$*. Gilt zusätzlich*

$$0 \leq \lambda_i \leq 1 \quad \forall i \in \{1, \ldots, m\} \text{ und } \sum_{i=1}^{m} \lambda_i = 1$$

so spricht man von einer ***Konvexkombination****.*

Angewendet auf den Spezialfall eines Dreiecks im Raum bedeutet diese Definition:

- Die aufspannenden Vektoren für ein Dreieck sind gerade die (Ortsvektoren der) Eckpunkte $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$. Man beachte, daß in der Definition von einem Erzeugendensystem die Rede ist, nicht notwendig von einer Basis.

- Die Eckpunkte des Dreiecks haben die baryzentrischen Koordinaten $(1, 0, 0), (0, 1, 0)$ und $(0, 0, 1)$. Dies liest man an Gleichung 3.1 ab.

- Die Umrechnung der beiden Koordinatensysteme ineinander ist ebenfalls durch Gleichung 3.1 festgelegt.

- Die baryzentrischen Koordinaten eines Punktes im Dreieck sind lineare Funktionen. Auf der Geraden durch $\mathbf{p}_2$ und $\mathbf{p}_3$ gilt $\lambda_1 = 0$. Für alle Punkte $\mathbf{p}$, die auf der gleichen Seite dieser Geraden liegen, hat λ_1 gleiches Vorzeichen. Insbesondere gilt $\lambda_1 > 0$ für alle Punkte, die auf der gleichen Seite liegen wie $\mathbf{p}_1$. Entsprechendes gilt für die Punkte $\mathbf{p}_2$ und $\mathbf{p}_3$ mit ihren gegenüberliegenden Seiten.

- Reduziert man das Erzeugendensystem auf eine Basis, für ein Dreieck also auf zwei Kantenvektoren, so gelten diese Beobachtungen natürlich nicht mehr. Dafür lassen sich andere Resultate wesentlich einfacher herleiten. Die folgenden Abschnitte verwenden beide Varianten.

Aus diesen Bemerkungen folgt direkt die wichtige Beobachtung:

FOLGERUNG 3.4
Ein Punkt liegt genau dann im Inneren eines Dreiecks, wenn seine baryzentrischen Koordinaten bezüglich des Dreiecks eine Konvexkombination bilden.

Schnitt Gerade-Dreieck

SATZ 3.5 (SCHNITT STRAHL–DREIECK)

Der Schnittpunkt $\mathbf{s}$ *im* $\mathbb{R}^3$ *einer Gerade* $g : \{\mathbf{a} + \gamma\mathbf{b}\}$ *mit einem Dreieck gegeben durch die Eckpunkte* $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ *ergibt sich aus der Lösung* (α, β, γ) *des linearen Gleichungssystems*

$$\mathbf{p}_1 + \alpha(\mathbf{p}_2 - \mathbf{p}_1) + \beta(\mathbf{p}_3 - \mathbf{p}_1) = \mathbf{a} + \gamma\mathbf{b} \tag{3.2}$$

wie folgt:

1. *Falls Gerade und Dreieck nicht parallel sind, liegt genau dann ein Schnittpunkt vor, wenn* $\alpha, \beta \in [0, 1]$ *mit* $\alpha + \beta = 1$.
2. *Dann erhält man die Koordinaten des Schnittpunkts durch Einsetzen in die Geradengleichung* g.

Gleichung (3.2) ist ein lineares Gleichungssystem mit drei Gleichungen und drei Unbekannten. Unter der Nichtparallelitätsbedingung hat es vollen Rang, sonst ist das Schnittproblem trivial lösbar. Anschaulich passiert folgendes: Zuerst wird die Gerade mit der Ebene geschnitten, die vom Dreieck aufgespannt wird. Dieser Test liefert unter obiger Voraussetzung immer einen Schnittpunkt. Dann werden die Koeffizienten des Schnittpunkts als baryzentrische Koordinaten bzgl. des Dreiecks aufgefaßt, um zu testen, ob er im Inneren des Dreiecks liegt. Die Berechnung der Schnittkoordinaten ist dann trivial. Wird zusätzlich eine Vorzeichenbedingung an γ vorgeschrieben, so wird der Schnittpunkt eines Dreiecks mit einem Strahl von $\mathbf{a}$ in Richtung $\mathbf{b}$ bzw $-\mathbf{b}$ berechnet. Dieses Verfahren ist sehr effizient, weil außer der Lösung eines 3×3-Systems (beispielsweise mit der Cramerschen Regel) wesentlich nur Vergleiche anfallen.

Odd-Even-Test

Um für einen beliebigen polygonal berandeten Körper K und einen Punkt $\mathbf{p}$ festzustellen, ob $\mathbf{p}$ im Inneren von K liegt, hat sich das folgende Zählverfahren durchgesetzt:

SATZ 3.6 (PUNKT–IN–VOLUMEN–TEST)

Sei $\mathbf{r} \neq \mathbf{0}$ *ein beliebiger Vektor.*
Wenn die Anzahl der echten Schnittpunkte (d.h. ohne Berührungspunkte) des Strahls $s : \{\mathbf{p} + \gamma\mathbf{r}, \quad \gamma \geq 0\}$ *mit dem Rand von* K *gerade ist, so liegt* $\mathbf{p}$ *außerhalb von* K*, ist sie ungerade, so liegt* $\mathbf{p}$ *innerhalb von* K.

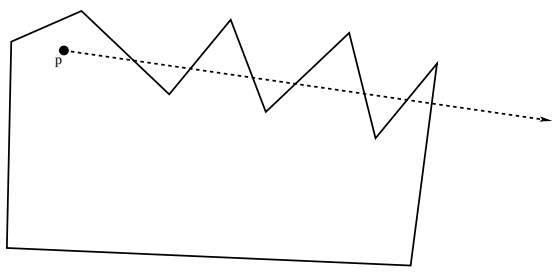


Abbildung 3.1: $\mathbf{p}$ *liegt im Inneren, weil die Anzahl der Schnittpunkte ungerade ist*

Da es sich in der Praxis für den dreidimensionalen Fall als nichttrivial herausstellt, alle möglichen auftretenden Sonderfälle zu berücksichtigen, kann folgende Heuristik verwendet werden: Das obige Verfahren wird mehrmals mit jeweils neuem, randomisiert gewählten Strahl durchgeführt und anschließend eine Mehrheitsentscheidung getroffen.

Für Körper, deren Oberfläche durch ein geschlossenes Dreiecksnetz gegeben ist, muß lediglich Gleichung 3.2 für jedes Dreieck in der Oberflächentriangulierung gelöst und die Trefferzahlen aufsummiert werden. Effizientere Techniken, die nicht alle Dreiecke testen, wenn bereits vor dem Test sicher ist, daß kein weiterer Treffer hinzukommt, werden in Kapitel 3.3 vorgestellt.

3.2.2 Abstandsprobleme

Abstände sind im Kontext dieser Arbeit immer euklidisch zu verstehen:

DEFINITION 3.7 (ABSTAND)
Ein Punkt $\mathbf{p} \in \mathbb{R}^3$ *hat von einem anderen Punkt* $\mathbf{q} \in \mathbb{R}^3$ *den* ***kürzesten Abstand*** d*, wenn gilt:*

$$d := ||\mathbf{p} - \mathbf{q}|| \leq ||\mathbf{p}' - \mathbf{q}|| \quad \textit{für alle } \mathbf{p}' \in \mathbb{R}^3 \setminus \{\mathbf{q}\}$$

Liegt $\mathbf{p}$ *auf der Oberfläche* S *eines Volumens im Raum, so heißt* $\mathbf{p}$ ***Lotfußpunkt*** *von* $\mathbf{q}$ *auf* S*.*

Kürzeste Abstände sind immer eindeutig, der Punkt, in dem der kürzeste Abstand angenommen wird, ist in der Regel jedoch nicht eindeutig bestimmt. Ein Beispiel zeigt Abbildung 3.2. Dies wird sich als Hauptnachteil für eine der folgenden Projektionstechniken erweisen.

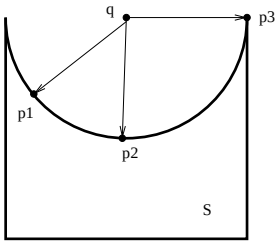


Abbildung 3.2: *Beispiel für nicht eindeutig bestimmte Lotfußpunkte:* $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ *sind gleichermaßen Lotfußpunkte von* $\mathbf{q}$ *auf* S*.*

Im Bezug auf Dreiecksnetze stehen für die Lage des Lotfußpunktes drei Möglichkeiten zur Verfügung:

1. Eckpunkte
2. Punkte auf Kanten
3. Punkte im Inneren eines Dreiecks

Experimente mit Dreiecken, die stark unterschiedlich lange Kanten haben, zeigen, daß die naheliegenden Heuristiken, beispielsweise den geometrischen Mittelpunkt des Dreiecks oder Kantenmittelpunkte als approximierten Lotfußpunkt zu verwenden, zu falschen Ergebnissen führen. Deshalb wird der kürzeste Abstand eines Punktes $\mathbf{q}$ zu einer Oberflächentriangulierung $T = \{T_1, \ldots, T_m\}$ wie folgt exakt berechnet:

Iteriere über alle Dreiecke $T_i \in T$:

1. berechne die Normale $\mathbf{n}_i$ des Dreiecks T_i als Kreuzprodukt zweier Dreiecksseiten
2. löse das Strahl-Schnitt-Problem mit Gleichung 3.2 für das Dreieck T_i und die Gerade $g : \{\mathbf{q} + \alpha\mathbf{n}_i\}$
3. berechne Abstand und Lotfußpunkt durch Einsetzen in die Geradengleichung
4. berechne für jede Kante von T_i den Lotfußpunkt
5. ist ein kürzerer Abstand als bisher gefunden, so speichere diesen als neues Zwischenergebnis

Algorithmus 3.1: Berechnung der Lotfußpunkte auf einer Oberflächentriangulierung

Schritt (2) führt immer dann zum Erfolg, wenn der Ausgangspunkt innerhalb des „dreiecksförmigen Zylinders" über dem Dreieck im Raum liegt. Ist so ein Lotfußpunkt gefunden, so kann der Algorithmus beendet werden, da es sich automatisch um den kürzesten Abstand handelt. Andernfalls muß der kürzeste Abstand auf einer der Dreieckskanten angenommen werden. Dies ist aber das aus der Schulmathematik bekannte Problem der Berechnung des Lotfußpunktes auf einer Geraden – kombiniert mit der Einschränkung des Parameterwertes auf die Länge der Kante – und kann durch einfache Formelauswertung gelöst werden. Der gesamte Algorithmus ist somit korrekt. Die Laufzeit setzt sich zusammen aus der Zeit zur Lösung eines 3×3-Systems und zur Berechnung von sechs Skalarprodukten. Sonst fallen nur Vergleiche an, die Normale des Dreiecks sollte – wird dieser Algorithmus häufig angewandt – vorberechnet werden.

Für Oberflächentriangulierungen kann der Algorithmus linear für jedes Dreieck aufgerufen werden, alternativ sollte eine der Suchdatenstrukturen aus dem nächsten Abschnitt verwendet werden.

3.3 Effizientere Datenstrukturen und Algorithmen

Ziel dieses Abschnitts ist es, für die im vorherigen Kapitel entwickelten Algorithmen effizientere Datenstrukturen anzugeben, um die lineare Suche zu vermeiden. Ausgangspunkt ist die folgende Beoachtung: Alle Verfahren (Schnitt Strahl-Dreieck (Satz 3.5), Punkt-in-Volumen-Test (Satz 3.6) und Abstandsberechnung (Algorithmus 3.1)) lassen sich auf das Problem reduzieren, für einen gegebenen Strahl und eine Menge von Objekten (Dreiecken) die Teilmenge der Objekte zu bestimmen, die vom Strahl getroffen wird. Dies wiederum ist in der Computergrafik, insbesondere im *ray tracing*, ein sehr detailliert untersuchtes Grundproblem. Wichtige Grundlagen und eine hocheffiziente Datenstruktur für dieses Kernproblem werden nun vorgestellt.

Anstatt also für jede Anfrage linear über das gesamte Dreiecksnetz zu iterieren, wird vorab für das Netz eine solche Suchdatenstruktur aufgebaut. Sie liefert bei Eingabe eines Strahls alle Dreiecke (oder alle Punkte), die vom Strahl auf dem Dreiecksnetz getroffen werden. Diese Ergebnisse können dann nachbearbeitet werden, um die konkrete Anfrage zu beantworten. Diese Suchdatenstrukturen sind also optimiert auf Antwortzeiten und nicht auf geringen Speicherplatz.

3.3.1 Flache und hierarchische Strukturen

Alle Suchdatenstrukturen enthalten als Kernidee die räumliche Unterteilung (engl. *spatial subdivision*) des zu durchsuchenden Gebiets in kleinere Teilgebiete. Das heuristische Ziel ist es dabei, nur noch wenige vergleichbar teure Schnitte zwischen Objekt und Strahl durchzuführen, weil eine kleinere Region weniger Objekte enthält. Stattdessen kommen heuristisch insgesamt billigere Zusatzkosten für die Entscheidung, ob ein Teilgebiet weiter untersucht werden muß, hinzu. Daher wird man die Teilgebiete möglichst einfach wählen.

Es werden zwei große Klassen von Verfahren unterschieden, **flache** und **hierarchische** Strukturen.

Flache Strukturen können in zwei Untergruppen unterteilt werden: Raumorientierte und objektorientierte Partitionen. Beispiele für raumorientierte Partitionen sind:

uniforme Gitter: Ein rechteckiges Teilgebiet der Ebene, das alle zu untersuchenden Objekte umfaßt, wird durch ein reguläres Gitter zerteilt. Beim Aufbau der Suchdatenstruktur wird für jede dieser Zellen gespeichert, ob und wenn ja welche Teilobjekte sie enthält. Der Durchlauf durch ein reguläres Gitter kann inkrementell (s. [19]) und damit effizient implementiert werden. Jede traversierte Zelle des Gitters wird getestet, dies ist sehr einfach möglich. Ist eine Zelle leer, muß sie nicht weiter betrachtet werden.

Voxelzerlegungen Dies ist die Analogie zu regulären Gittern in 3D. Aufbau und Traversierung lassen sich einfach übertragen.

Makrozerlegungen Hier werden rechteckige oder quaderförmige maximale Teilgebiete des Raums gesucht, die leer sind. Der Aufbau der Datenstruktur ist komplizierter, dafür wird heuristisch deutlich weniger Speicher benötigt.

Im Gegensatz dazu umhüllen objektorientierte Ansätze (dies hat nichts mit dem bekannten Programmierparadigma gleichen Namens zu tun) die gegebenen Objekte mit geometrisch einfacheren Objekten, beispielsweise Kreisen oder Quadern. Ansätze aus dieser Gruppe heißen daher auch Hüllvolumen-Strukturen. Variationsmöglichkeiten ergeben sich in der Ausrichtung der Hüllvolumina: von achsenparallel über objektparallel hin zu Hüllpolygonen. Hüllquader werden häufig mit dem englischen Begriff *bounding box* bezeichnet.

Hierarchische Strukturen werden unterteilt in uniforme und nichtuniforme Ansätze, je nach Struktur der Teilregionen. Uniforme Gitter bzw. Voxelmodelle sind einfach eine hierarchische Version ihrer flachen Namensgeber, sämtliche Algorithmen lassen sich – erweitert um die hierarchischen Auf- und Abstiege – übernehmen. Beispiele für nichtuniforme Suchstrukturen sind

binäre Raumunterteilungen (engl. *binary space partition*), k-D-Bäume und *Octrees*. Eine umfassende Übersicht über die Vor-und Nachteile dieser Strukturen sowie über einige praktische Aspekte bietet [12].

3.3.2 Octrees

Octrees sind die Suchstruktur, die die weiteste Verbreitung gefunden hat. Als hierarchische Struktur sind sie rekursiv definiert:

DEFINITION 3.8 (OCTREE)
Sei S eine Menge von Objekten im Raum, $Q = [x_{min}(Q), x_{max}(Q)] \times [y_{min}(Q), y_{max}(Q)] \times [z_{min}(Q), z_{max}(Q)]$ der Hüllquader von S.

1. *Ist $card(S) \leq m$ für einen vorher festgelegten Schwellwert (Lastfaktor) m, besteht der Octree aus einem einzigen Knoten (Blatt), in dem Referenzen auf alle Elemente in S sowie auf Q gespeichert sind.*
2. *Andernfalls besteht der Octree aus einem inneren Knoten mit acht Kindern, von denen jedes wieder ein Octree ist. Sei $(x_{med}, y_{med}, z_{med})$ der Mittelpunkt von Q. Der Hüllquader des inneren Knotens ist Q. Die Hüllquader der acht Kinder sind $Q_{ijk} = X_i \times Y_j \times Z_k$ mit $i \in \{LEFT, RIGHT\}$, $j \in \{DOWN, UP\}$, $k \in \{FRONT, BACK\}$ und*

$$\begin{array}{rclrcl} X_{LEFT} & = & [x_{min}, x_{med}] & X_{RIGHT} & = & [x_{med}, x_{max}] \\ Y_{DOWN} & = & [y_{min}, y_{med}] & Y_{UP} & = & [y_{med}, y_{max}] \\ Z_{FRONT} & = & [z_{min}, z_{med}] & Z_{BACK} & = & [z_{med}, z_{max}]. \end{array}$$

   In den Kindern werden Referenzen auf die Objekte gespeichert, die im jeweiligen Hüllquader Q_{ijk} enthalten sind.

In dieser klassischen Definition werden Referenzen auf Objekte also nur in Blättern des Baumes gespeichert. Ein rekursiver Algorithmus zur Konstruktion eines Octrees ausgehend von einer Menge S ergibt sich direkt aus der Definition.

Es gibt unzählige Varianten eines Octrees. Als Abbruchkriterium der Rekursion kann beispielsweise eine Minimalgröße der beteiligten Hüllquader oder eine maximale Baumtiefe dienen. Anstelle der Unterteilung der Hüllquader am räumlichen Mittelpunkt kann auch am Mittel der Objektmittelpunkte unterteilt werden, dies ist sinnvoll bei nichtuniformer Verteilung der Objekte. Die einzelnen Bereiche der Kinderknoten sind dann jedoch auch nicht mehr gleichförmig.

Traversierung eines Octrees, Suche Die Suche in einem Octree ist ebenfalls auf viele verschiedene Arten möglich. In der Literatur wird der Prozeß, alle von einem gerichteten Strahl durchquerten Teilquader in einem Octree zu finden, auch als *octree traversal* bezeichnet. Eine mögliche Klassifizierung ist die Einteilung in rekursive (*top down*) und nichtrekursive (*bottom up*) Strategien. Hier soll ein hocheffizienter parametrischer rekursiver Ansatz vorgestellt werden, der im Jahr 2000 vorgeschlagen wurde (s. [51]). Weil der Algorithmus in 2D (also beim Octree-Analogon *Quadtree*) einfacher zu verstehen ist, wird dieser Fall beschrieben. Die

Übertragung in die dritte Dimension ist elementar und kann dem zitierten Aufsatz entnommen werden.

Die Grundidee des Algorithmus besteht darin, den Strahl $\mathbf{r} = \mathbf{p} + t\mathbf{d}$ mit Parameter $t \geq 0$, Startpunkt $\mathbf{p}$ und normalisierter Richtung $\mathbf{d}$ (d.h. $||\mathbf{d}|| = 1$) zu parametrisieren:

$$\mathbf{r}(t) = \begin{pmatrix} x_r(t) \\ y_r(t) \end{pmatrix} = \begin{pmatrix} p_x + td_x \\ p_y + td_y \end{pmatrix} \tag{3.3}$$

Wie oben sei $Q = [x_{min}(Q), x_{max}(Q)] \times [y_{min}(Q), y_{max}(Q)]$ ein Hüllrechteck innerhalb des Quadtrees. Ein Schnitt zwischen Q und $\mathbf{r}$ tritt genau dann auf, wenn es mindestens ein t gibt, so daß

$$x_{min}(Q) \leq x_r(t) < x_{max}(Q) \text{ und } y_{min}(Q) \leq y_r(t) < y_{max}(Q) \tag{3.4}$$

Mit $t_{x,min}(Q), t_{x,max}(Q), t_{y,min}(Q), t_{y,max}(Q)$ werden die Parameterwerte bezeichnet, für die der Strahl $\mathbf{r}$ das Hüllrechteck Q betritt bzw. verläßt. Sie lassen sich wie folgt berechnen:

$$\begin{array}{llll} t_{x,min}(Q) = (x_{min}(Q) - p_x)/d_x & & t_{x,max}(Q) = (x_{max}(Q) - p_x)/d_x \\ t_{y,min}(Q) = (y_{min}(Q) - p_y)/d_y & & t_{y,max}(Q) = (y_{max}(Q) - p_y)/d_y \end{array} \tag{3.5}$$

Diese Werte werden zu Beginn des Algorithmus für den Wurzelknoten des Quadtrees berechnet und danach inkrementell aktualisiert. Bei klassischer Unterteilung am Mittelpunkt des umhüllenden Rechtecks gilt gerade für $j \in \{UP, DOWN\}$:

$$\begin{array}{rcll} & & t_{x,min}(Q_{LEFT,j}) = t_{x,min}(Q) \\ t_{x,max}(Q_{LEFT,j}) & = & t_{x,min}(Q_{RIGHT,j}) = (t_{x,max}(Q) - t_{x,min}(Q))/2 \\ & & t_{x,max}(Q_{RIGHT,j}) = t_{x,max}(Q) \end{array} \tag{3.6}$$

Die Beziehungen für die y-Koordinate ergeben sich analog.

Mit Hilfe all dieser Beziehungen kann nun die Ausgangsgleichung 3.4 parametrisiert umgeschrieben werden:

$$\mathbf{r} \cap Q \neq \emptyset \Leftrightarrow \exists\, t \geq 0 \text{ mit } t_{x,min}(Q) \leq t < t_{x,max}(Q) \text{ und } t_{y,min}(Q) \leq t < t_{y,max}(Q) \tag{3.7}$$

Mit der vereinfachenden Schreibweise

$$t_{min}(Q) = \max(t_{x,min}(Q), t_{y,min}(Q)) \quad t_{max}(Q) = \min(t_{x,max}(Q), t_{y,max}(Q))$$

ist Gleichung 3.7 äquivalent zu

$$\mathbf{r} \cap Q \neq \emptyset \Leftrightarrow t_{min}(Q) < t_{max}(Q) \tag{3.8}$$

Wenn diese Bedingung erfüllt ist, korrespondieren alle Parameterwerte t aus dem Interval $[t_{min}(Q), t_{max}(Q)[$ mit Punkten $(x_r(t), y_r(t))$, die im Inneren von Q liegen, und umgekehrt. Ist die Bedingung nicht erfüllt, kann es keinen Schnitt geben.

Nach diesen Vorarbeiten kann der Algorithmus zusammengefaßt werden:

1. Berechne die vier Werte $t_{x,min}(Q), t_{x,max}(Q), t_{y,min}(Q), t_{y,max}(Q)$ für die Wurzel des Baumes.
2. Werte die Bedingung 3.8 für die Wurzel aus, ist sie nicht erfüllt, schneidet der Strahl keines der Objekte im Quadtree, der Algorithmus gibt also die leere Menge zurück.
3. Werte rekursiv die vier Kinderknoten der Wurzel mit inkrementell berechneten Parameterwerten aus:
4. Ist ein so nicht verworfener Knoten Blatt, so füge die dort referenzierten Objekte der Ergebnisliste hinzu.
5. Ist ein so nicht verworfener Knoten innerer Knoten, so werte ihn rekursiv aus.

Algorithmus 3.2: Parametrischer Durchlauf durch einen Quadtree

Der nächste Schritt besteht darin, die Zellen zu bestimmen, die von einem Strahl durchquert werden. Dies erfolgt in zwei Schritten:

1. Bestimme den ersten Teilknoten, der vom Strahl getroffen wird.

2. Für jeden getroffenen Knoten, wähle den nächsten Knoten, solange bis der Parameterbereich des aktuelle Elternknoten (die Wurzel des aktuellen Teilbaumes) verlassen wird.

Der erste Schritt kann wie folgt gelöst werden: Zunächst wird die „Eingangskante" für das aktuelle Rechteck bestimmt. Dies kann sehr elegant durch Vergleiche der Werte $t_{x,min}(Q)$ und $t_{y,min}(Q)$ geschehen, wie in den folgenden Abbildungen skizziert. Jetzt sind noch zwei Kinder potentielle Kandidaten, ob eines oder zwei weiter untersucht werden müssen, wird durch Vergleiche dieser Werte mit den Parameterwerten des Mittelpunkts des aktuellen Elternknoten entschieden. Die Skizzen 3.3 und 3.4 verdeutlichen das Vorgehen. Der gesamte Entscheidungsprozeß kann sehr effizient in einen Bitvektor der Länge 4 (bzw. 8 beim Octree) codiert werden, um die rekursive Weiterauswertung zu steuern.

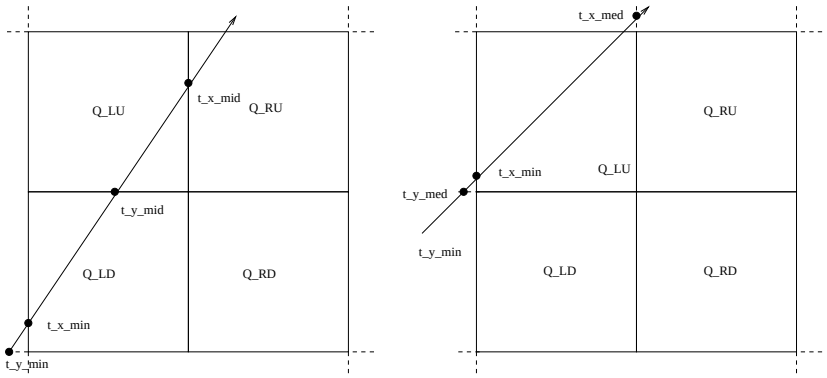


Abbildung 3.3: *Traversierte Kindknoten im Fall* $t_{x,min}(Q) > t_{y,min}(Q)$

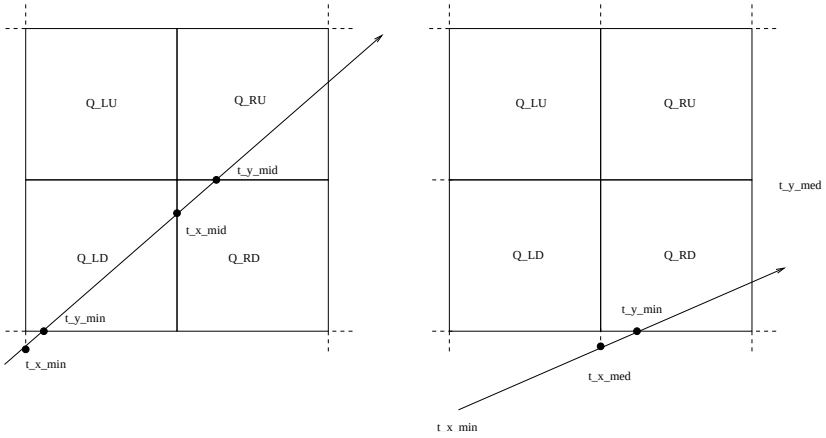


Abbildung 3.4: *Traversierte Kindknoten im Fall* $t_{y,min}(Q) > t_{x,min}(Q)$

Der zweite Schritt, die Bestimmung des nächsten Knotens, kann ähnlich angegangen werden, nur daß die „Austrittskanten" gesucht sind und entweder ein weiterer Kindknoten untersucht werden muß oder der Parameterbereich des Elternknotens verlassen wird. In [51] ist hierzu ein endlicher Automat angegeben, der alle auftretenden Fälle beinhaltet und in einer Implementierung nur in eine große Fallunterscheidung übersetzt werden muß.

Die Implementierung dieses Algorithmus ist einfach, und er zählt zu den schnellsten Suchalgorithmen für allgemeine, klassische Octrees. Probleme hinsichtlich der numerischen Stabilität und dem Auftreten von Rundungsfehlern sind nicht zu erwarten, weil der gesamte inkrementelle Aktualisierungsprozeß durch Divisionen erfolgt, die im Gegensatz zu Additionen numerisch stabil sind.

3.3.3 Probleme in der praktischen Umsetzung

Ein Hauptproblem der klassischen Definition tritt auf, wenn beim Unterteilen eines Hüllquaders Objekte am Rand eines oder mehrerer Kindknoten „überstehen", d.h. nicht komplett in einen der neuen Hüllquader passen. Hierzu wurde in dieser Arbeit mit fünf verschiedenen heuristischen Lösungsansätzen experimentiert:

Abbruch Breche die rekursive Unterteilung ab, wenn ein Objekt nicht in den Wurzelknoten des aktuellen Teilbaums paßt. Diese Technik ist offensichtlich zu unperformant, da sich schnell Beispiele konstruieren lassen, für die schon die Wurzel nicht mehr unterteilt wird.

geordnete Einfachreferenzierung Das nicht passende Objekt wird im ersten Kind gespeichert, in dessen Hüllquader es teilweise paßt. Diese Technik ist nicht mit dem effizienten parametrisierten Suchalgorithmus kombinierbar, da der passende Kindknoten bei „ungeeigneten" Strahlrichtungen nicht traversiert wird.

Mehrfachreferenzierung mittels überlappender Hüllquader Hier wird das Objekt in allen Kindknoten gespeichert, in die es teilweise paßt. Die Hüllquader der Kindknoten werden angepaßt, so daß sie weiterhin alle repräsentierten Objekte enthalten. So überlappen

sich benachbarte Hüllquader. Die Invariante des klassischen Octrees, daß ein Wurzelquader eines Teilbaums alle Kindquader enthält und diese in ihrer Vereinigung genau den Wurzelquader ergeben, bleibt erhalten. Die inkrementelle Aktualisierung der Such-Parameterwerte bleibt erhalten, die angegebenen Methoden zum Auffinden des ersten und nächsten traversierten Teilknotens müssen jedoch aktualisiert werden, da die Vereinigung der Kindquader nicht mehr disjunkt ist. Darunter leidet die Effizienz des Verfahrens. Die Implementierung ist recht komplex, da im *worst case* alle 8 Teilquader getestet werden müssen.

innere Referenzierung Die Definition des klassischen Octrees wird dahingehend relaxiert, daß auch in inneren Knoten Objektreferenzen gespeichert werden dürfen. Nicht passende Objekte werden so nicht in Kindern, sondern direkt im aktuellen Wurzelknoten abgelegt. Die Invariante bleibt gültig. Bei der Traversierung wird neben dem rekursiven Aufruf eine lineare Suche auf den lokal gespeicherten Objekten durchgeführt. Um die Effizienz zu erhöhen, können diese Objekte noch mit einer objektorientierten (eine raumorientierte wird gerade scheitern) flachen Suchheuristik (s.o.) umhüllt werden.

Zerteilung Die unpassenden Objekte werden zerteilt, so daß ihre einzelnen Bestandteile klassisch veiterverarbeitet werden können. Dieses Verfahren läßt die Aufbauzeit für den Octree bei „schlechten" Modellen explodieren, und die Tiefe (und damit die zu erwartende Antwortzeit des Suchalgorithmus) steigt schnell an. Analytisch ist dies jedoch die einzige Möglichkeit, die klassische Definition einzuhalten.

Experimente zeigen, daß die Zerteilungsvariante bei den verwendeten Testobjekten nicht mit der inneren Referenzierung konkurrieren kann. Eine detailiertere Betrachtung ergibt schnell den Grund: Oberflächentriangulierungen praktisch relevanter Geometrien zeichnen sich durch stark variierende Dreiecksgrößen aus: wenige große Dreiecke genügen zur Modellierung ebener Teilflächen, während die Approximation stark gekrümmter Bereiche viele kleine Dreiecke impliziert. Als Beispiel betrachte man die Oberflächentriangulierung des Automodells aus Abbildung 3.7, und hier insbesondere die Bereiche um die Seitentüren und die Frontpartie. Obwohl hier schon in der ersten Unterteilungsebene (an der Wurzel des Gesamtbaums) bei Verwendung der inneren Referenzierung lokal Daten gespeichert werden müssen, sind die Antwortzeiten besser. Im Zerlegungsalgorithmus werden diese großen Dreiecke einfach zu häufig in zu viele kleinere Teildreiecke zerlegt.

3.4 Projektionstechniken

Dieser Abschnitt stellt verschiedene Projektionstechniken vor. Die Darstellung bezieht sich bereits auf das angestrebte Szenario innerhalb einer Finite-Element-Diskretisierung auf Hexaederbasis (s. Kap. 2.2). Das Kernproblem, das zu lösen ist, lautet:

DEFINITION 3.9 (PROJEKTIONSAUFGABE)
Sei $\mathbf{p} \in \mathbb{R}^3$ *ein Punkt im Raum,* $S \subset \mathbb{R}^3$ *eine Oberflächentriangulierung und* $\mathbf{r} \in \mathbb{R}^3$ *eine Projektionsrichtung. Gesucht ist ein Punkt* $\mathbf{q}$ *auf* S*, der von* $\mathbf{p}$ *ausgehend entlang* $\mathbf{r}$ *erreicht wird. Dieser Punkt heißt* ***Projektion*** *von* $\mathbf{p}$ *auf* S*.*

Man beachte die allgemeine Formulierung dieser Definition.

3.4.1 Projektion mit kürzesten Abständen

Diese Projektionstechnik ist die „klassische Lösung". Es wird unter allen Punkten auf der Oberfläche der Lotfußpunkt gesucht. Dies wird gerade durch Algorithmus 3.1 gelöst. Weil die Projektionsrichtung für jedes Dreieck neu berechnet werden muß (sie verläuft gerade entlang der Normale des Dreiecks), ist dieses Verfahren mit dem im vorherigen Kapitel vorgestellten *raytrace*-Octree nicht zu beschleunigen. Allerdings eignen sich viele anderen Suchstrukturen zur Beschleunigung. Ein Beispiel ist eine Voxelzerlegung des Gebiets. Mittels einfacher Koordinatenvergleiche wird zunächst die Zelle bestimmt, in der der zu projizierende Punkt liegt. Alle Nachbarzellen werden dann solange mit zunehmendem Abstand durchsucht, bis eine nichtleere Zelle und damit Kandidaten für den kürzesten Abstand gefunden sind. Nach Ende der Projektion wird eine Glättung über alle inneren Knoten durchgeführt (s. Kap. 2.3), um die Gitterqualität lokal und global zu verbessern.

Der Hauptnachteil bei der Verwendung kürzester Abstände ist, wie oben begründet, ihre Uneindeutigkeit. Bei der Projektion neu entstandender Punkte auf die Oberfläche im Kontext der Hexaeder-Verfeinerung können so beliebig verformte Hexaeder entstehen. Abbildung 3.5 zeigt ein Beispiel.

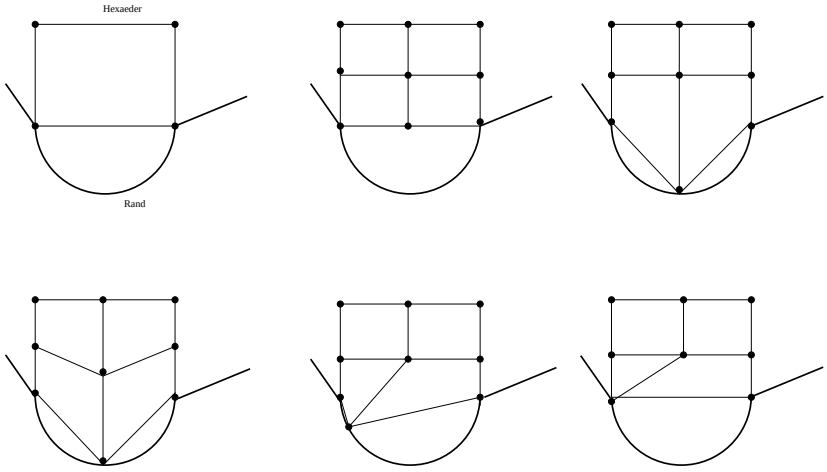


Abbildung 3.5: *Beispiel für Projektionen mit kürzesten Abständen im Subdivisionsschritt: Obere Reihe: Ausgangssituation vor der Unterteilung, Unterteilung ohne Projektion, korrekte Projektion ohne Glättung. Untere Reihe: korrekte Projektion mit lokaler Glättung, suboptimale Projektion mit verzerrten Zellen, worst case: zusammenfallende Knoten*

Dieses Problem ist nicht trivial lösbar. Einige Heuristiken versprechen aber in der Praxis gute Ergebnisse:

- Das Grobgitter wird als optimal an den Rand angepaßt vorausgesetzt, d.h. alle „Unebenheiten" in der Geometrie müssen bereits hinreichend fein durch die Randflächen

der Randhexaeder abgedeckt sein. Dies ist aus zwei Gründen unbefriedigend: Erstens verschiebt es das Problem auf die Generierung des Grobgitters, und zweitens enthalten Grobgitter dieser Form häufig zu viele Hexaeder, um einen effizienten Mehrgitter-Lösungsprozeß durchführen zu können: Die Berechnungen auf dem gröbsten Level dominieren die Laufzeit.

- Es wird nicht ein Lotfußpunkt gesucht, sondern alle. Analog zu den „smart Laplace"-Techniken aus Kapitel 2.3 wird derjenige ausgewählt, der die beste lokale Gitterqualität impliziert. Eine mögliche Erweiterung wäre die Gewichtung dieser Kriterien mit der Inversen des Abstandes, um die Lokalität der Projektion sicherzustellen.

3.4.2 Gewichtete Projektion

Diese Technik ist speziell an die Randanpassung während der Gitterverfeinerung angepaßt. Ausgangspunkt ist die folgende Beobachtung: Wenn die Eckpunkte des unverfeinerten Hexaeders bereits auf dem Rand liegen und das Hexaeder qualitativ gut ist, so kann erwartet werden, daß bei Positionierung des neu entstandenen Punkts durch Gewichtung der Positionen einiger Ausgangspunkte wieder eine gute Gitterqualität erzielt wird. Für die konkrete Beschreibung muß unterschieden werden zwischen neuen Knoten, die durch Bisektion der Randkanten des Ausgangshexaeders entstehen, und solchen, die aus den Mittelpunkten einer Rand-Seitenfläche entstehen.

Im Detail ergibt sich folgender Algorithmus:

Für einen neuen Randpunkt:

1. berechne die Seitenflächen, die für die Projektion benötigt werden:
   - für Knoten, die als Mittelpunkt einer Ausgangs-Seitenfläche entstanden sind, ist dies nur die Seitenfläche selbst
   - für Knoten, die als Kantenmittelpunkte entstanden sind, sind dies alle alten Seitenflächen, die die unterteilte Kante enthalten
2. berechne die Projektionsrichtung:
   - approximiere die Normalen aller eben berechneten Flächen als gewichtete Summe der einzelnen Normalen in den Eckpunkten der Fläche (Kreuzprodukt der beiden Kanten)
   - bestimme die Projektionsrichtung als gewichtete Summe all dieser Normalen
   - stelle sicher, daß diese Normale äußere Normale des Hexaeders ist: suche eine Kante des Hexaeders, die nicht Teil des Faces ist, und berechne den Winkel zwischen dieser Kante und der Normale. Ist er kleiner $\pi/2$, so invertiere die Normale

Algorithmus 3.3: Berechnung der Projektionsrichtung bei gewichteter Projektion

Eine Variationsmöglichkeit ist es, den Abstand der zur Normalenberechnung einbezogenen Punkte in die Gewichtung einfließen zu lassen und Eckpunkten, die näher am neu entstandenen Punkt liegen, ein höheres Gewicht zu geben.

Dieses Vorgehen wird folgendermaßen in den Unterteilungsprozeß integriert, um effiziente Suchdatenstrukturen nutzen zu können:

Iteriere über alle Hexaeder und führe einen Unterteilungsschritt durch.
Iteriere über alle neu entstandenen Punkte: Für jeden dieser Punkte:

1. entscheide, ob er auf den Rand projiziert werden muß
2. berechne den Zielrand
3. berechne die Projektionsrichtung
4. berechne den Auftreffpunkt der Projektion auf dem Rand

Iteriere nochmals über alle Punkte und ändere die Koordianten entsprechend.

Algorithmus 3.4: Kompletter Unterteilungsprozeß mit gewichteter Projektion

Die Änderung der Koordinaten im letzten Schritt muß offenbar Jacobi-artig erst nach der Berechnung aller Projektionsrichtungen durchgeführt werden, andernfalls sind die Projektionsrichtungen nicht korrekt, weil das Verschieben eines Punktes die Normalen benachbarter Seitenflächen verändert. Die Berechnung der neuen Koordinaten kann effizient über den *ray tracing* Octree erfolgen, da sowohl der Ursprung des Strahls (die unprojizierten Koordinaten des Knotens) als auch die Richtung (die Projektionsrichtung) für einen Punkt feststehen, und zwar unabhängig von der Oberflächenbeschreibung. Es ergibt sich nichts anderes als das klassische Schnittproblem zwischen einem Strahl und einer Oberflächentriangulierung.

An die Projektion kann sich eine Glättung der inneren Knoten anschließen.

Diese Technik vermeidet elegant die Uneindeutigkeit der Projektion mit kürzesten Abständen (vgl. Abb. 3.5). Ihr schwerwiegender Nachteil besteht darin, daß die Schnittberechnung nicht notwendig eine Lösung liefern muß: Der Strahl kann „am Objekt vorbeischießen". Analog können sich auch Strahlen kreuzen und damit zu invertierten Hexaedern führen. Ein Beispiel zeigt Abbildung 3.6.

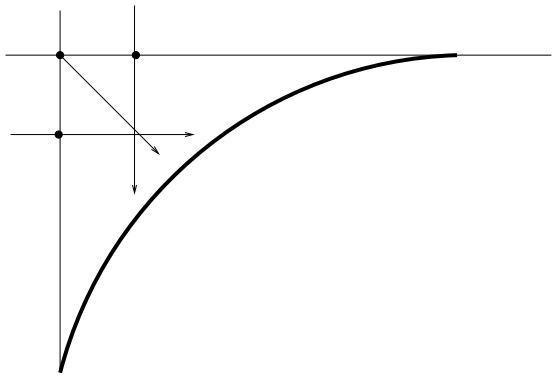


Abbildung 3.6: *Beispiel für Fehler bei gewichteter Projektion*

Als Ausweg wird, wenn ein solcher Fall festgestellt wird, für den gerade betrachteten Knoten auf die Projektion mittels kürzester Abstände umgeschaltet. Als Indikatoren für das Auftreten eines Projektionsfehlers können wieder alle Qualitätsmerkmale dienen (s. Kap. 4.4).

3.4.3 Umbrella-Projektion

Beide oben vorgestellte Projektionsalgorithmen sind für praktisch relevante, komplexe Geometrien nicht robust genug: Die Projektion mit kürzesten Abständen ist nicht eindeutig und kann zu invertierten Elementen führen, die gewichtete Projektion muß nicht notwendig eine Lösung besitzen. Die Grundidee für den eigenen Projektionsalgorithmus basiert auf aktiven Konturen (engl. *active contours, snakes*) [13, 34, 44, 47, 55, 56]. Aktive Konturen wurden zur Extraktion von relevanten Bestandteilen (Segmentierung) aus zwei- oder dreidimensionalen Bilddaten vorgeschlagen. Eine typische Anwendung (s. obige Referenzen) ist die Extraktion einer Arterie aus einem Voxelmodell, das von einem Computertomographen geliefert wird.

Anschaulich kann eine aktive Kontur aufgefaßt werden als der Prozeß, um ein gegebenes Objekt eine Plastikmembran zu legen und diese dann zu evakuieren (engl. *shrink wrapping*). Im Kontext der Randanpassung einer Finite-Elemente-Diskretisierung ist das Analogon zur Plastikmembran die Menge der Hexaeder-Seitenflächen, die mit ihren vier Knoten auf dem geometrischen Randobjekt zu liegen kommen sollen. Wie bei einer realen Plastikmembran ist es hier wichtig, daß keine Risse, Überlagerungen oder „Falten" in der Kontur entstehen. In der für diese Arbeit relevanten Form eignet sich das Verfahren nur für Objekte mit Genus 0, d.h. für Objekte ohne „Löcher".

Diese Idee läßt sich in das folgende zweidimensionale Modell übersetzen (die Übertragung auf den dreidimensionalen Fall kann [54] entnommen werden): Eine aktive Kontur ist eine Funktion $\nu : [0; 1] \rightarrow \mathbb{R}^2$, die auf einem „Bild" $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ positioniert wird und sich durch Minimierung ihrer Energie in eine optimale Position bewegen soll. Die Energie dieser Kontur setzt sich aus zwei Bestandteilen zusammen, der Bildenergie und der internen Energie:

$$E(\nu, f) = E_{image}(\nu, f) + E_{int}(\nu)$$

Die interne Energie ist dabei definiert als

$$E_{int}(\nu) = \int \alpha(s)|\nu'(s)|^2 + \beta(s)|\nu''(s)|^2 ds$$

Der erste Term dieser Gleichung, genannt Membran-Energie, nimmt große Werte an, wenn große Abstände zwischen benachbarten Punkten auf der Kontur vorliegen. Der zweite Term beschreibt die Biegeenergie der Kontur. Man vergleiche hiermit auch die Herleitung des Umbrella-Operators (s. Kap. 2.3.1). Die Gewichtsfunktionen α und β beschreiben die Elastizität (engl. *elasticity*) und Steifheit (engl. *rigidity*). Als Bildenergie kann beispielsweise ein Filter zur Extraktion von Kanten oder ähnliches verwendet werden. Die Minimierung dieser Gesamtenergie führt auf Euler-Lagrange-Gleichungen, die beispielsweise mit der Methode der finiten Differenzen (s. obige Referenzen) gelöst werden können.

Im Kontext der Randanpassung in Finite-Elemente-Diskretisierungen wird die Bildenergie ersetzt durch eine Geometrie-Energie, die typischerweise um so größer ist, je weiter ein Punkt von der Geometrie entfernt ist, und auf dem Rand des Objekts den Wert Null annimmt. Ein denkbares Beispiel für eine konkrete Formulierung dieser Energie ist die Funktion, die jedem Punkt seinen Abstand zur Geometrie zuordnet.

Diese Methode ist ansprechend, erfolgsversprechend, aber leider viel zu aufwendig. Hinzu kommt, daß zur Voxelzerlegung eines Finite-Elemente-Gitters (klassifizierend in „im Inneren des Gebiets", „außerhalb des Gebiets" und „auf dem Rand") die Auflösung viel zu grob ist. Daher wurde in dieser Arbeit (in Analogie zu [41]) der folgende Ausweg gewählt: Die Geometrie-Energie wird ersetzt durch einen Projektionsoperator, der einen gegebenen Punkt direkt auf den Rand bewegt. Hier empfielt sich die Verwendung der gewichteten Projektion, weil heuristisch das Hexaedergitter schon nach wenigen Schritten gut mit dem zu approximierenden Objekt übereinstimmt. Der in Kapitel 3.3.2 vorgestellte effiziente Octree eignet sich zur Beschleunigung dieses Verfahrens. Die interne Energie wird modelliert durch den Umbrella-Operator (s. Kap. 2.3.1):

$$\mathbf{p}_{neu} := (1-\alpha)\mathbf{p} + \frac{\alpha}{\sum d_j} \sum_{\mathbf{q}_j \in Adj(\mathbf{p})} d_j \mathbf{q_j}$$

Man erhält zwei verschiedene Algorithmen, je nach Verwendung aller Nachbarpunkte oder nur der Randpunkte als Träger. Das im nächsten Abschnitt präsentierte Beispiel zeigt, daß beide Algorithmen praktisch relevant sind.

Insgesamt erhält man den folgenden Projektionsalgorithmus:

1. Setze $\alpha = 0,5$.
2. Solange kein Abbruchkriterium greift:
   - Führe einen Projektionsschritt durch.
   - Führe einen Relaxationsschritt mit dem Umbrella-Operator durch.
   - Verringere α um einen kleinen Betrag.
3. Stelle durch eine abschließende Projektion sicher, daß sich alle Knoten auf dem Rand befi nden.

Algorithmus 3.5: Umbrella-Projektion

Als Abbruchkriterium kann das Unterschreiten einer Toleranzgrenze zwischen alter und neuer Position nach einer Iteration oder das Überschreiten einer maximalen Anzahl Iterationen gewählt werden. Zusammengefaßt ist dieser Algorithmus also eine Kombination aus einem klassischen Projektionsalgorithmus und einer Randglättung zur Relaxation.

Dieser Algorithmus läßt sich gut mit der Grundidee des Mehrgitteralgorithmus vereinbaren, möglichst viel Arbeit auf niedrigen Hierarchieebenen zu verrichten. Ein Gitter, daß auf niedrigen Leveln mit dieser Projektionstechnik bereits gut an den Rand angepaßt ist, benötigt auf einem feinen Level (auf dem nur die neu hinzugekommenen Punkte auch projiziert werden müssen) heuristisch entweder nur sehr wenige Iterationen, oder es ist bereits so gut, daß die gewichtete Projektion keine Fehler verursacht.

Dieser Algorithmus ist deutlich robuster als die beiden anderen vorgestellten Projektionstechniken. Nach wie vor kann er in ungünstigen Fällen invertierte Elemente erzeugen. Es bietet sich folgende Heuristik zur Verbesserung der Stabilität an: Zur Detektierung lokaler Fehler kann nach der Projektion beispielsweise das Jacobi-Verhältnis (s. Kap. 4.4.6) in den projizierten

Punkten verwendet werden. In solchen Punkten wird die Projektion lokal zurückgenommen, d.h. der Knoten wird auf seine Position vor der Projektion verschoben. Seine Nachbarknoten implizieren bereits lokal akzeptable Elementqualität, also wird der Knoten reprojiziert mit der Technik der gewichteten Projektion. Dies liefert eine gute Heuristik, um die Anzahl der Projektionsfehler zu minimieren.

Die Wahl der Parameter, insbesondere, in welchem Umfang der Gewichtungsfaktor α verringert wird und wie viele Iterationen durchgeführt werden, ist problemabhängig und somit vom Anwender festzulegen.

Eine Glättung der inneren Knoten kann und sollte der Projektion folgen.

3.4.4 Beispiele

Die folgenden Beispiele stammen aus einem parallel zur Arbeit implementierten Gittereditor. Dargestellt ist eine Sequenz von Teilschritten unter Verwendung obiger robuster Projektionstechnik. Gerendert werden aus Gründen der Übersichtlichkeit nur die Seitenflächen der Hexaeder, die auf dem zu umströmenden Objekt liegen.

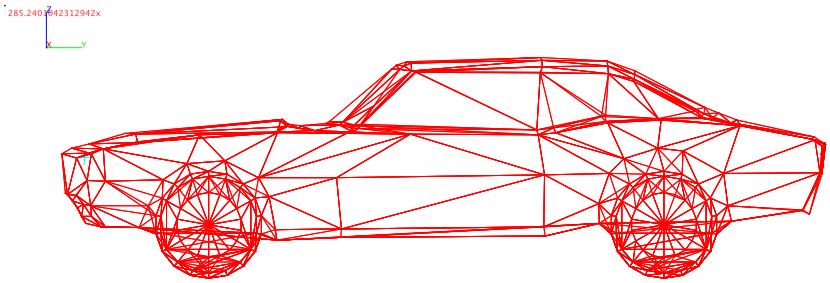


Abbildung 3.7: *Das zu umströmende Objekt als Oberflächentriangulierung*

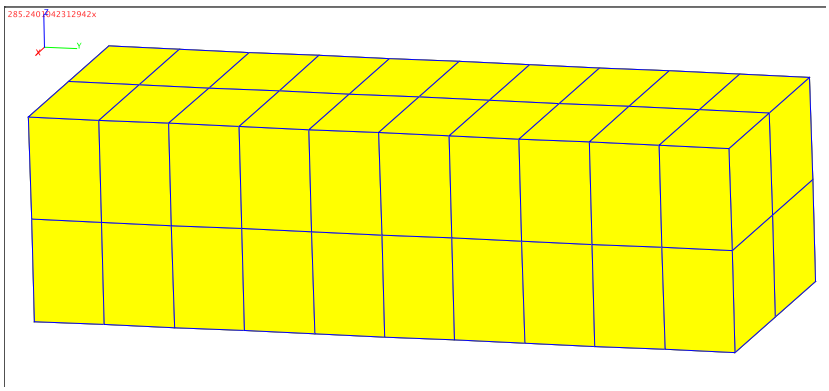


Abbildung 3.8: *Nach der Gittergenerierung: Seitenflächen der Hexaeder vor der Projektion*

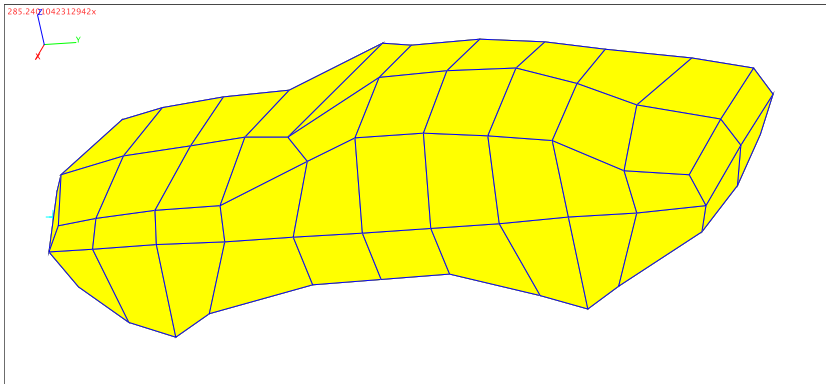


Abbildung 3.9: *Projektion auf die Oberflächen mit kürzesten Abständen. Man beachte die schlechte Elementqualität in der Umgebung der „Frontscheibe" des Modells.*

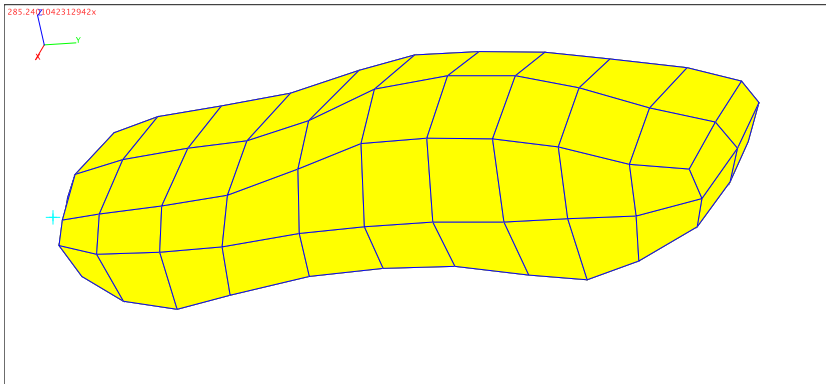


Abbildung 3.10: *Einmalige Anwendung des Umbrella-Operators mit einem Gewichtsfaktor von* $\alpha = 0.5$ *und Verwendung der Randknoten als Träger. Man beachte: Schon erfaßte Details der Geometrie („Räder") gehen teilweise verloren. Dafür wird die globale Elementqualität (insbesondere auf der „Frontscheibe") deutlich verbessert. Wichtig: Die dargestellten Flächen fallen jetzt nicht mehr mit der Oberflächentriangulierung zusammen, sondern liegen nach Konstruktion des angewendeten Operators im Inneren des Fahrzeugvolumens.*

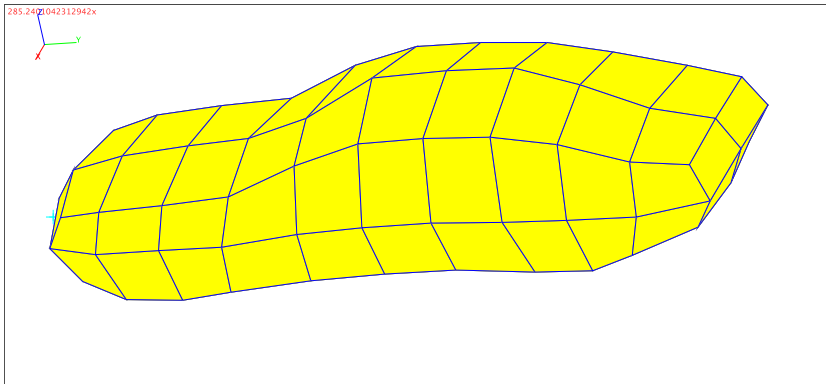


Abbildung 3.11: *Ergebnis der Reprojektion mit kürzesten Abständen. Die Elementqualität ist im Vergleich zu Abbildung 3.9 deutlich besser, Details der Geometrie werden im Vergleich zum Ergebnis nach der ersten Projektion nicht neu erfaßt, weil ein kürzerer Abstand statt auf den „Rädern" auf dem "Bodenblech" angenommen wird.*

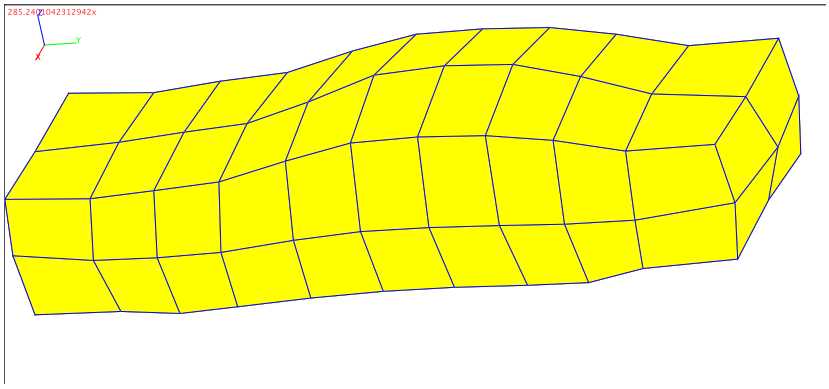


Abbildung 3.12: *Anwendung des Umbrella-Operators mit vollständiger Nachbarschaftsdefinition und Gewicht $\alpha = 0.5$. Die Elementqualität wird verbessert, neue Details („Räder") werden implizit durch das vergrößerte Volumen erfaßt. Wichtig: Die Seitenflächen liegen wieder nicht auf dem Objekt, sondern befinden sich nach Konstruktion des angewendeten Operators außerhalb des Fahrzeugvolumens.*

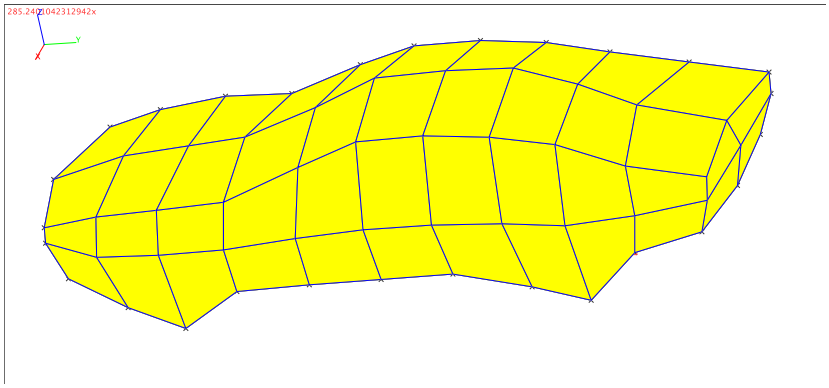


Abbildung 3.13: *Die abschließende Reprojektion (wieder mit kürzesten Abständen) liefert ein gutes Gitter um das zu umströmende Modell.*

3.5 Reparatur von Dreiecksnetzen

3.5.1 Oberflächengenerierung

In der Praxis werden hauptsächlich zwei Möglichkeiten genutzt, um ein zu umströmendes Objekt im Computer zu repräsentieren. Zum einen werden Oberflächenbeschreibungen in CAD-Tools erstellt (Beispiele sind AutoCAD [3] oder die freie Software Blender3D [7]). Zum anderen können sie mit 3D-Scannern aus einem realen Modell gewonnen werden. Dies ist ein hochgradig nichttriviales Problem, weil 3D-Scanner nur Punktwolken liefern, aus denen dann die Oberfläche rekonstruiert werden muß. Ein Übersichtsartikel zu diesem Thema ist [5]. Beide Möglichkeiten liefern allgemein keine perfekten Oberflächenrepräsentationen.

3.5.2 Oberflächenreparatur

Ein Problem in der Oberflächendarstellung ist es, daß die entstehenden Modelle oft nicht „wasserdicht" sind, d.h. daß sie kein geschlossenes Volumen umranden: Wenn eine Oberfläche in einem CAD-Programm geschlossen „aussieht", muß sie noch lange nicht geschlossen sein. Konkret treten beispielsweise Risse, Degenerationen, Verdoppelungen, Löcher und Überlappungen auf. Die Gründe sind vielfältig: ungenaue Arithmetik, Seiteneffekte der Modelltransformationen (beispielsweise bei der Volumengenerierung durch Extrusion oder Spiegelung), Fehler des Designers oder der Software. Diese Fehler behindern die weitere Verwendung der Oberflächen, beispielsweise kann bei der Projektion (s. Kap. 3.4) der Projektionsstrahl durch ein Loch in der Oberfläche verläuft und somit ein falscher Schnittpunkt, d.h. als Folge eine invertierte Hexaederzelle berechnet wird.

Isolierte und doppelte Ecken Ecken, die nicht zu einem Dreieck der Oberflächentriangulierung gehören, können ersatzlos gestrichen werden. Analog können auch isolierte Kanten verworfen werden. Knoten, die nur einen Abstand kleiner als ein vom Benutzer vorgegebener Schwellwert voneinander haben, werden zu einem Knoten verschmolzen. Entstehen dabei Kanten der Länge bzw. Dreiecke des Volumens null, werden sie gelöscht.

Schließen von Löchern: Einfacher Ansatz Falls die Oberflächentriangulierung keine Verzweigungen bzw. Versatzstücke enthält, d.h. falls jede Kante im Schnitt von maximal zwei Dreiecken liegt, so ist eine Kante genau dann Teil der Umrandung eines Lochs, wenn sie nur in einem Dreieck enthalten ist. Stehen Nachbarschaftsinformationen oder hierarchische Eltern-Kind-Beziehungen zur Verfügung, genügt eine Iteration über das Netz, um eine Liste dieser Kanten zu erstellen. Mittels eines *greedy*-Ansatzes über die Adjazenzinformationen können diese Kanten dann zu geschlossenen Polygonzügen verbunden werden. Das Ergebnis ist eine Menge von Polygonzügen, die jeweils ein Loch in der Triangulierung beranden, sowie eine Menge an Kanten, die nicht zugeordnet werden konnten. Letztere werden gelöscht. Jedes Loch kann nun elementar durch eine Triangulierung geschlossen werden, beispielsweise, indem der Mittelpunkt des Lochs berechnet wird und neue Dreiecke zwischen jeweils einer Randkante und dem Mittelpunkt eingefügt werden.

Reparatur durch Kantenvereinigung Der Triangulierungsansatz liefert zufriedenstellende Ergebnisse bei relativ großen Löchern. Kleine Risse bzw. Unstetigkeiten im Dreiecksnetz sollten jedoch nicht mit dieser Methode geschlossen werden, weil die Zahl der erzeugten Dreiecke zu groß und deren Qualität schon in einfachen Fällen zu schlecht ist. Ein Ausweg ist der Reparaturalgorithmus von Barequet und Kumar (s. [4]): Der Algorithmus berechnet zunächst die sogenannten *connected components*, d.h. die Teilmengen von Dreiecken, die untereinander verbunden sind. Für jede Randkante dieser Mengen werden nun die Randkanten der anderen Mengen berechnet, die geeignete Kandidaten für das dann folgende Verschmelzen von Kanten durch Identifikation ihrer Ecken sind. Der Algorithmus erkennt bei geschickt vorgegebenen Schwellwerten, ob es sich bei einer Randkante um einen Riß (der geschlossen werden kann) oder ein großes Loch (das trianguliert wird) handelt.

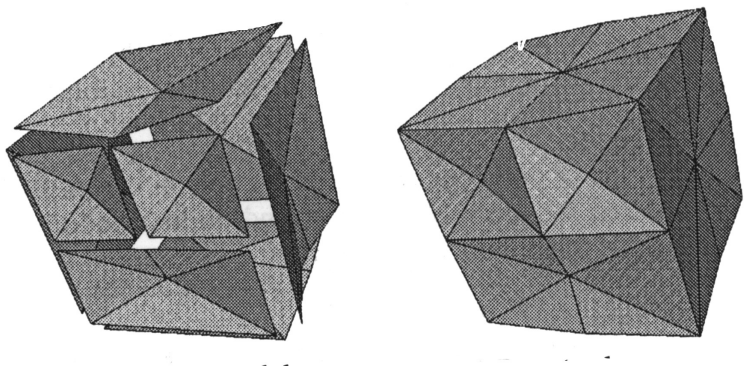


Abbildung 3.14: *Beispiel für die Kantenvereinigung nach [4]*

Durch geschickte Implementierung erkennt der Algorithmus auch Verdoppelungen und Überlappungen. Mittels eines Adjazenzgraphen bzw. einer Tiefensuche auf dem Netz wird verhindert, daß die Berechnung der *connected components* quadratische Zeit in Anspruch nimmt. Ein intelligentes *score*-System sorgt dafür, daß nur passende Kanten vereinigt werden und Löcher von Rissen unterschieden werden.

Visuelle Reparatur Die wichtigste Technik bleibt nach wie vor die visuelle Analyse durch den Anwender. Die obigen Verfahren dienen dabei zur Vorbereitung, um dem Benutzer mögliche Fehler und Reparaturvorschläge aufbereitet präsentieren zu können. Viele Fehler sind auch besser wahrnehmbar, wenn im Modellierungssystem nicht die Anzeige als Drahtmodell gewählt wird, sondern ein Beleuchtungsverfahren genutzt wird.

Kapitel 4

Gittergenerierung

4.1 Einleitung und Übersicht

Dieses Kapitel bietet einen kurzen Überblick über verschiedene Techniken zur Generierung von Grobgittern aus Hexaederzellen. Es ist als Serviceleistung für interessierte Leser konzipiert, weil dieses Thema mit dem Kontext der Arbeit untrennbar verbunden ist.

4.1.1 Allgemeine Beobachtungen

In diesem Kapitel sollen nur Methoden zur Erstellung strukturierter (konformer) Hexaedergitter betrachtet werden. Dies liegt im Grunde daran, daß der Bereich der unstrukturierten Hexaedergenerierung trotz langjähriger Forschung immer noch keine wirklich umfassenden, allgemeinen Ansätze hervorgebracht hat. Eine empfehlenswerte Informationsquelle zum Thema ist die Konferenz *International Meshing Roundtable (IMR)*, alle Tagungsbände der dort präsentierten Arbeiten sind unter `http://imr.sandia.gov` im Internet verfügbar. Die Darstellung in diesem Kapitel basiert im wesentlichen auf Übersichtsartikeln, die auf dieser Konferenz vorgestellt wurden (s. z.B. [6]).

Im Vergleich zu Tetraedern bieten Hexader aus numerischer Sicht einen eindeutigen Vorteil: Durch mehr Flexibilität bei der Wahl der Ansatzfunktionen (s. Kap. 2.1.3) kann eine höhere Genauigkeit erzielt werden. Aus praktischer Sicht ist es möglich, mit einer Hexaederdiskretisierung im Schnitt mit Faktor vier bis zehn weniger Elementen auf dem Grobgitter auszukommen und somit die Grobgittergleichung im Mehrgitteralgorithmus schneller zu lösen ist. Allerdings ist die Erzeugung von Hexaedergittern eine deutlich komplexere Aufgabe als die Erzeugung von Tetraedergittern.

Häufig dominiert die Zeit, um ein solches Gitter zu generieren, den gesamten Simulationsprozeß. Es gibt viele verschiedene Ansätze, die jedoch alle auf bestimmte Geometrien zugeschnitten sind. Kommerzielle Gittergenerierungssoftware bietet immer nur eine Teilmenge dieser Ansätze. Selbst bei sehr teuren kommerziellen Produkten benötigt ein Fachmann häufig noch

mehrere Tage, um für ein gegebenes Simulationsszenario ein Grobgitter zu erstellen, da sehr viel manuelle Arbeit nötig ist und die Struktur des gewünschten Gitters oftmals per Versuch und Irrtum geplant werden muß.

In diesem Abschnitt werden daher zunächst verschiedene Bedingungen formuliert, die ein idealer Gittergenerator einhalten sollte. Danach werden aus der zunächst einfach klingenden Strukturiertheitsforderung Folgerungen gezogen, die für die Erzeugung von Gittern und deren Qualität fundamentale Bedeutung haben. Der nächste Abschnitt untersucht exemplarisch existierende Ansätze, der übernächste stellt die Adaption der (interaktiven) Projektionstechnik aus dem letzten Kapitel auf den Gittererzeugungsprozeß vor.

4.1.2 Eigenschaften eines idealen Hexmeshers

Die Eigenschaften eines idealen Gittergenerators sollen unabhängig von existierenden Ansätzen und natürlich – soweit möglich – vom zu untersuchenden Problem sein. Es ist klar, daß sich einige dieser Ansprüche widersprechen, in der Praxis wird man bei der Auswahl der Algorithmen daher vorsichtig abwägen müssen, welche Aspekte für den konkreten Anwendungsfall wichtiger sind.

Geometrische Allgemeinheit Ein Gittergenerator sollte unabhängig von der Form, Topologie und der Konnektivität des zu diskretisierenden Volumens sein. Beispielsweise sollten in einem Strömungskanalszenario beliebig viele Hindernisse und Kanalverästelungen möglich sein.

Geometrische Anpassung Das Gitter soll sich in seiner Krümmung und seinem Verlauf der Geometrie anpassen. Insbesondere soll die Auflösung und die Qualität in den Regionen hoch sein, in denen in der Simulation interessante Effekte zu erwarten sind. Ein Beispiel ist das Grobgitter für die Kugelgeometrie (s. Kap. 5.2) aus Abbildung 4.1: Bei insgesamt 17 Elementen wird der Bereich um die Kugel mit 12 Hexaederzellen aufgelöst, dies entspricht einem Anteil von 70%. Man beachte, daß zum Zeitpunkt der Gittergenerierung schon bekannt war, daß sich der Einströmrand auf der linken Seite des Kanals befindet.

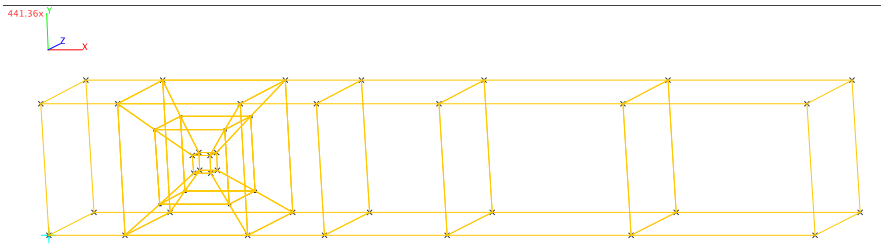


Abbildung 4.1: *Grobgitter für die Kugel-Konfiguration*

Geometrische Toleranz Da nicht immer von perfekten Modellen ausgegangen werden kann, soll ein Algorithmus tolerant gegenüber kleinen Fehlern in der Geometriebeschreibung wie Rissen, Löchern, oder Überlappungen sein.

Größenkontrolle Die Größe und die Anzahl der erzeugten Elemente soll vorgebbar sein, wenigstens in einer groben Ordnung. Beispielsweise kommt es vor, daß ein Simulationspaket sensibel auf große Seitenverhältnisse (*aspect ratios*, s. Kapitel 4.4) reagiert. Um von kleinen Zellen um ein Objekt zu großen Zellen in der Peripherie des Strömungsgebiets zu gelangen, sollte einem Gittergenerator beispielsweise der Gradient dieser Größenänderung vorgebbar sein, oder ein Gittergenerator sollte automatisch Übergangsschichten (engl. *transition layers*) einfügen. Das Gitter aus Abbildung 4.1 enthält eine solche Schicht, um den Größenunterschied zwischen Kugel und Kanal zu überwinden.

Geschwindigkeit Die Laufzeit soll deutlich unter der Laufzeit der numerischen Simulation liegen. Aus praktischen Gründen soll kein Supercomputer (wie für die eigentliche Berechnung) benötigt werden, sondern ein PC ausreichen.

4.1.3 Konsequenzen aus der Strukturiertheitsforderung

Aus der einfach klingenden Forderung, es sollen *strukturierte* Gitter erzeugt werden, lassen sich einige Folgerungen ziehen, die für die Generierung von Gittern und insbesondere für die Beurteilung der Gitterqualität wichtige Konsequenzen nach sich ziehen.

Gegenüberliegende Seitenflächen Ein Hexaeder kann als Volumen aufgefaßt werden, das von sechs paarweise gegenüberliegenden Seitenflächen begrenzt wird. So einfach wie diese Beobachtung auch klingt, die Konsequenzen sind wichtig: Weil sich Elemente Seitenflächen teilen, kann das gesamte Gitter als in alle drei Raumrichtungen verkettete Menge von „Hexaeder-Stapeln" aufgefaßt werden. Jeder dieser Stapel ist entweder eine geschlossene Kette von Zellen oder beginnt und endet am Rand des Simulationsgebiets. Insbesondere ist die Zahl der Randflächen immer gerade.

Fortpflanzung von Verfeinerungen Jede Verfeinerung eines einzelnen Elements muß sich daher fortpflanzen zu allen Elementen, die über die von der Verfeinerung betroffenen Seitenflächen des Elements erreichbar sind. Als Konsequenz sind nur sehr eingeschränkt lokale Verfeinerungen in Hexaedergittern möglich, jede Verfeinerung wird eine durchgehende neue Schicht Elemente erzeugen.

Oberflächengitter Jedes Volumengitter aus Hexaedern deckt die Geometrien mit einem Oberflächengitter aus Viereckszellen ab.

4.2 Kurzüberblick über existierende Ansätze

Existierende Ansätze zur Erzeugung von strukturierten Hexaedergittern lassen sich in vier verschiedene Kategorien einteilen. Dieser Abschnitt stellt die Vor- und Nachteile dieser Verfahrenstypen vergleichend gegenüber. Die Darstellung basiert auf Übersichtsartikeln, die im *Meshing Roundtable* veröffentlicht wurden, insbesondere auf [6]. Dort lassen sich auch Referenzen zu den zugrundeliegenden Originalarbeiten nachschlagen.

4.2.1 Schablonen für Primitive

Blöcke in einem Volumen mit einfacher Gestalt können mit einer Schablonentechnik vergittert werden. Beispiele für solche Blöcke sind Quader, Kugeln, Zylinder oder Pyramiden. In Systemen, die diese Technik unterstützen, wird der Anwender typischerweise zunächst manuell Blöcke im Gebiet definieren und diese Blöcke dann in Elemente unterteilen lassen.

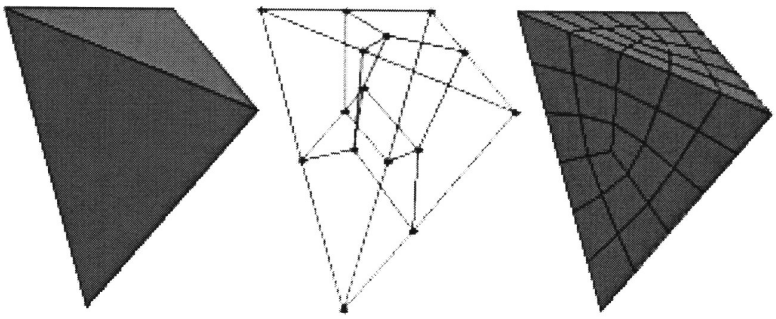


Abbildung 4.2: *Möglichkeit für eine Schablone für ein Tetraeder nach [6]*

Ein wichtiger, sehr mächtiger Spezialfall der Schablonentechnik ist der sogenannte *„general sweep"*. Ein Oberflächengitter wird hier entlang eines beliebigen Pfades automatisch extruiert, ein Beispiel zeigt Abbildung 4.3. Das entstehende Hexadergitter ist in der dritten Dimension strukturiert, das Oberflächengitter kann unstrukturiert sein. Alle ausgefeilten Techniken für die Erzeugung von Viereckgittern auf Oberflächen können als Ausgangspunkt dienen, so z.B. die Q-MORPH - Technik (s. [48]). Man nennt Verfahren dieses Typs auch 2.5D-Algorithmen.

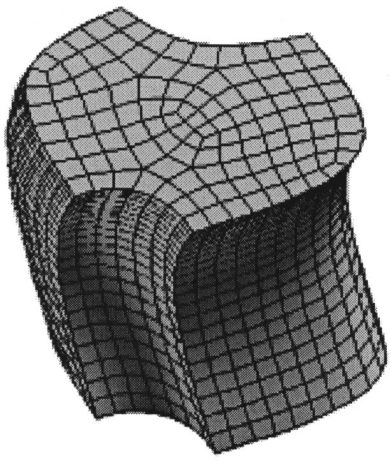


Abbildung 4.3: *Beispiel für einen general sweep nach [6]*

Die Vorteile dieser Verfahren liegen in der Geschwindigkeit, der guten Randanpassung und der

Qualität der erzeugten Elemente. Leider sind sie nur auf wenige, einfach strukturierte Geometrien anwendbar. Kompliziert ist es auch, geeignete Schnittstellen zwischen verschiedenen so vergitterten Blöcken zu erhalten. Man spricht dann von *block structured grids*. Abbildung 4.4 zeigt ein Beispiel für eine Zerlegung in Blöcke mit der kommerziellen Software ICEM CFD (s. `http://www.ansys.com`).

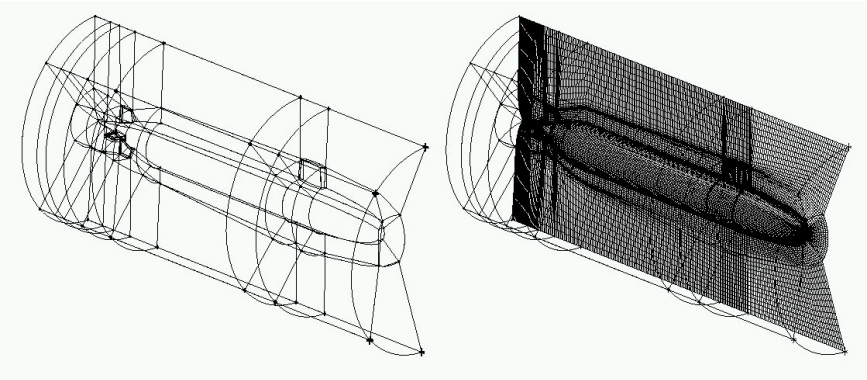


Abbildung 4.4: *Multiblock-Struktur eines Hexaedergitters um ein U-Boot, entnommen aus der Demoversion von ICEM CFD*

In einigen kommerziellen Softwarepaketen werden daher die Blöcke nicht über ihre Ausdehnung festgelegt, sondern über die Definition von Konnektoren. Erstellt wird in langwieriger Handarbeit gewissermaßen ein Skelett des geplanten Gitters. Die Erzeugung der eigentlichen Hexaederzellen ist danach automatisch und schnell möglich. Ein Beispiel ist TRUEGRID (`http://www.truegrid.com`).

4.2.2 Overlays

Overlay-Techniken arbeiten zweiphasig: Im ersten Schritt wird ein strukturiertes Hintergrundgitter aus vielen kleinen Zellen erzeugt. Es deckt den gesamten Hüllquader der Geometrie ab und ist meist achsenparallel ausgerichtet. Alternativ kann es auch am Objekt ausgerichtet sein. Danach werden die Knoten dieses Gitters klassifiziert in innere und äußere Knoten bzgl. des eigentlich zu vergitternden Volumens. Äußere Knoten werden eliminiert. Die verbleibenden Zellen, die jetzt keine Flächennachbarn mehr haben, müssen noch an den Rand der Geometrie angepaßt werden.

Das Verfahren ist vollautomatisch und mit Hilfe einer Octree-basierten Unterteilung auch effizient zu implementieren. Negativ fällt auf, daß gerade die Randelemente diejenigen von schlechter Qualität sind. Bereits existierende Oberflächengitter werden ignoriert oder alternativ in einigen Spezialfällen mit einer Map-Funktion integriert. Variationen in der Zellgroße (feine Zellen in der Nähe der Objekte, grobe Zellen im Ausströmbereich beispielsweise im virtuellen Windkanal), insbesondere *„transition layers”* sind nicht einfach zu integrieren.

4.2.3 Automatische Zerlegung

Diese Technik kann als Erweiterung der Schablonentechnik aufgefaßt werden. Die Idee besteht darin, die Zerlegung in Blöcke automatisch vorzunehmen, so daß die separaten Blöcke wieder wie Primitive behandelt werden können. Die konkreten Entscheidungskriterien, wo und wie ein gegebenes Volumen unterteilt wird, sind vielfältig: Einfache lokale geometrische Eigenschaften wie Winkelmessungen bis hin zu komplexen Skelettierungs- und Segmentierungsalgorithmen sind nur zwei Beispiele für die Bandbreite der vorgeschlagenen Methoden.

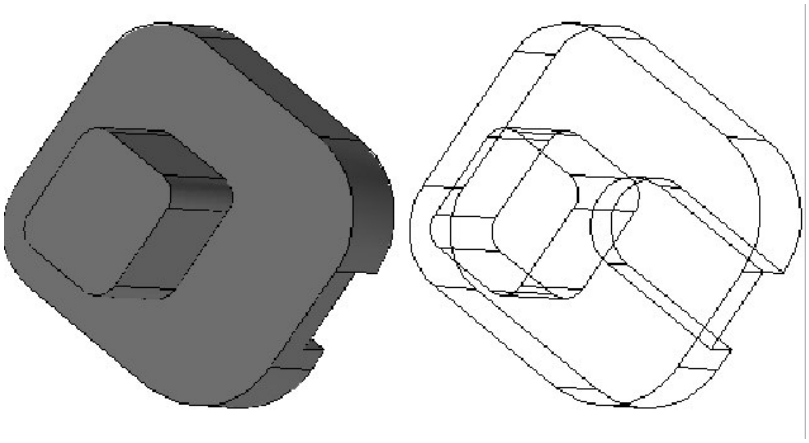


Abbildung 4.5: *Segmentierung einer Geometrie mit dem Cooper-Tool nach [6]*

Eine in der Praxis häufig genutzte Variation ist das *cooper tool* (s. [6]), eine Technik, die die Dekomposition parallel zur Gittererstellung und nicht nacheinander durchführt.

Die erzeugten Gitter sind typischerweise gut an den Rand angepaßt. Der Übergang zwischen verschiedenen Blöcken ist nur durch hohen Mehraufwand sicherzustellen. Die Menge der Geometrien, die nach aktuellem Stand der Forschung mittels einer automatischen Zerlegung segmentiert werden kann, ist recht eingeschränkt.

4.2.4 Advancing Front

Ausgehend von einem Oberflächengitter werden bei dieser Technik neue Elemente schichtweise hinzugefügt. Solange sich Elemente einer Schicht nicht schneiden, werden generell gute Gitter erzeugt. In allgemeinen Fällen ist es jedoch unmöglich, die aufeinanderprallenden Schichten strukturiert zu verbinden. *Advancing front* - Ansätze werden daher eher bei der unstrukturierten Gittergenerierung auf Tetraederbasis verwendet.
Ein neues Verfahrens namens *whisker weaving* zeigt vielversprechende Ergebnisse in speziellen Situationen. Hier wird nicht das Oberflächengitter ins Innere des Gebiets propagiert, sondern auf dem Dualen des Gitters gearbeitet.

4.3 Eigene Experimente

Dieser Abschnitt stellt einige Algorithmen vor, mit denen speziell im Kanalszenario (ein oder mehrere Objekte in einem Kanal) schnell ein Volumengitter erzeugt werden kann. Sie adaptieren die Idee der *overlay grids*. Sie wurden jedoch nur implementiert, um schnell Testgitter erzeugen zu können. Für komplexe, praktisch relevante Geometrien eignen sie sich nur sehr eingeschränkt.

4.3.1 Innerer Hüllquader

Dieser Algorithmus arbeitet in zwei Phasen: In der ersten Phase wird das komplette Volumen des Kanals mit einem Hintergrundgitter aus Hexaederelementen gefüllt. Im Gegensatz zum klassischen *overlay*-Ansatz geschieht die Generierung dieses Gitters in Abhängigkeit von verschiedenen Benutzerparametern, um die generelle Struktur des Gitters gerade für die Strömungssimulation zu beeinflussen. Im zweiten Schritt werden die Zellen gelöscht, die im Inneren des inneren Hüllquaders liegen, und die so entstehenden Randpunkte werden auf das innere Objekt projiziert.

Im Detail werden folgende Benutzerparameter benötigt:

- die Anzahl der gewünschten Unterteilungen (Schichten) des Volumens in x-, y- und z-Richtung jeweils „vor" und „hinter" dem inneren Objekt
- die Anzahl der Unterteilungen wieder entlang der drei Achsen „auf" dem inneren Objekt
- sechs Gradienten für die Verteilung der Zellen „vor" und „hinter" dem Objekt, jeweils entlang der drei Achsen

4.3.2 Mehrfache innere Hüllquader

Dieser Algorithmus ist eine Erweiterung des vorherigen: Anstelle eines großen inneren Hüllquaders werden beliebig viele benutzt, die vom Benutzer vorgegeben werden. So ist es möglich, auch Objekte mit Löchern (im einfachsten Beispiel ein Torus) oder mehrere innere Objekte zu behandeln.

4.3.3 Ein interaktiver Hexmesher

Beide Techniken haben sich für einfache Testgeometrien bewährt. Sie haben jedoch einen fundamentalen Nachteil gemeinsam: Die Zahl der erzeugten Zellen ist für praktische Anwendungen als Grobgitter in einem Mehrgitterszenario viel zu groß, wenn zur Auflösung der zu umströmenden Geometrie bereits viele Zellen benötigt werden. Andernfalls ist die Qualität der Zellen am Rand häufig zu schlecht. Dies gilt insbesondere auch für den zusätzlich implementierten klassischen Overlay-Algorithmus.

In Kombination mit dem Umbrella-Operator (s. Kap. 2.3.1) und den in Kapitel 2.3 vorgestellten Glättungsoperatoren konnte jedoch ein einfach zu bedienender interaktiver Gittergenerator implementiert werden. Seine Fähigkeiten sind in den Abbildungen 3.7 bis 3.13 in Kapitel

3.4.4 exemplarisch dokumentiert. Die Grundidee dieses Gittergenerators besteht in der Beobachtung, daß sich der für die Randanpassung in Mehrgitter-Szenarien entworfene Umbrella-Projektionsalgorithmus (in Kombination mit einer Gitterglättung für die inneren Knoten) gut für die interaktive Erzeugung eines qualitativ hochwertigen Grobgitters eignet.

4.4 Qualitätskriterien und Fehlererkennung

4.4.1 Allgemeines

Die folgenden Abschnitte beschreiben Qualitätsmaße für Hexaedergitter. Die Darstellung basiert im wesentlichen auf Kelly (s. [35]). Folgende Kriterien werden vorgestellt:

- Längenverhältnis (engl. *aspect ratio*)
- Winkelabweichung (engl. *angle deviation*)
- Parallelabweichung (engl. *parallel deviation*)
- Innenwinkel (engl. *corner angle*)
- Jacobi-Verhältnis (engl. *jacobian ratio (deviation)*)
- Verzerrung (engl. *warping factor*)

Alle diese Kriterien sind Funktionen der Elementgeometrie, d.h. sie verknüpfen Punktkoordinaten, Kanten und Flächen eines Hexaeders zu einem Maß für die Elementqualität. Die Schwellwerte, ab denen Elemente nicht mehr benutzt werden sollten, sind stark von der verwendeten Simulationssoftware abhängig. In einer Implementierung sollte ihre Festlegung also einem erfahrenen Anwender überlassen werden.

Im Gegensatz zum zweidimensionalen Fall lassen sich *aspect ratios*, *parallel deviations* und *warping factors* nicht direkt berechnen bzw. die direkte Berechnung liefert unzulässige Vereinfachungen. Für diese Qualitätsmaße wird daher indirekt vorgegangen: Alle sechs Seitenflächen sowie die drei Querschnittsflächen (Abbildung 4.6) werden wie Vierecke (engl. *quadrilaterals*) in 3D behandelt. Das Qualitätsmaß für das Element ist dann das der schlechteste Wert der einzelnen Testflächen. Ein wichtiger Hinweis: Auch wenn hier von „Flächen" gesprochen wird, sind diese im Allgemeinen nicht planar. Die weiter unten vorgestellten Algorithmen basieren dabei teilweise auf planaren Approximationen. Dies ist im übrigen auch der Grund, warum die sechs Seitenflächen nicht ausreichen, um die Qualität eines Hexaeder-Elements zu beurteilen.

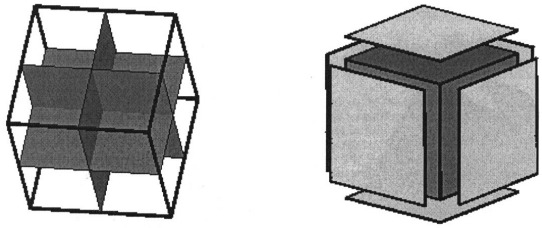


Abbildung 4.6: *Querschnittsflächen und Seitenflächen eines Hexaeders*

4.4.2 Längenverhältnis

Längenverhältnisse zählen zu den bekanntesten Qualitätskriterien. Sie sind ein Indikator für die zu erwartende numerische Stabilität während der Simulationsberechnung: Wie im Kapitel über mathematische Grundlagen dargelegt (s. Kap. 2.1.3), werden *Ansatzfunktionen* in den Punkten, Kanten und Flächen eines Finiten Elements ausgewertet und zwischen diesen Werten dann interpoliert. Hohe *aspect ratios* können zu starken Fehlern führen, sofern keine geeigneten Stabilisierungsverfahren eingesetzt werden.

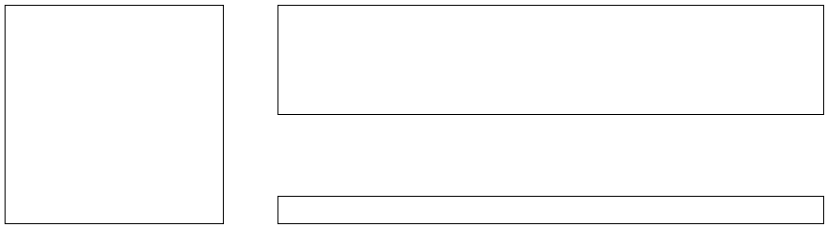


Abbildung 4.7: *Beispiele für aspect ratios von 1:1, 1:5 und 1:20*

Im Zweidimensionalen bezeichnet *aspect ratio* das Längenverhältnis von kürzester zu längster Kante eines Viereckselements. Ein Analogon in 3D, nämlich das Verhältnis größter zu kleinster Seitenfläche, ist vorstellbar, erfordert jedoch bei nichtplanaren Seitenflächen einen zu hohen Berechnungsaufwand zur Approximation dieser sog. *Regelflächen* oder *Minimalflächen*. Stattdessen wird eine Approximation der neun oben genannten Testflächen vorgenommen und jeweils das Seitenverhältnis eines solchen planaren 3D-Viereckselements berechnet. Als *aspect ratio* des Hexaeders wird dann der schlechteste Wert genommen. Klar ist: Ein ideales Element hat einen *aspect ratio* von 1, alle Kanten sind gleich lang, das Element ist würfelförmig.

Für jede Seitenfläche F, gegeben duch ihre vier Eckpunkte, wird der *aspect ratio* nach folgendem Algorithmus berechnet:

1. Berechne eine Ebene E als planare Approximation von F: E verläuft durch das Mittel der Eckpunkte und steht senkrecht zur gemittelten Normalen in den Eckpunkten.
2. Projiziere die Punkte senkrecht auf die Ebene E. Arbeite die restlichen Schritte mit den projizierten Punkten ab.
3. Konstruiere zwei Schnittlinien, die jeweils durch die Kantenmittelpunkte gegenüberliegender Kanten verlaufen.
4. Konstruiere zwei Rechtecke, deren Seiten von zwei gegenüberliegenden Kantenmittelpunkten halbiert werden und deren andere Seiten die verbleibenden beiden Kantenmittelpunkte enthalten.
5. Der *aspect ratio* von F ist das Maximum des Quotienten aus längster und kürzester Kante beider Rechtecke.

Algorithmus 4.1: Berechnung des aspect ratios für ein 3D-Viereckselement

4.4.3 Winkelabweichung

Die Winkelabweichung mißt, wie stark die Innenwinkel zwischen jeweils zwei adjazenten Kanten in einem Hexaeder oder einem Viereckselement von einem rechten Winkel abweichen. Ihre Berechnung ist trivial. Als Winkelabweichung wird das Maximum aller so berechneten Werte bezeichnet. In einigen Simulationspaketen ist dieser Wert wichtig, wenn beispielsweise für die Berechnung von Ableitungswerten rechteckige Zellen vorausgesetzt werden.

Abbildung 4.8: *Beispiele für Winkelabweichungen von 0,10 und 45 Grad*

4.4.4 Parallelabweichung

Die Parallelabweichung mißt in Grad, wie weit gegenüberliegende Seiten eines Viereckselements oder Hexaeders von einer parallelen Lage zueinander abweichen. Weil die Seitenflächen eines Hexaeders nicht notwendig planar sind (und somit der klassische Parallelitätsbegriff keinen Sinn macht), wird analog zu Algorithmus 4.1 die Parallelabweichung für planare Approximationen der neun Testflächen (s.o.) berechnet und darüber das Maximum gebildet. Stark von einer parallelen Lage abweichende Seiten können zu Konvergenzproblemen führen.

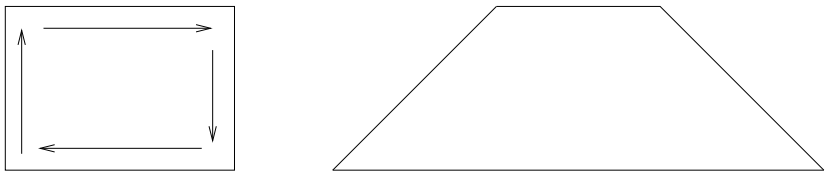


Abbildung 4.9: *Beispiel für Parallelabweichungen von 0 bzw. 90 Grad*

Die Berechnung erfolgt über das Skalarprodukt jeweils gegenüberliegender Seiten, wobei auf eine konsistente Richtung der Seitenvektoren (wie in Abb. 4.9 durch die Pfeile skizziert) geachtet werden muß.

4.4.5 Innenwinkel

Die Berechnung maximaler Innenwinkel wird für alle Paare adjazenter Kanten eines Hexaeders oder Viereckselements durchgeführt. Mit diesem Maß können nicht tolerierbar starke Elementdegradationen schnell detektiert werden. Beispielsweise hat ein zu einem Dreieck degradiertes Viereckselement einen maximalen Innenwinkel von $180°$.

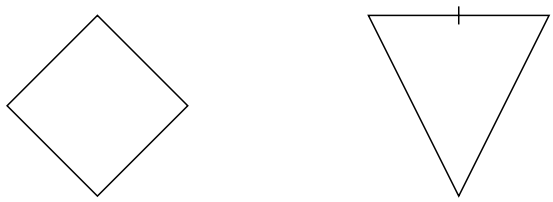


Abbildung 4.10: *Beispiele für maximale Innenwinkel von von 90 bzw. 180 Grad*

4.4.6 Jacobi-Verhältnis

Dieses Maß ist für viele Simulationscodes und auch in der Visualisierung (*particle tracing*) von fundamentaler Bedeutung, und zwar immer dann, wenn die verwendeten Algorithmen eine Koordinatentransformation zwischen einem Tensorproduktgitter (*computational space*) und einem verzerrten, randangepaßtem Gitter (*physical space*) beinhalten. Man spricht von einer Transformation auf Referenzelemente. Die Jacobi-Matrix zu einem zu transformierenden Punkt bzw. ihre Inverse ist dann nämlich gerade die Transformationsmatrix, ihre Determinante gibt den Größe dieser Transformation an. Als Jacobi-Verhältnis wird das Verhältnis von größter zu kleinster Determinante der Jacobi-Matrix in ausgesuchten Testpunkten des Elements bezeichnet. Ein Jacobi-Verhältnis nahe 0 impliziert somit die Singularität der Matrix, sehr hohe Werte bedeuten, daß die Transformation unzuverlässig wird. In einem idealen Element gibt es keinen Vorzeichenwechsel in den Testpunkten, und die Werte sind alle ähnlich groß. Deshalb wird häufig nicht nur von Jacobi-Verhältnis, sondern auch von Jacobi-Abweichung gesprochen.

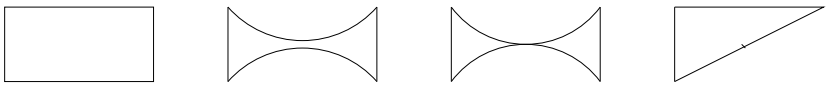


Abbildung 4.11: *Beispiele für Jacobi-Verhältnisse von 1, 30, 100 und 0*

Für Hexaeder wird das Jacobi-Verhältnis aus allen Eckpunkten berechnet: In jedem Punkt wird die Jacobi-Matrix aufgestellt, dabei ist sicherzustellen, daß die drei Vektoren immer gleich gerichtet sind (sie zeigen alle vom Testpunkt weg) und immer in der gleichen Reihenfolge als Spaltenvektoren in die Matrix eingetragen werden. Hier hilft die *„rechte-Hand-Regel"*.

4.4.7 Verzerrungsfaktor

Mit „Verzerrung" eines Viereckselements ist hier die Abweichung aus der Ebene gemeint. Übertragen auf Hexaeder mißt sie auch die Verdrehung zweier gegenüberliegender Seitenflächen zueinander und ist ein indirekter Indikator für deren Nichtplanarität.

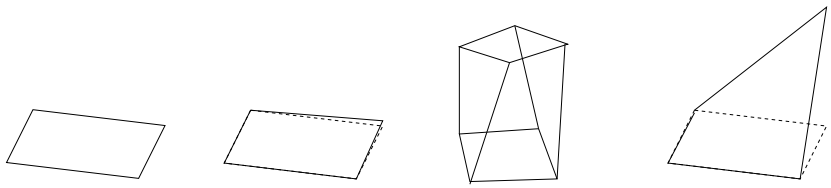


Abbildung 4.12: *Beispiele für warping factors von 0,* 0.4*,* 0.4 *(bzw. 45 Grad) und 5*

Die Berechnung für Viereckselemente erfolgt nach folgendem Algorithmus:

1. Berechne eine Approximationsebene E. E verläuft durch den Schnittpunkt der Elementdiagonalen und steht senkrecht auf diese Diagonalen im Schnittpunkt.
2. Projiziere die vier Eckpunkte auf E.
3. Berechne den Flächeninhalt A des projizierten Elements.
4. Berechne den Abstand h entlang der Normale vom ursprünglichen zum projizierten Punkt.
5. Der Verzerrungsfaktor für das Element ergibt sich als Quotient aus h und der Wurzel aus A.

Algorithmus 4.2: Berechnung des Verzerrungsfaktors für Viereckselemente

Für Hexaeder wird dieser Algorihmus auf die sechs Seitenflächen angewendet, der Verzerrungsfaktor ist einfach das Maximum der Flächenfaktoren.

Kapitel 5

Ergebnisse

In diesem Kapitel werden die numerisch-experimentellen Ergebnisse der Arbeit vorgestellt. Anhand zweier Benchmark-Konfigurationen, die beide sowohl mit einer analytischen Randbeschreibung als auch mit Oberflächentriangulierungen verschiedener Feinheit durchgeführt werden, sollen zunächst folgende Fragen experimentell beantwortet werden:

1. Wie groß ist der Einfluß der (Feinheit der) Triangulierung auf die Genauigkeit der Lösung? Gibt es insbesondere Unterschiede bezüglich einer qualitativen und einer quantitativen Analyse?

2. Sind bei hinreichender Feinheit des Dreiecksnetzes Unterschiede zur analytischen Randbeschreibung (die per Definition keinen Randapproximationsfehler impliziert) feststellbar? Andersherum: Verursacht eine zu grobe Oberflächentriangulierung zusätzliche Fehler?

3. Welchen Einfluß übt die approximative Randdarstellung auf das Konvergenzverhalten eines Mehrgitterlösers aus?

In einem dritten Test wird versucht, die gleichen Fragen zu beantworten, allerdings nicht im Bezug auf den Einfluß einer Oberflächentriangulierung, sondern im Bezug auf den Einfluß verschiedener Gitterglättungsstrategien.

Zusätzlich wird abschließend eine qualitative Analyse eines Fahrzeugs (als Beispiel für eine nicht analytisch darstellbare Geometrie) in einem Windkanal durchgeführt. Eine quantitative Untersuchung ist hier – unabhängig von den in der Arbeit entwickelten Techniken – nicht möglich. Dies leitet über zu einer kritischen Diskussion der Ergebnisse und einem Ausblick auf mögliche Auswege jenseits des Themengebiets dieser Arbeit.

Alle Implementierungen, mit denen die nun folgenden numerischen Experimente durchgeführt wurden, basieren auf einer Erweiterung der Software FEATFLOW , konkret des `cc3d`-Lösers. Dieser Löser ist geeignet zur Simulation stationärer Strömungen bei niedrigen Reynoldszahlen und Flußgeschwindigkeiten. Die abgebildeten Visualisierungen wurden mit GMV generiert, zu finden unter `http://www-xdiv.lanl.gov/XCM/gmv/GMVHome.html` .

5.1 DFG–Benchmark

Definition des Benchmarks Der DFG–Benchmark *„Laminar flow around a cylinder"* wurde im Rahmen des Schwerpunktprogramms *„Flow simulation with high-performance computers"* von Schäfer und Turek im Jahr 1996 definiert (s. [52]). Aktuelle Veröffentlichungen wie [8, 33] zeigen die unverminderte Aktualität und Relevanz dieses Benchmarks.

Das Gebiet Ω ist in Abbildung 5.1 skizziert.

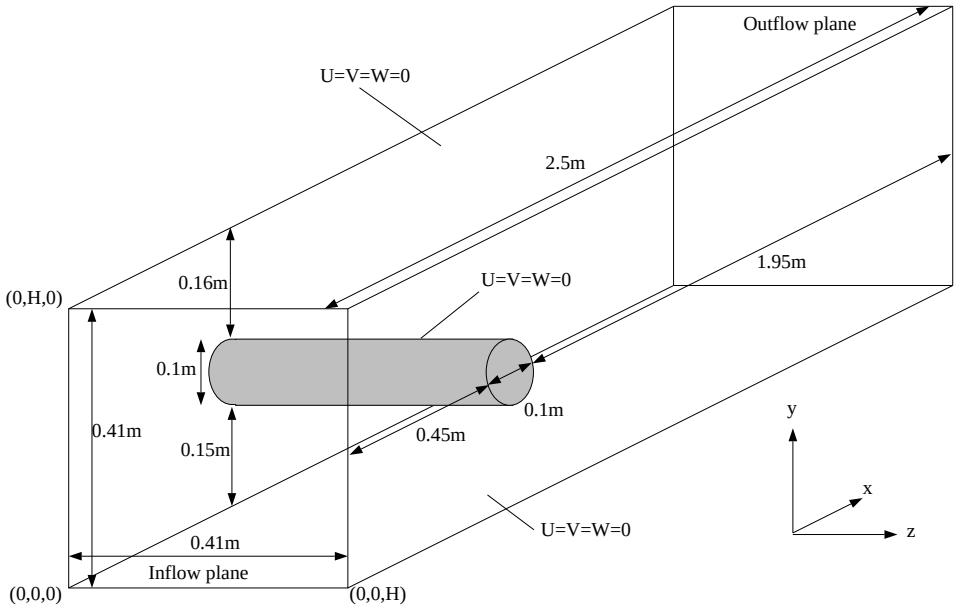


Abbildung 5.1: *Zylinderbenchmark-Geometrie*

Der Kanal hat die Höhe $H = 0.41m$, der Zylinder den Durchmesser $D = 0.1m$. Gelöst werden sollen die inkompressiblen Navier-Stokes-Gleichungen (s. Kap. 2.3) in den Unbekannten Geschwindigkeit ($\mathbf{u}$) und Druck (p), mit einer Viskosität des Fluids von $\nu = 10^{-3}m^2/s$ und einer Dichte von $\rho = 1kg/m^3$. Die Strömung betritt den Kanal durch die in obiger Abbildung mit *inflow plane* bezeichnete Seitenfläche, und zwar mit dem parabolischen Profil

$$\mathbf{u} = \begin{pmatrix} 16Uyz(H-y)(H-z)/H^4 \\ 0 \\ 0 \end{pmatrix},$$

wobei $U = 0.45m/s$ die Einströmgeschwindigkeit angibt. Die Strömung ist mit einer Reynoldszahl von 20 stationär, die mittlere Einströmgeschwindigkeit ist $\bar{U} = 0.2m/s$.

Die Werte, die im Benchmark berechnet werden sollen, sind die sogenannten Auftriebs- und Widerstandsbeiwerte (engl. *lift*, *drag*) c_l, c_d sowie die Druckdifferenz Δp zwischen den beiden Punkten $(0.45; 0.2; 0.205)$ und $(0.55; 0.2; 0.205)$ auf der Oberfläche S des Zylinders.

Mit der äußeren (d.h. ins Innere des Gebiets zeigenden) Normale des Zylinders $\mathbf{n}_S$ und den beiden Tangentialvektoren $\mathbf{t}_1, \mathbf{t}_2$,

$$\mathbf{n}_S = \begin{pmatrix} n_x \\ n_y \\ 0 \end{pmatrix}, \mathbf{t}_1 = \begin{pmatrix} n_y \\ -n_x \\ 0 \end{pmatrix}, \mathbf{t}_2 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

lassen sich die einwirkenden Kräfte des Luftwiderstands und des Auftriebs, F_d und F_l, aufstellen:

$$F_d = \int_S \left(\rho\nu \frac{\partial(\mathbf{u}\cdot\mathbf{t}_1)}{\partial \mathbf{n}_S} n_y - p n_x \right) ds, \quad F_l = -\int_S \left(\rho\nu \frac{\partial(\mathbf{u}\cdot\mathbf{t}_1)}{\partial \mathbf{n}_S} n_x + p n_y \right) ds \tag{5.1}$$

Hieraus ergeben sich die gesuchten Koeffizienten wie folgt:

$$c_d = \frac{2F_d}{\rho \tilde{U}^2 DH}, \quad c_l = \frac{2F_l}{\rho \tilde{U}^2 DH}$$

Es gelten die Referenzwerte aus der folgenden Tabelle:

Autor	Drag	Lift	$\Delta(p)$	# Unbekannte
John [33]	6.18533	0.009401	0.170875	55.6 Mio.
Braack,Richter [8]	6.185331	0.00940136	0.171342	62 Mio.

Tabelle 5.1: Referenzwerte im Zylinder-Benchmark

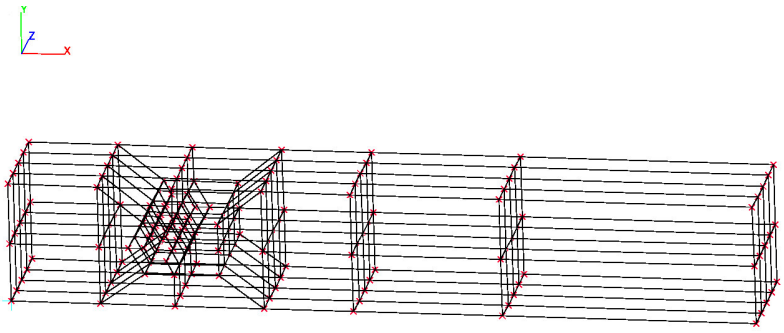


Abbildung 5.2: *Grobgitter im Zylinderbenchmark*

Konstruktion von Testfällen Der Zylinder hat eine Höhe von $H = 0.41$, sein Durchmesser beträgt $D = 0.1$. Das Grobgitter (s. Abbildung 5.2) verwendet zur Approximation des Zylinderumfangs $U = D\pi = \pi/10$ in der x, y-Ebene 6 Kanten. Analog werden zur Approximation der Zylinderhöhe in Richtung z-Achse 4 Kanten verwendet. Für die Längen der Kanten, die den Umfang auf verschiedenen Leveln approximieren (d.h. für die maximale Auflösung des Zylinderumfangs durch das Hexaedergitter) ergibt sich folgende Tabelle:

Level	# Kanten	Kantenlänge
1	6	0.0524
2	12	0.0262
3	24	0.0131
4	48	0.0065
5	96	0.0033
6	192	0.0016

Tabelle 5.2: Kantenlängen in der x, y-Ebene

Drei Testfälle sollen im folgenden betrachtet werden. Sie unterscheiden sich in der Zahl der Dreiecke, die zur Zylinderapproximation verwendet werden:

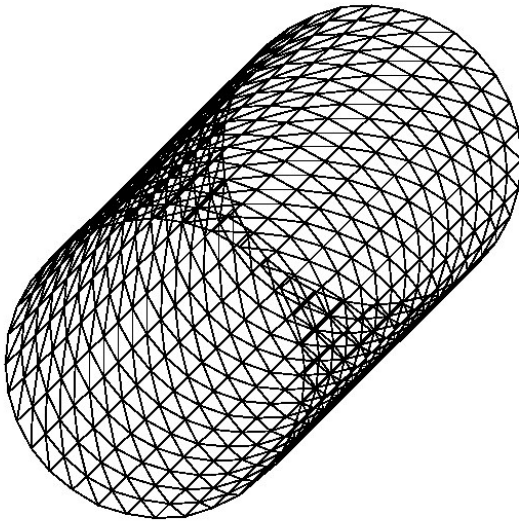


Abbildung 5.3: *Approximation des Zylinders durch Dreiecke*

Eine sehr grobe Diskretisierung (genannt T16) löst den Zylinderumfang mit 16 Kanten auf, eine mittlere (genannt T64) verwendet 64 Kanten und eine feine Version (genannt T1024) 1024 Kanten. Die Konfigurationen wurden so gewählt, daß die grobe Auflösung schon visuelle Artefakte erzeugen, die mittlere mit einer Kantenlänge von 0.05 bis einschließlich Level 4 den Zylinder genauer als die Hexaeder-Kanten auflösen, während die feine Diskretisierung mit einer Kantenlänge von 0.0003 selbst auf Level 6 hinreichend genau sein sollte. In Richtung der z-Achse können nach Konstruktion diese Diskretisierungsfehler nicht auftreten, von daher ist die Zahl der „Schichten" nicht von Bedeutung.

Man beachte, daß nach Konstruktion der Geometrie und des Grobgitters die Projektion mit kürzesten Abständen und die gewichtete Projektion (s. Kap. 3.4) hier äquivalent sind. Außerdem ist in diesem Fall die Projektionsaufgabe eindeutig lösbar. Daher wurden alle Tests nur mit der klassischen Projektionstechnik durchgeführt.

Ein Vergleich von Visualisierungen der analytischen Projektion mit Datensätzen, die eine Oberflächentriangulierung verwenden, ergibt schon bei der mittleren Auflösung des Zylinders keine sichtbaren qualitativen Unterschiede. Die Abbildungen 5.4 und 5.5 zeigen Beispiele für das qualitative Strömungsverhalten, jeweils durchgeführt mit einer analytischen Beschreibung des Zylinders. Die Abbildungen 5.6 und 5.7 zeigen dieselbe Simulation, diesmal jedoch durchgeführt mit der Zylinderbeschreibung T64.

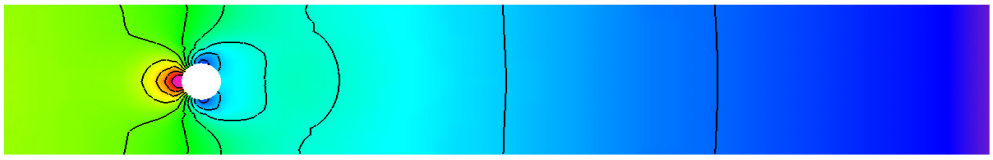


Abbildung 5.4: *Visualisierung der Druckverteilung mit Konturlinien in der zentralen Schnittebene durch den Strömungskanal, analytische Randbeschreibung*

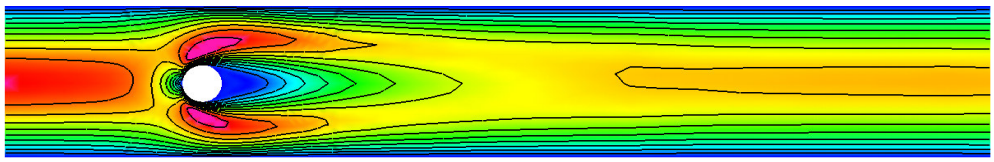


Abbildung 5.5: *Visualisierung der Norm des Geschwindigkeitsvektors mit Konturlinien in der zentralen Schnittebene durch den Strömungskanal, analytische Randbeschreibung*

Abbildung 5.6: *Visualisierung der Druckverteilung mit Konturlinien in der zentralen Schnittebene durch den Strömungskanal, Oberflächentriangulierung T64*

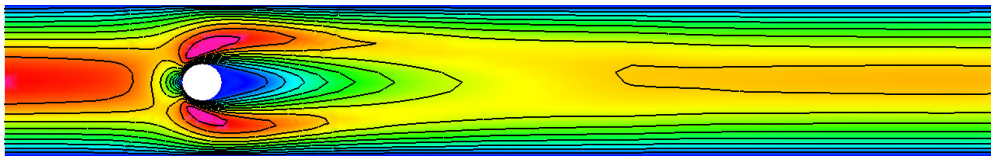


Abbildung 5.7: *Visualisierung der Norm des Geschwindigkeitsvektors mit Konturlinien in der zentralen Schnittebene durch den Strömungskanal, Oberflächentriangulierung T64*

Analyse der Testläufe Das verwendete Grobgitter führt zu den Problemgrößen, die in der folgenden Tabelle wiedergegeben sind. Dabei ergibt sich die Anzahl der Unbekannten gerade als Summe der Anzahl der Flächen für die x,y und z-Komponente der Geschwindigkeit und der Zahl der Elemente für den Druck.

Level	# Flächen	# Elemente	# Unbekannte
1	396	96	1284
2	2576	768	8496
3	19520	6144	64704
4	151808	49152	504576
5	1197056	393216	3984384
6	9506816	3145782	31666230

Tabelle 5.3: Problemgröße des Benchmarks auf verschiedenen Leveln

Die Rechenzeit betrug pro Durchlauf auf Level 6 jeweils drei Stunden auf einem Opteron-System mit 1.8 GHz. Pro Durchlauf wurden 4.5 GB RAM benötigt, was Rechnungen auf feineren Leveln mit der zur Verfügung stehenden Hardware-Ausstattung unmöglich macht.

Die Resultate für die Drag-Berechnung sind der folgenden Tabelle zu entnehmen. Die ersten vier Spalten enthalten die konkret berechneten Werte, jeweils für die sehr grobe, mittlere und feine Oberflächentriangulierung sowie eine analytische Beschreibung des Zylinders. Die letzten vier Spalten enthalten für diese Werte jeweils den relativen Fehler ($|c_d - c_d^{ref}|/c_d^{ref}$) gegenüber [8], gemessen in Prozent.

Level	T16	T64	T1024	analytisch	Err T16	Err T64	Err T1024	Err analytisch
3	5.7711	5.8333	5.8550	5.8746	6.6970	5.6914	5.3406	5.0237
4	6.0499	6.1160	6.1387	6.1427	2.1896	1.1209	0.7539	0.6892
5	6.0900	6.1562	6.1786	6.1795	1.5412	0.4719	0.1088	0.0943
6	6.0983	6.1612	6.1840	6.1842	1.4071	0.3901	0.0215	0.0183

Tabelle 5.4: Berechnete Werte und relative Fehler in Prozent bei der Drag-Berechnung

Wie erwartet ist der Unterschied zwischen den verschieden groben Zylinderdiskretisierungen klar erkennbar. Vergleicht man beispielsweise die relativen Fehler zwischen T16 und T64, so ist die Genauigkeit bei T64 auf Level 4 besser als auf Level 6 bei der schon optisch nicht mehr runden Zylinderapproximation. Dies entspricht einem Gewinn von zwei Leveln, also einem Rechenzeitgewinn um den Faktor 64. Diese Aussage ist allerdings mit Vorsicht zu genießen, da in praktischen Anwendungsfällen selbstverständlich keine Referenzwerte zur Verfügung stehen und es daher a priori nicht bekannt ist, wie genau ein Ergebnis ist. Im Fall eines standardisierten Benchmarks ist diese Aussage jedoch legitim. Vergleicht man T64 mit T1024, so ist der Effekt eines vollen Levelgewinns bei Level 5 der feinen Randbeschreibung analog festzustellen. Auf der anderen Seite ist die feine Triangulierung T1024 in diesem Fall so fein gewählt, daß der (bei einer analytischen Randbeschreibung prinzipiell nie auftretende) Randapproximationsfehler in allen betrachteten Leveln (noch) nicht dominiert: Der Unterschied in

den jeweiligen relativen Fehlern liegt im Promillebereich.

Schon ab Level 4 sind die auftretenden Fehler kleiner als ein Prozent, während diese Schranke mit der sehr groben Triangulierung nicht erreichbar ist. Der Verlauf der Fehlerreduktion bei dieser Triangulierung (s. nächste Abbildung, man beachte die logarithmische Skalierung) zeigt auch, daß selbst bei noch mehrmaliger Unterteilung diese Schwelle nicht zu unterschreiten wäre. Einen zusätzlichen Fehler neben dem Randapproximationsfehler bewirkt die Verwendung einer zu groben Triangulierung nicht, lediglich der Zeitpunkt, ab dem der Diskretisierungsfehler am Rand den gesamten Fehler dominiert und damit eine weitere Unterteilung überflüssig macht, wird deutlich nach vorne verschoben.

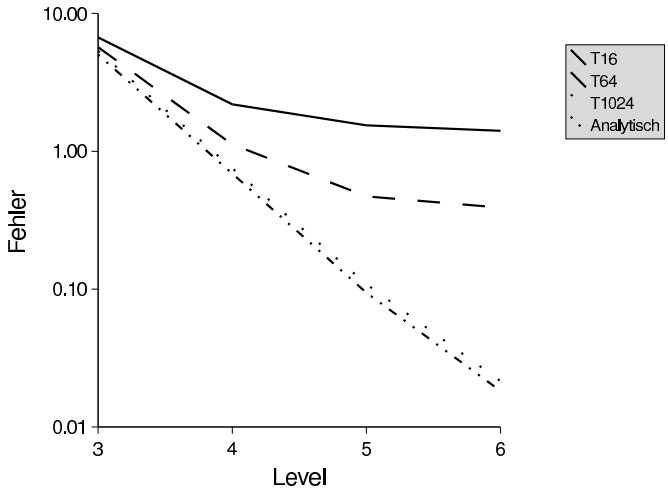


Abbildung 5.8: *Verlauf der relativen Fehler in Prozent, gemessen für die vier Randbeschreibungen. Die y-Achse ist* ***logarithmisch*** *skaliert!*

Dieses Diagramm bestätigt die Annahmen, die zur Konstruktion dieser Testfälle getroffen wurden: Die sehr grobe Diskretisierung ist unbrauchbar, die mittlere Version ist bis Level 4 brauchbar, die feine ist von der analytischen Randbeschreibung bis zu Level 6 nicht zu unterscheiden. Ein Vergleich der Kantenlängen (s.o.) impliziert, daß bei T1024 ab Level 9 der Diskretisierungsfehler dominieren wird, auf diesem Level werden 1024 Dreieckskanten von 1536 Hexaeder-Kanten abgedeckt. Dies entspricht allerdings einem Speicherbedarf von über 2 TB, was die experimentelle Verifikation ausschließt.
Für den Lift-Koeffizienten ergeben sich folgende Tabellen:

Level	T16	T64	T1024	analytisch
3	0.015026	0.005960	-0.000349	-0.000579
4	0.008837	0.006338	0.004585	0.004499
5	0.008364	0.007949	0.007780	0.007758
6	0.008558	0.008803	0.008951	0.008946

Tabelle 5.5: Berechnete Lift-Werte.

Level	Err T16	Err T64	Err T1024	Err analytisch
3	59.8279	36.6060	103.7081	106.1605
4	6.0008	32.5842	51.2315	52.1463
5	11.0341	15.4473	17.2492	17.4832
6	8.9685	6.3657	4.7904	4.8489

Tabelle 5.6: Relative Fehler in der Lift-Berechnung in Prozent.

Die oben notierten Beobachtungen zur erreichbaren Genauigkeit mit den verschiedenen Oberflächentriangulierungen spiegeln sich hier wider. Es muß jedoch erwähnt werden, daß die Berechnung dieses Koeffizienten in diesem Benchmark sehr schwierig ist. Ein Indikator dafür ist die auftretende Oszillation auf den groben Leveln bzw. bei T16, ein weiterer ist der Vergleich der relativen Fehler in der analytischen Projektion zwischen dem Drag- und dem Lift-Koeffizienten: Beim Drag-Koeffizient kann ein relativer Fehler von weniger als ein Promille erreicht werden, beim Lift-Wert beträgt das beste Ergebnis fast fünf Prozent. Auch ein Levelgewinn durch die Verwendung einer hochauflösenderen Randbeschreibung ist nicht erkennbar. Da dieser Gewinn noch nicht einmal im Vergleich zur analytischen Zylinderrepräsentation auftritt, können diese Ergebnisse diesbezüglich nur tendenziell bewertet werden. Nicht nur tendenziell, sondern deutlich bestätigt diese Tabelle allerdings den verschwindenden Unterschied zwischen der feinen Triangulierung und der analytischen Randbeschreibung bis zum maximalen Level 6.

Die Ergebnisse in der Berechnung der Druckdifferenz zwischen zwei gegenüberliegenden Punkten auf der Zylinderoberfläche sind in Tabelle 5.7 zusammengestellt.

Level	T16	T64	T1024	analytisch	Err T16	Err T64	Err T1024	Err analytisch
3	0.1343	0.1350	0.1352	0.1352	21.61	21.22	21.1	21.07
4	0.1551	0.1558	0.1561	0.1561	9.49	9.04	8.9	8.9
5	0.1640	0.1648	0.1651	0.1651	4.27	3.8	3.64	3.64
6	0.1680	0.1683	0.1686	0.1686	1.97	1.76	1.58	1.58

Tabelle 5.7: Relative Fehler in der Berechnung der Druckdifferenz in Prozent.

Die Berechnung dieser Druckdifferenz erfolgt über zwei Punktauswertungen, die in allen Konfigurationen schon durch das Grobgitter mit einem Knoten exakt auf der Zylinderoberfläche erfaßt werden. Diese Tabelle spiegelt also den Einfluß der Feinheit in der Randauflösung auf die Genauigkeit einer Punktauswertung wider. Liest man die Tabelle spaltenweise, so ist erkennbar, daß der Druck im Strömungsgebiet natürlich um so genauer berechnet werden kann, desto feiner die Diskretisierung im Ort ist. Liest man die Tabelle hingegen zeilenweise für die beiden Level 5 oder 6, so ist ein leichter Effekt durch die Genauigkeit der Oberflächentriangulierung erkennbar: Die relativen Fehler auf Level 6 zwischen der sehr groben und der analytischen Randbeschreibung unterscheiden sich um ca. 0.5%. Man wird erwarten, daß bei Punktauswertungen der Diskretisierungsfehler keine Auswirkungen hat, gerade weil der relevante Punkt schon im Grobgitter exakt auf dem Randobjekt positioniert ist. Daß trotzdem ein (leichter) Effekt notierbar ist, hat zwei Gründe: Der Druck ist eine zellbasierte Größe und wird für diese Punktauswertungen aus den vier adjazenten Hexaedern gemittelt, deren weitere

Randknoten nicht im Grobgitter enthalten sind und somit von der Feinheit der Oberflächentriangulierung beeinflußt werden. Andererseits ist der Diskretisierungsfehler keine lokale Größe, die Fehler in der Randapproximation bewirken globale Fehler in der Druckberechnung, obwohl der relevanten Punkt exakt positioniert ist. Im Vergleich zu den vorherigen Koeffizienten des Auftriebs und Luftwiderstands ist der Einfluß einer groben Oberflächentriangulierung auf die Genauigkeit einer Punktauswertung deutlich geringer als der Einfluß auf ein vollständiges Oberflächenintegral.

Zum Vergleich zeigt die nächste Tabelle die Ergebnisse in der Drag-Auswertung, die mit leicht anderen Parametern in FEATFLOW berechnet wurden. Hier werden nur die grobe und feine Oberflächentriangulierung mit den Referenzwerten verglichen. Die Tabelle soll den nur geringen Einfluß von Variationen an den FEATFLOW – Parametern auf die hier zu untersuchenden Fragestellungen belegen.

Level	T64	T1024	Err T64	Err T1024
3	5.75060	5.77110	7.0234	6.6920
4	6.09170	6.11430	1.5085	1.1431
5	6.14980	6.17210	0.5691	0.2086
6	6.15960	6.18240	0.4107	0.0420

Tabelle 5.8: Relative Fehler in der Berechnung des Drags in Prozent, alternative Konfiguration.

Zwei Dinge sind zu notieren: Zum einen erkennt man im direkten Vergleich mit den in Tabelle 5.4 dargestellten relativen Fehlern, daß sich die Genauigkeit der Oberflächentriangulierung wieder in einem vollständigen Level-Gewinn bemerkbar macht und die gleichen Tendenzen zeigt. Zum anderen fällt auf, daß die Variation von Parametern in FEATFLOW die Genauigkeit nicht stark beeinflußt. Dies wird durch die verbleibenden drei gerechneten Konfigurationen, die hier nicht weiter vorgestellt werden sollen, bestätigt.

Einfluß auf das Konvergenzverhalten des Mehrgitterlösers Abschließend soll noch betrachtet werden, wie sich die verschieden genaue Randauflösung auf das Konvergenzverhalten des Mehrgitterlösers auswirkt. Die relevanten Parameter hierfür sind die Anzahl der vom Löser ausgeführten Schritte in der äußeren nichtlinearen Iteration und in der inneren Mehrgitteriteration. Hieraus ergibt sich als Indikator für die Robustheit des Lösers die durchschnittliche Anzahl Mehrgitterschritte pro nichtlinearer Iteration.

Für die Zylinderkonfiguration ergibt sich die folgende Tabelle:

Level	analytisch	T1024	T64	T16
2	20/3.55	20/3.55	20/3.55	20/3.55
3	13/3.31	13/3.38	13/3.31	13/3.23
4	8/5	8/5	8/5	8/5
5	7/4	7/4	7/4	7/4
6	6/3.5	6/3.5	6/3.5	7/3.57

Tabelle 5.9: Anzahl nichtlinearer Iterationen gesamt, durchschnittliche Anzahl Mehrgitterschritte pro Iteration für verschieden feine Oberflächenbeschreibungen auf allen Leveln.

An dieser Tabelle liest man ab, daß sich das Mehrgitterverfahren hier sehr gut verhält: Die durchschnittlichen Werte schwanken nur sehr wenig, d.h. das Mehrgitterverfahren ist (wie theoretisch gefordert) unabhängig vom Level. Außerdem erkennt man die lineare Laufzeit (in der Anzahl der Unbekannten auf dem jeweils maximalen Level) des gesamten Algorithmus. Die Feinheit der Oberflächenbeschreibung hat in diesem Beispiel offenbar (ignoriert man das statistische Rauschen auf dem zu groben Level 3) keinen bemerkenswerten Einfluß auf das Konvergenzverhalten. Dies liest man zeilenweise aus der Tabelle ab. Der Rückgang der notwendigen nichtlinearen Iterationen zur Fehlerreduktion bis zur vorgeschriebenen Grenze bei feineren Leveln wird verursacht durch den Vorgehen, als Startlösung für ein feines Level nicht die Null-Lösung wie auf Level 1, sondern die Lösung des nächstgröberen Levels zu verwenden. Die hier aus Gründen der Übersichtlichkeit nicht weiter tabellierten verbleibenden vier Konfigurationen bestätigen dieses Ergebnis.

Zusammenfassung Bei der qualitativen Untersuchung des Strömungsverlaufs ist kein Unterschied zwischen der analytischen Randbeschreibung und der Beschreibung durch eine Oberflächentriangulierung festzustellen, sofern die gewählte Randdarstellung nicht schon optische Artefakte beinhaltet. Bei der quantitativen Untersuchung der drei im Benchmark zu berechnenden Größen Luftwiderstand, Auftrieb und Druckdifferenz treten die erwarteten Ergebnisse auf: Eine hinreichend feine Triangulierung liefert bei vernachlässigbarem Unterschied die gleiche Genauigkeit wie die analytische Randbeschreibung. Der Einfluß der Feinheit der Triangulierung ist hierbei skaliert: Bei einer zu groben Diskretisierung dominiert schon auf vergleichsweise niedrigen Leveln der Randapproximationsfehler den Gesamtfehler, während eine hinreichend feine Diskretisierung keine Dominanz des Randfehlers bis zum maximal möglichen Level impliziert. Einen zusätzlichen Fehler durch die Verwendung einer zu groben Diskretisierung erhält man jedoch nicht, was auch durch die qualitative Beurteilung bestätigt wird. Auf das Konvergenzverhalten des Mehrgitterlösers hat die Darstellung der Geometrie als Oberflächentriangulierung unabhängig von der gewählten Feinheit keinen meßbaren Effekt.

5.2 Kugel-Benchmark

Beschreibung des Benchmarks Die Konfiguration des zweitens Benchmarks beinhaltet zwei Randkomponenten, einen äußeren Kanal und eine innere, zu umströmende Kugel. Auch hier ist es möglich, Vergleichsrechnungen mit einer analytischen Randbeschreibung vorzunehmen.

Aus der Zylinderkonfiguration wird die Größe des Kanals sowie die Ein- und Ausströmbedingungen unverändert übernommen. Die Kugel befindet sich mit dem Mittelpunkt $(0.45; 0.19; 0.2)$ an einer leicht asymmetrischen Position im Kanal. Ihr Radius beträgt 0.05.

Für diese Konfiguration existieren keine Referenzwerte. Mittels einiger hochgenauer Testrechnungen mit analytischer Randbeschreibung wurde ein eigener Referenzwert durch Mittelung und Extrapolation ermittelt. Es ergibt sich die folgende Referenzgröße:

$$c_d^{ref} = 1.496$$

Das Grobgitter für alle durchgeführten Testrechnungen ist in Abbildung 5.9 skizziert.

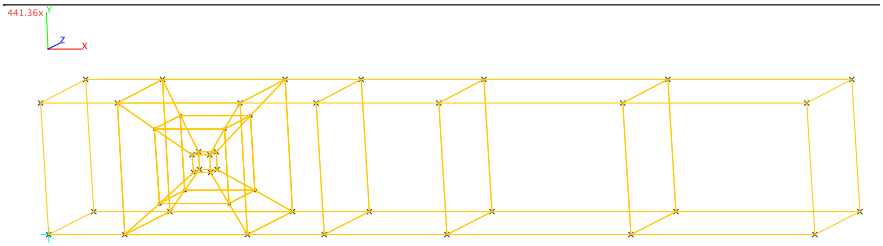


Abbildung 5.9: *Grobgitter für die Kugel-Konfiguration*

Basierend auf diesem Grobgitter ergeben sich die folgenden Problemgrößen:

Level	# Flächen	# Elemente	# Unbekannte
1	67	17	218
2	472	136	1552
3	3520	1088	11648
4	27136	8704	90112
5	212992	69632	708608
6	1687552	557056	5919712
7	13434880	4456448	44761088

Tabelle 5.10: Problemgröße der Kugel-Konfiguration auf verschiedenen Leveln

Mit dieser Konfiguration soll nochmals der Einfluß des Feinheitsgrades einer Oberflächentriangulierung auf die zu erreichende Genauigkeit in der Berechnung des Drag-Wertes und auf das Konvergenzverhalten des Mehrgitterlösers untersucht werden. Dazu werden die relativen Fehler bezogen auf obige analytisch berechnete Referenzgröße verglichen.

Mit Hilfe der CAD-Software `Blender3D` (s. `http://www.blender3d.org`) wurden Triangulierungen der Kugeloberfläche in den folgenden Auflösungen generiert:

Name	# Ringe	# Dreiecke
T16	$16 \cdot 16$	480
T32	$32 \cdot 32$	1984
T64	$64 \cdot 64$	7936
T96	$96 \cdot 96$	17620

Tabelle 5.11: Definition verschiedener Oberflächentriangulierungen für die Kugel-Konfiguration

Vergleich der Visualisierungen Die Abbildungen 5.10, 5.11 und 5.12 zeigen Beispiele für das qualitative Strömungsverhalten, jeweils durchgeführt mit einer analytischen Beschreibung, einer groben (T16) und einer feinen (T96) Diskretisierung der Kugel.

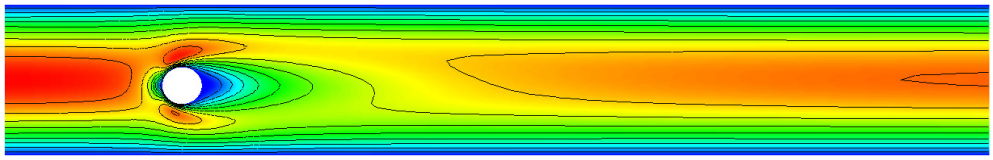


Abbildung 5.10: *Visualisierung der Norm des Geschwindigkeitsvektors mit Konturlinien in der zentralen Schnittebene durch den Strömungskanal, analytische Randbeschreibung*

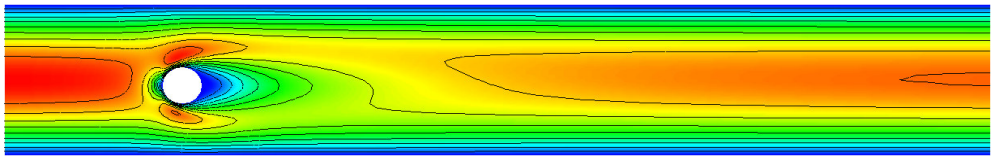


Abbildung 5.11: *Visualisierung der Norm des Geschwindigkeitsvektors mit Konturlinien in der zentralen Schnittebene durch den Strömungskanal, sehr grobe Oberflächentriangulierung (T16)*

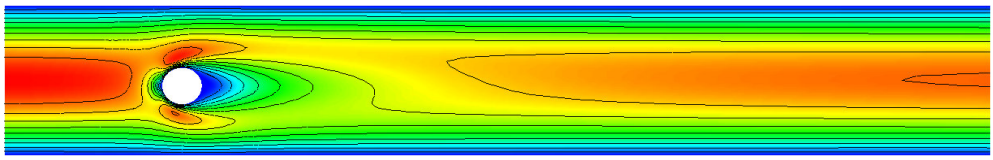


Abbildung 5.12: *Visualisierung der Norm des Geschwindigkeitsvektors mit Konturlinien in der zentralen Schnittebene durch den Strömungskanal, feine Oberflächentriangulierung (T96)*

Man erkennt minimale Unterschiede zwischen T16 (einer Beschreibung der Kugel, die schon bei geringer Zoomstufe nicht mehr rund aussieht) und der analytischen Randbeschreibung, zwischen T96 und der analytischen Randbeschreibung jedoch nicht mehr.

Analyse der Genauigkeit Die berechneten Drag-Werte für verschiedene Level sind in der folgenden Tabelle zusammengestellt:

Level	T16	T32	T64	T96
3	0.9111	0.9202	0.9226	0.9228
4	1.1749	1.1885	1.1919	1.1925
5	1.3684	1.3832	1.3873	1.3880
6	1.4438	1.4592	1.4633	1.4639
7	1.4648	1.4800	1.4840	1.4847

Tabelle 5.12: Berechnete Drag-Werte für verschiedene Kugel-Diskretisierungen auf verschiedenen Leveln.

Aus dieser Tabelle ergeben sich die folgenden relativen Fehler zu obigem Referenzwert $c_d^{ref} = 1.496$:

Level	T16	T32	T64	T96
3	39.1	38.49	38.33	38.32
4	21.46	20.55	20.33	20.29
5	8.53	7.54	7.27	7.22
6	3.49	2.46	2.19	2.15
7	2.09	1.07	0.8	0.76

Tabelle 5.13: Relative Fehler in Prozent zwischen den berechneten Werten und dem Referenzwert.

In Diagrammform erhält man aus dieser Tabelle:

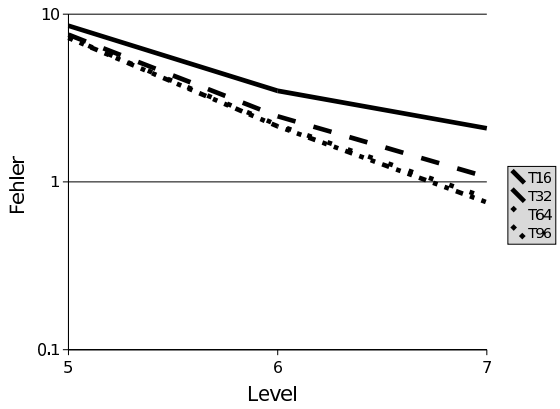


Abbildung 5.13: *Verlauf der relativen Fehler auf den drei feinsten Leveln, logarithmische Skalierung, Angabe in Prozent*

Folgende Tendenzen lassen sich feststellen: Der Verlauf der Fehlerkurven entspricht den Erwartungen. Vergleicht man die Werte für Level 6 bei der feinsten mit Level 7 in der gröbsten Triangulierung, so ist auch hier wieder der Gewinn eines (beinahe) kompletten Levels in der

Genauigkeit ersichtlich. Im Gegensatz zum Zylinderbenchmark ist die maximal erreichbare Genauigkeit durch die verwendeten Triangulierungen allerdings nicht mit einem Fehler im Promillebereich gekennzeichnet, sondern nur knapp unterhalb eines Prozents angesiedelt. Dies liegt daran, daß der Kugeläquator mit maximal 96 Dreieckskanten aufgelöst wird, während das Hexaedergitter auf dem höchsten Level 256 Kanten verwendet. Hier ist also der Randapproximationsfehler in allen verwendeten Triangulierungen erkennbar. Bis auf dieses Detail bestätigt die Tabelle die bei der Zylinderkonfiguration getroffenen Aussagen.

Konvergenzverhalten Für das Konvergenzverhalten des Mehrgitterlösers ergibt sich folgende Tabelle:

Level	analytisch	T96	T64	T32	T16
3	12/2.1	12/2.1	12/2.1	12/2.1	12/2.1
4	11/2.6	9/2.6	9/2.6	9/2.6	9/2.6
5	8/2.3	8/2.3	8/2.3	8/2.3	8/2.3
6	8/1.9	8/1.9	8/1.9	8/1.9	8/1.9
7	7/1.7	7/1.7	7/1.7	7/1.7	7/1.7

Tabelle 5.14: Durchgeführte nichlineare Iterationen, durchschnittliche Anzahl Mehrgitterschritte pro nichtlineare Iteration für verschieden feine Oberflächenbeschreibungen auf allen Leveln

Die konkreten Zahlwerte sind wie beim Zylinder wieder weitgehend identisch, die durchschnittlich nötigen Mehrgitterschritte pro nichtlinearer Iteration sind über alle Level hinweg vergleichbar. Auch diese Tabelle zeigt damit die Unabhängigkeit der Art der Oberflächenbeschreibung auf das Konvergenzverhalten und bestätigt so die entsprechenden Resultate beim Zylinderbenchmark.

Zusammenfassung Dieser Benchmark bestätigt die Ergebnisse aus der standardisierten Zylinderkonfiguration: Die maximal erreichbare Genauigkeit ist stark abhängig von der Feinheit der verwendeten Triangulierung. Bei zu groben Triangulierungen dominiert der Randapproximationsfehler schon ab relativ groben Leveln den gesamten Fehler. Vergleicht man die Ergebnisse jedoch visuell, so ist im qualitativen Verlauf der Strömung selbst bei nicht mehr rund aussehenden Kugelbeschreibungen kein Unterschied erkennbar. Auf das Konvergenzverhalten hat die Feinheit der Oberflächenbeschreibung keinen Einfluß.

5.3 Glättungsalgorithmen

In diesem Abschnitt soll untersucht werden, wie sich verschiedene Glättungsverfahren und Glättungsstrategien für innere Knoten auf die erreichbare Genauigkeit einer Simulation auswirken. Als Testkonfiguration dient der Zylinderbenchmark (s.o.), allerdings mit einer leicht veränderten Setzung der FEATFLOW – Parameter.
In einer mehrstündigen Gewaltrechnung werden auf Level 4 (504576 Unbekannte) die in Kapitel 2.3 vorgestellten Operatoren mit verschiedenen Iterationszahlen (3,5,10,50) und Gewichten (0 bis 0.9 in Schritten von 0.1) getestet. Insgesamt werden 116 verschiedene Fälle in einer

Rechnung untersucht. Eine solche Rechnung wird für zwei verschiedene Glättungsstrategien durchgeführt: Glättung des Grobgitters und Glättung des Feingitters (was aufgrund der Speicherarchitektur von FEATFLOW äquivalent ist mit einer simultanen Glättung auf allen Leveln). Diese Ergebnisse sind natürlich von der Geometrie, dem initial zur Verfügung stehenden Gitter und der verwendeten Software FEATFLOW abhängig. Als tendenzielle Aussagen – gerade was die Leistungsfähigkeit des Umbrella-Operators angeht – sind sie trotzdem brauchbar.
Verglichen werden in diesen Konfigurationen die relativen Fehler in der Drag-Berechnung zum Referenzwert von Braack und Richter (s.o.). Um Seiteneffekte durch die Verwendung einer Oberflächentriangulierung zu vermeiden, werden alle Tests mit einer analytischen Randbeschreibung durchgeführt. In allen Fällen ist das Konvergenzverhalten des Lösers nicht betroffen, d.h. es wird mit der gleichen Anzahl nichtlinearer Iterationen und Mehrgitterschritten und nur mit dem (minimalen) Zusatzaufwand der Glättung eine (hoffentlich) höhere Genauigkeit erzielt.

Glättung des Grobgitters Die folgende Tabelle stellt die besten vier Ergebnisse bei der Anwendung aller Glättungsoperatoren auf dem Grobgitter zusammen. Zum Vergleich ist auch die Lösung auf dem ungeglätteten Gitter angegeben.

Methode	# Iterationen	α	Drag	rel. Fehler in %
Original	n/a	n/a	6.1952	0.1651
Laplace-Jacobi	50	0.1	6.1850	0.0004
Laplace-Jacobi	5	0.9	6.1851	0.0010
Laplace-Jacobi	10	0.5	6.1849	0.0019
Laplace-Jacobi	5	0	6.1848	0.0037

Tabelle 5.15: Vergleich der besten Resultate bei einer Grobgitterglättung

Man erkennt zunächst eine deutliche Verminderung des relativen Fehlers: Es ist bemerkenswert, wie stark ein nur leicht verändertes Grobgitter (das danach vollkommen regulär weiter unterteilt wird) die relative Genauigkeit beeinflußt. Dieser Effekt ist jedoch nur bis zu diesem noch relativ groben Level so ausgeprägt, führt man die Rechnung weiter bis Level 6 durch, relativiert sich diese Tendenz. Der Grund für diese Tatsache liegt in tiefliegenden Konvergenzaussagen für den Mehrgitteralgorithmus (für eine Übersicht und weitere Referenzen siehe Kapitel 2.2), die das Verhalten des Lösers zurückführen auf „möglichst" rechtwinklige Gitter. Genau dies wird durch die Glättung erreicht.
Mit dieser Stategie arbeitet der einfache Laplace-Glätter offenbar am besten zusammen, was die allgemeine Beschreibung aus Kapitel 2.3 bestätigt: In Situationen, in denen der Laplace-Operator anwendbar ist, ist er konkurrenzlos schnell und liefert gute Ergebnisse.

Simultane Glättung aller Gitter Für die besten vier Ergebnisse bei der Glättung des feinsten Gitters (man beachte, daß durch die in FEATFLOW verwendeten Datenstrukturen hierbei alle gröberen Level simultan mitgeglättet werden) ergibt sich folgendes Bild:

Methode	# Iterationen	α	Drag	rel. Fehler in %
Original	n/a	n/a	6.1952	0.1651
Umbrella-Jacobi	3	0.4	6.1850	0.0007
Umbrella-Gauss-Seidel	5	0.2	6.1849	0.0016
Umbrella-Gauss-Seidel	3	0.3	6.1846	0.0069
Laplace-Jacobi	3	0.6	6.1845	0.0073

Tabelle 5.16: Vergleich der besten Resultate bei einer Glättung des feinsten Gitters

Diese Tabelle bestätigt die Vorteile des Umbrella-Operators im Vergleich zum Laplace-Operator, da bei dieser Strategie die Gefahr deutlich höher ist, bei einer einfachen Laplace-Glättung Elemente geringer Qualität zu erzeugen. Aus der Gesamttabelle (aus Gründen der Übersichtlichkeit nicht wiedergegeben) geht hervor, daß in dieser Testkonfiguration bis auf einige statistische Ausrutscher fast jede Form von Glättung bessere Ergebnisse liefert im Vergleich zum ungeglätteten Gitter.

Fazit Die Anwendung geeigneter Glättungsoperatoren auf ein Gitter kann bei geeigneter Konfiguration der Operatoren einen starken Effekt auf die Genauigkeit der Lösung haben. Die Bestimmung der Parameter für die Operatoren ist jedoch nicht einfach und wird hier durch eine Gewaltrechnung auf einem noch recht groben Level durchgeführt. Tendentiell verspricht – wie zu erwarten war – die Strategie mehr Erfolg, eine Glättung auf allen Leveln simultan durchzuführen, weil hier der Einfluß einer suboptimalen Parametersetzung für die Glätter weniger dominant ist. Die Experimente bestätigen die (nach meinem Kenntnisstand unveröffentlichte, aber nach Definition des Operators naheliegende) Eignung des Umbrella-Operators für die Glättung von Finite-Elemente-Diskretisierungen: In „einfachen Fällen" ist er der Laplace-Glättung unterlegen, in Fällen, in denen die Laplace-Glättung qualitativ schlechte oder sogar invertierte Elemente erzeugt, ist er deutlich überlegen. Folgt man der Strategie, auf allen Gitterebenen eine Glättung durchzuführen, so ist der Umbrella-Operator sehr robust. Es ist allerdings auch bemerkenswert, die sehr die Optimierung des Grobgitters in diesem Fall die erreichbare Genauigkeit beeinflußt. Sofern keine invertierten Elemente erzeugt werden, beschränkt sich der Mehraufwand auf die Glättung (eine reine Präprozessor-Aufgabe) und beeinflußt das Konvergenzverhalten des Mehrgitter-Lösers nicht.

5.4 Chevrolet Camaro im Windkanal

Beschreibung der Konfiguration Mit diesem Beispiel soll demonstriert werden, daß die in der Arbeit entwickelten Verfahren geeignet sind, Strömungssimulationen um komplexe dreidimensionale Objekte durchzuführen. Gleichzeitig dient es dazu, vorbereitend auf die im nächsten Abschnitt folgende Diskussion die Grenzen der in der Arbeit untersuchten Techniken aufzuzeigen.
Das verwendete Modell ist ein Chevrolet Camaro und stammt im Original aus der freien Modellgalerie unter `http://www.3dcafe.com`. Eine mit Blender [7] gerenderte Ansicht des Autos zeigt Abbildung 5.14.

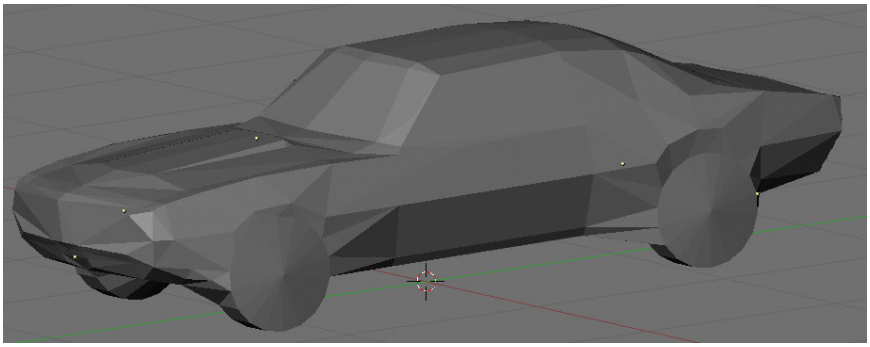


Abbildung 5.14: *Das verwendete Modell.*

Die Oberflächentriangulierung besteht aus insgesamt 616 Dreiecken. Man erkennt deutlich, daß das Modell nicht „perfekt” ist, sondern einige Artefakte enthält, insbesondere in der Umgebung der Räder. Diese Artefakte wurden absichtlich nicht aus dem Originalmodell entfernt, um die Robustheit der Randanpassung im Bezug auf kleine Fehler, Unstetigkeiten, steile Gradienten usw. im zugrundeliegenden Modell auf die Probe zu stellen.
Hier werden nur qualitative Untersuchungen der Umströmung vorgenommen, eine Diskussion der (prinzipiellen) Grenzen für eine quantitative Simulation folgt im nächsten Abschnitt. Das verwendete Grobgitter ist in einer Seitenansicht in Abbildung 5.15 dargestellt. Das Fahrzeug ist leicht asymmetrisch im Kanal positioniert.

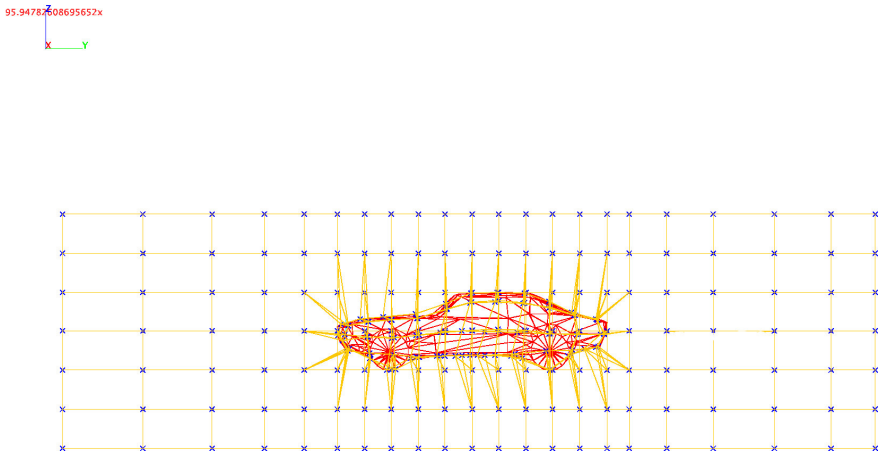


Abbildung 5.15: *Seitenansicht auf das verwendete Grobgitter*

Dieses Grobgitter führt auf die folgenden Problemgrößen:

Level	# Flächen	# Elemente	# Unbekannte
1	2480	716	9236
2	18512	5728	61264
3	142784	45824	474176
4	1121024	366592	3729664

Tabelle 5.17: Problemgröße der Konfiguration auf verschiedenen Leveln. Man beachte, daß von 716 Zellen im Grobgitter lediglich 88 zur Auflösung der Fahrzeuggeometrie verwendet werden, dies entspricht einem Anteil von 12%.

Der Einströmrand liegt auf der linken Seite des Kanals, der Ausströmrand auf der rechten. Weil die Generierung des Grobgitters dadurch wesentlich vereinfacht wurde, schwebt das Modell im Widerspruch zu physikalischen Grundregeln im Kanal. Desweiteren entsprechen die Reynoldszahl und die Einströmgeschwindigkeit in der realen Welt maximal einem Windhauch. Dies ist mit den Fähigkeiten des zugrundeliegenden FEATFLOW – Moduls zu erklären: Die Strömung muß stationär sein. Eine Implementierung in den instationären Löser, der auch Konfigurationen mit höheren Reynoldszahlen verarbeiten kann, ist vorgesehen. All diese physikalischen Unzulänglichkeiten sind mit Blick auf das Thema dieser Arbeit jedoch vollkommen legitim.

Auflösung des Modells auf verschiedenen Leveln Die Beschreibung dieses Experiments besteht aus kommentierten Serien von Visualisierungen des Modells und des Strömungsverlaufs. Die erste Serie von Visualisierungen (Abbildungen 5.16 bis 5.19) zeigt die immer genauere Auflösung der Geometrie mit steigendem Level. Die Oberfläche bzw. Kontur des Fahrzeugs wird folgendermaßen extrahiert: Durch die Wahl der Randbedingungen wird sichergestellt, daß auf der Oberfläche des Fahrzeugs die Geschwindigkeit null ist. Sie läßt sich also einfach über eine Isofläche extrahieren. Auf dieser Fläche wird der dort errechnete Druck dargestellt. Die Farbe rot repräsentiert hierbei einen hohen Druck, die Farbe blau einen niedrigen.

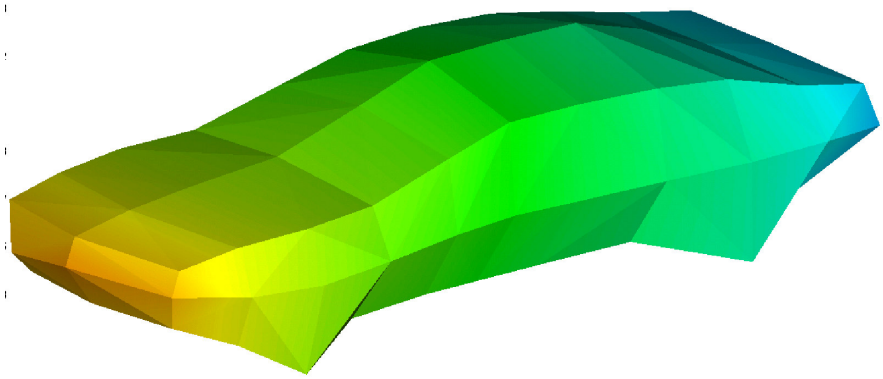


Abbildung 5.16: *Auflösung des Automodells auf Level 1*

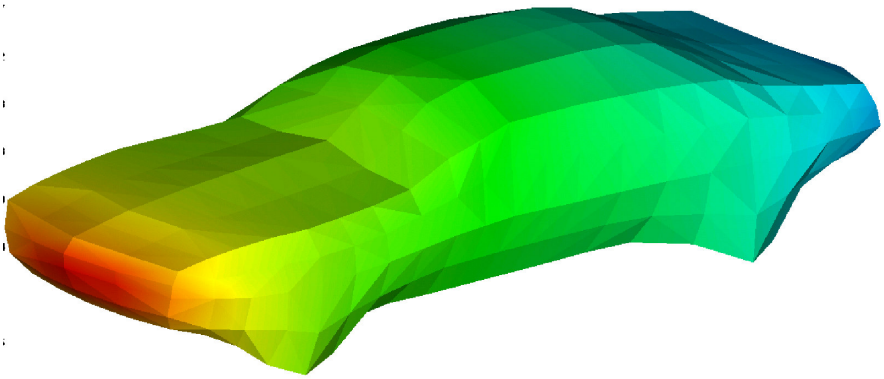


Abbildung 5.17: *Auflösung des Automodells auf Level 2*

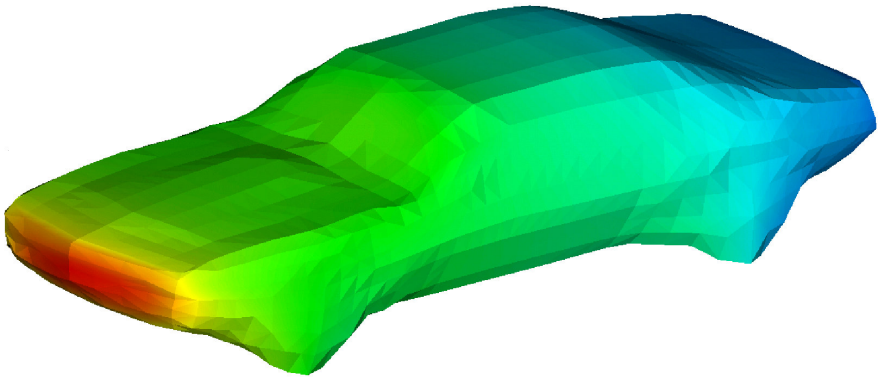


Abbildung 5.18: *Auflösung des Automodells auf Level 3*

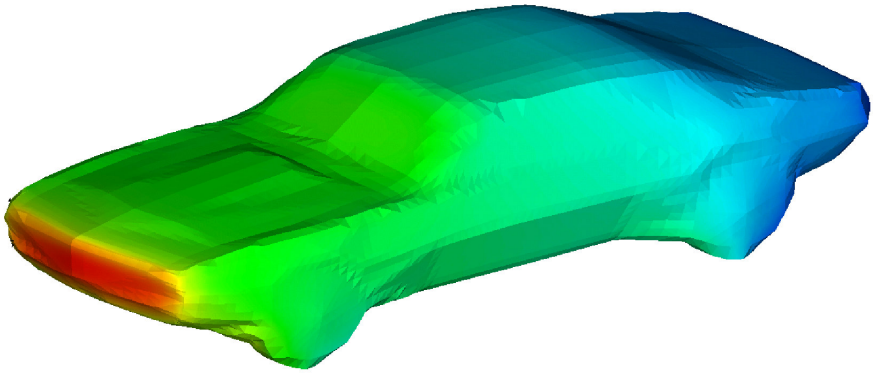


Abbildung 5.19: *Auflösung des Automodells auf Level 4*

Man beachte die insgesamt gute Übereinstimmung der extrahierten Kontur mit dem CAD-

Modell aus Abbildung 5.14. Einige der sichtbaren kleinen Artefakte, gerade um die Räder herum, resultieren aus dem angewendeten Trick der Oberflächenextraktion. Andere stammen aus nicht optimal geformten Elementen durch die Randanpassung, insbesondere aus nicht planaren Seitenflächen, die bei der Farbinterpolation während der Visualisierung Probleme bereiten. Weitere sind schon im Modell vorhanden (s.o.). Eine Überprüfung des Gitters gemäß der in Kapitel 4.4 definierten Qualitätskriterien ergibt jedoch keine außergewöhnlich stark deformierten oder ungültigen Elemente. ein weiteres Indiz ist die problemlose Konvergenz des Lösers.

Der Versuch, das Gitter noch einmal mehr auf Level 5 zu verfeinern, scheitert: An den Rädern und im Bereich des Kühlergrills der Fahrzeugkontur treten nach der Reprojektion invertierte (bzw. qualitativ miserable) Zellen auf, und zwar so viele, daß die in dieser Arbeit vorgestellten Glättungs- und Reprojektionstechniken dieses Problem nicht beheben können. Abbildung 5.20 versucht diesen Effekt in einer Ausschnittsvergrößerung der Randzellen im vorderen Drittel des Modells hervorzuheben. Mit den zur Verfügung stehenden Visualisierungswerkzeugen ist es mir nicht möglich, eine aussagekräftigere Darstellungen zu generieren.

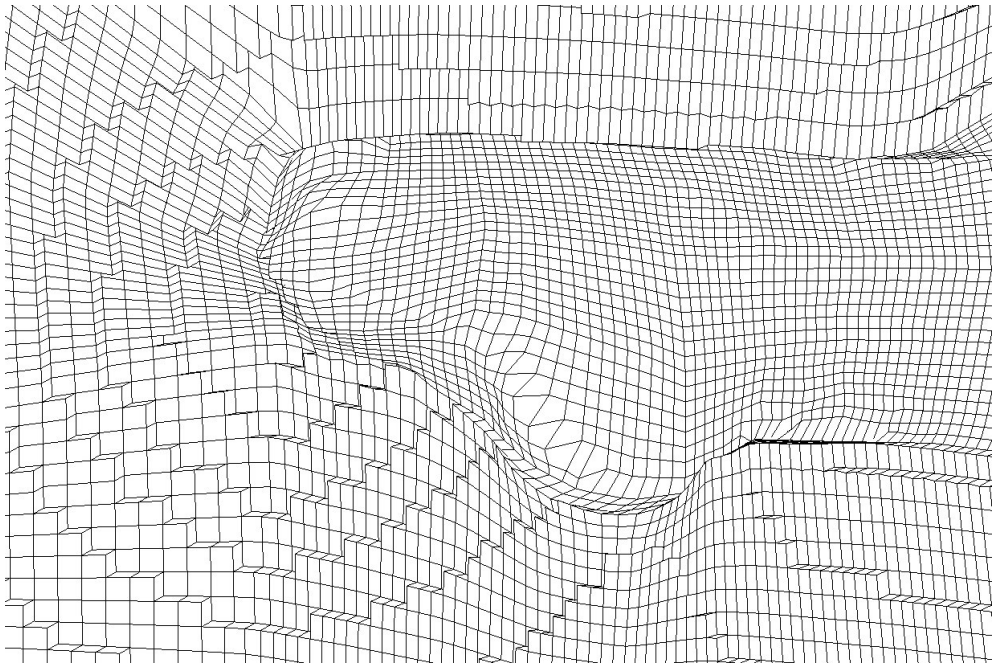


Abbildung 5.20: *Einfluß des Grobgitters auf die lokale Elementqualität in kritischen Regionen der Geometrie*

Man erkennt deutlich den Einfluß des Grobgitters (vgl. Abbildung 5.15): In der Frontpartie und um das Rad treten genau dort schwache Zellen auf, wo das Grobgitter die Geometrie nicht hinreichend genau auflöst. Ein weiteres Beispiel ist die Grenze zwischen „Radkappe" und „Reifen" in der Abbildung. Die reguläre Unterteilung kombiniert mit einer reinen Randanpassung ohne Gitterdeformation im Inneren des Gebiets ist offenbar nicht geeignet, solch dominante Details nachträglich (d.h. jenseits des Grobgitters) zu erfassen. Daß diese Defekte erst beim Übergang von Level 4 auf Level 5 auftreten, ist insofern verblüffend, weil die Visualisierung

der Fahrzeugkontur auf Level 4 (s. Abbildung 5.19) bereits sehr gut mit dem CAD-Modell übereinstimmt.

Um diese These bezüglich der Dominanz des Grobgitters zu bestätigen, wird die Simulation wiederholt, und zwar mit einem Automodell (genannt Modell B), das von den Rädern befreit wird. Die Konstruktion eines Grobgitters um diese bei obigem Modell kritischen Bereiche ist somit einfach möglich. Ebenso wird um die Frontpartie eine andere Gestalt des Grobgitters gewählt. Das Grobgitter enthält so nur noch 540 Elemente, seine Randapproximation kann Abbildung 5.21 entnommen werden.

Die Verfeinerung bis zu Level 5 klappt ohne Probleme, dies entspricht einem Speicherbedarf von 5 GB, was die Verfeinerung zu Level 6 ausschließt. Die Serie der nächsten 5 Abbildungen zeigt die Geometrieauflösung für dieses veränderte Modell:

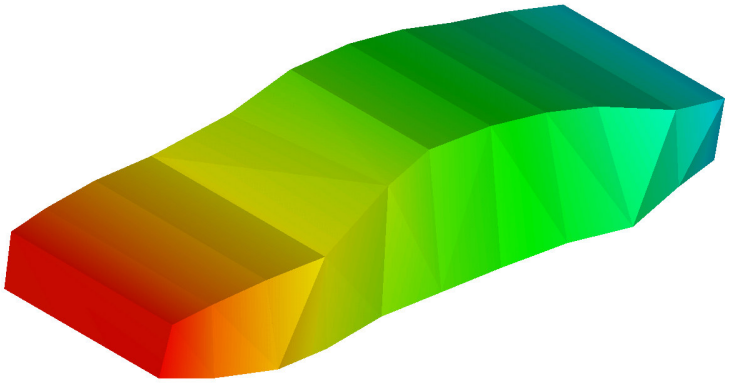


Abbildung 5.21: *Auflösung der Geometrie auf Level 1 für Modell B*

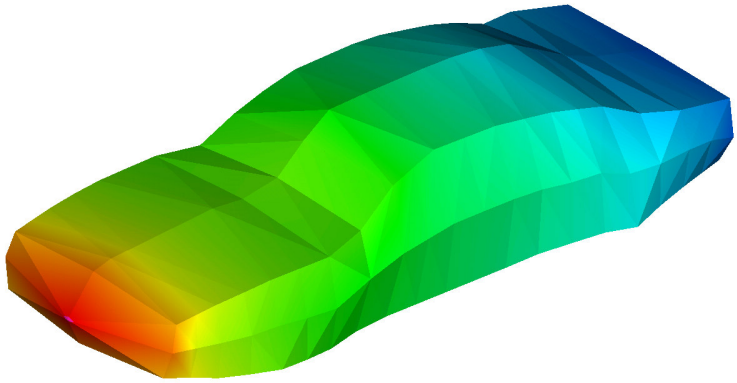


Abbildung 5.22: *Auflösung der Geometrie auf Level 2 für Modell B*

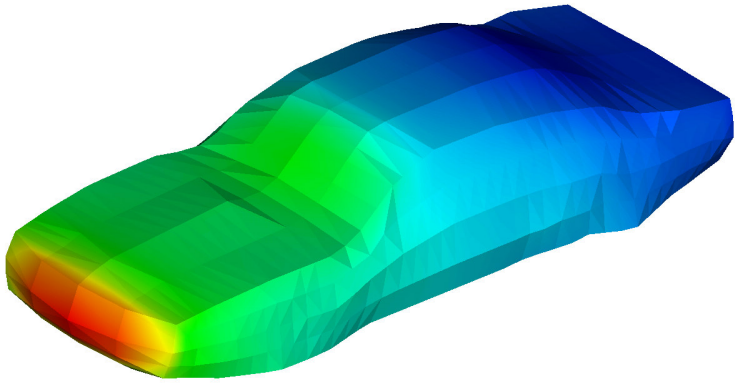


Abbildung 5.23: *Auflösung der Geometrie auf Level 3 für Modell B*

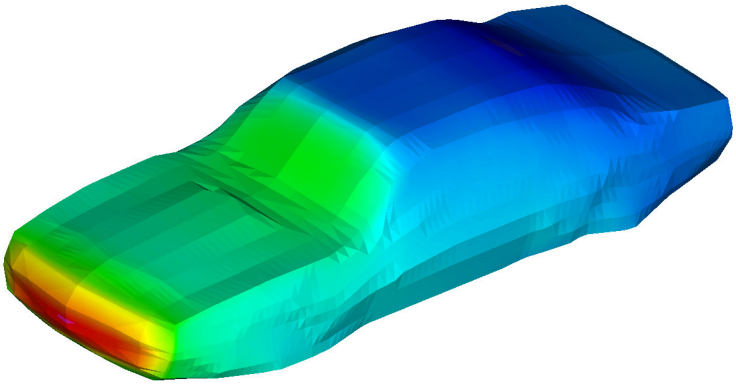


Abbildung 5.24: *Auflösung der Geometrie auf Level 4 für Modell B*

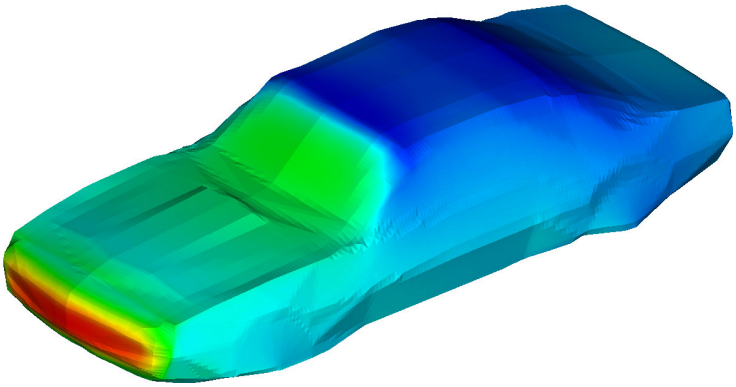


Abbildung 5.25: *Auflösung der Geometrie auf Level 5 für Modell B*

Diese Serie von Visualisierungen bestätigt die aufgestellte These: Die Güte der Randanpassung durch das Grobgitter ist für die in der Arbeit verwendeten streng geometrischen Randanpassungstechniken essentiell. Diese These impliziert allerdings eine aus praktischer Sicht zu starke Anforderung, denn wie gerade das erste Beispiel gezeigt hat, ist es nicht von vorne herein klar, welche Details einer Geometrie ab welchem Level die weitere Unterteilung unmöglich machen.

Weitere Visualisierungen Die nächste Serie zeigt einige Schnittebenen an verschiedene Stellen des Kanals. Dargestellt ist jeweils die Norm des Geschwindigkeitsvektors. Alle Visualisie-

rungen basieren auf der errechneten Lösung auf Level 4 für das Originalmodell mit Rädern.

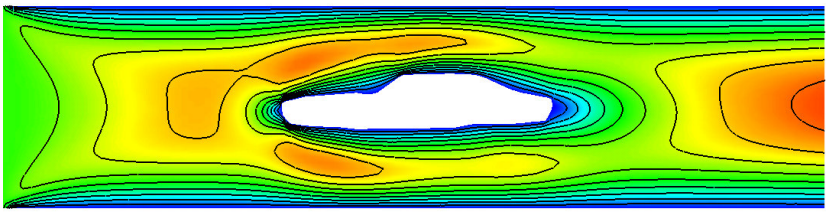


Abbildung 5.26: *Schnitt durch die Mitte des Fahrzeugs.*

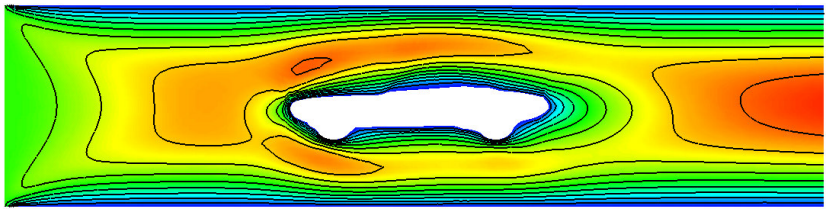


Abbildung 5.27: *Schnitt durch die linke Fahrzeugseite, mittig durch die Räder.*

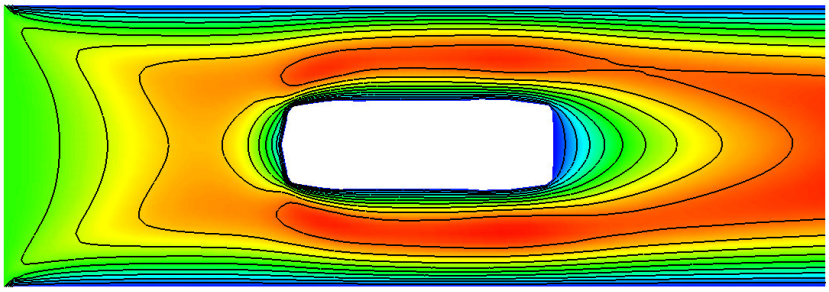


Abbildung 5.28: *Schnitt durch die Fahrzeugmitte.*

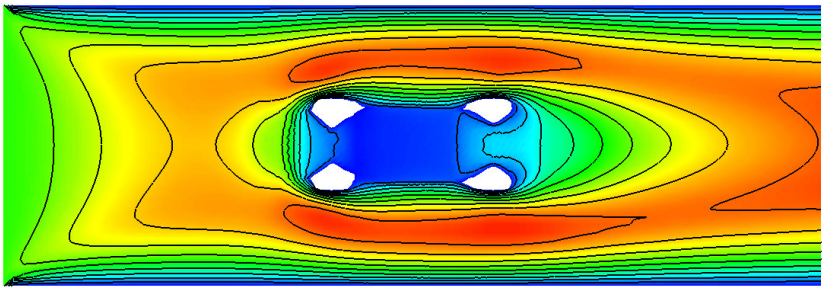


Abbildung 5.29: *Schnitt durch den Fahrzeugboden.*

Die letzte Darstellung zeigt die Druckverteilung im Kanal. Auch sie basiert auf der Simulation des Originalmodells auf Level 4.

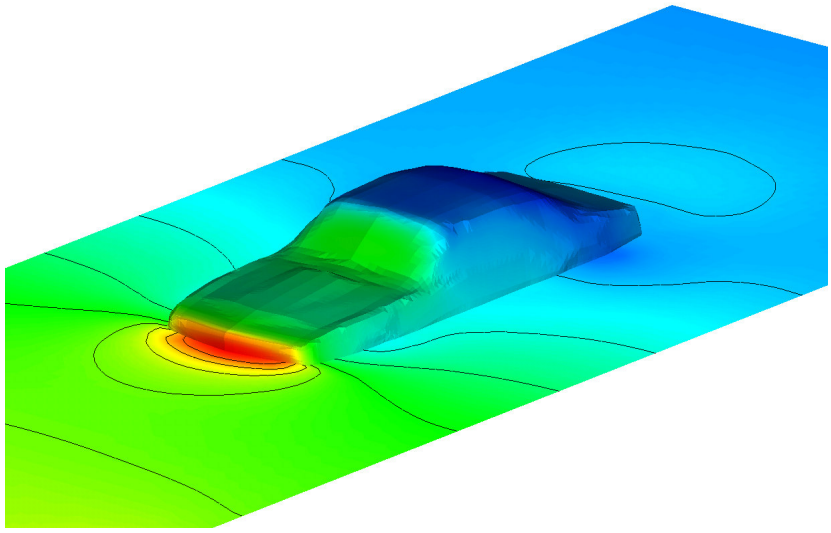


Abbildung 5.30: *Druckverteilung im Kanal.*

Analyse des Konvergenzverhaltens Das Konvergenzverhalten für beide Modelle faßt die folgende Tabelle zusammen:

Level	Modell A (mit Rädern)	Modell B (ohne Räder)
2	6/1.17	7/1.14
3	4/1.00	4/1.00
4	3/1.33	3/1.33
5	–	3/2.33

Tabelle 5.18: Anzahl nichtlinearer Iterationen gesamt, durchschnittliche Anzahl Mehrgitterschritte pro Iteration für beide Fahrzeuggeometrien auf allen Leveln.

Der Grund für die absolut gesehen geringe Anzahl an Iterationen liegt in den schwachen Abbruchkriterien im Vergleich zu den Benchmark-Konfigurationen, die auf quantitative Analysen ausgerichtet sind. Die Tabelle bestätigt die Robustheit des Mehrgitters. Vergleiche zum Einfluß der Triangulierung auf den Konvergenzprozeß können in Ermangelung einer analytischen Referenzlösung nicht gezogen werden, allerdings liefert der Vergleich des Konvergenzverhaltens beider Modelle ein Indiz dafür, daß auch in diesem Fall die Wahl der Triangulierung keinen Einfluß ausübt.

5.5 Diskussion der Ergebnisse

In diesem Abschnitt sollen die Ergebnisse der Arbeit sowie die Erfahrungen, die während der Bearbeitungszeit gewonnen wurden, kritisch diskutiert werden. Dazu werden zunächst die vier Teilgebiete, die sich aus der Aufgabenstellung ergeben, getrennt voneinander betrachtet. Einige Kritikpunkte werden identifiziert und mögliche Auswege aufgezeigt, die jedoch über das Thema dieser Arbeit hinausgehen. Abschließend werden prinzipielle Grenzen der untersuchten Methodik diskutiert.

5.5.1 Diskussion der verschiedenen Teilbereiche der Arbeit

Das Thema der Arbeit, *„Geometrische Projektionstechniken auf Oberflächentriangulierungen zur numerischen Strömungssimulation mit hierarchischen Mehrgitterverfahren”*, beinhaltet vier Teilaspekte. Positiv ist festzuhalten, daß die in der Arbeit untersuchten Verfahren für den Bereich der *hierarchischen Mehrgitterverfahren* durchaus gut geeignet sind: Im Vergleich zu analytischen Randbeschreibungen verhält sich der verwendete Mehrgitterlöser sehr robust, denn die Verwendung nichtanalytischer, approximativer Oberflächentriangulierungen als Randbeschreibung beeinflußt das Konvergenzverhalten nicht. Desweiteren sind *Oberflächentriangulierungen* in weiten Teilen des geometrischen Modellierens die Standardbeschreibung für Geometrieobjekte, trotz ihrer stückweise linearen Natur ermöglichen sie eine beliebig genaue Approximation jeder vorgegebenen Topologie und Glattheit. Außerdem ist es möglich, Oberflächentriangulierungen adaptiv an eine Geometrie anzupassen, zur Modellierung großer planarer Teilgebiete reichen wenige Dreiecke aus, während stark gekrümmte Details der Geometrie viele Dreiecke implizieren. Wird eine hinreichend feine Geometriebeschreibung verwendet, d.h. ist der Approximationsfehler gegenüber einem gegebenen Objekt durch die Verwendung eines Dreiecksnetzes kleiner als Randapproximationsfehler des Hexaedergitters, so kann durch die Verwendung von Oberflächentriangulierungen kein zusätzlicher Fehler oder eine (zu frühe) Dominanz des Randapproximationsfehlers auftreten. Diese theoretisch zu erwartende Aussage wird durch die untersuchten Benchmarkkonfigurationen experimentell verifiziert.
Der Nachteil, daß zur Approximation einer starken Krümmung sehr viele Dreiecke benötigt werden, daß das Modell also in komplexen Fällen aus mehreren Millionen Dreiecken bestehen kann, ist durch die Verwendung geeigneter Suchdatenstrukturen (insbesondere Octrees, vgl. Kapitel 3.3.2) in der Praxis kompensierbar.

Für den Aspekt der *numerischen Strömungssimulation* ist festzustellen, daß die verwendeten Techniken für eine qualitative Analyse von Umströmungen durch Oberflächentriangulierungen gegebener dreidimensionaler Geometrien gut geeignet sind: Schon recht grobe Triangulierungen liefern bei Visualisierungen des Strömungsverhaltens keine sichtbaren Unterschiede zu analytischen Randbeschreibungen. Dies ermöglicht die qualitative Analyse komplexer Objekte, die analytisch nicht darstellbar sind, was bisher mit der verwendeten Simulationssoftware nicht möglich ist. Für die ebenso wichtige quantitative Untersuchung von Strömungsvorgängen gilt allerdings, daß bei einer (abhängig vom gewünschten bzw. durch die Rechnerleistung vorgegebenen maximalen Level) zu grob gewählten Triangulierung der Fehler in der Randapproximation zu schnell dominiert: Es wird zwar eine korrekte Lösung berechnet, dies allerdings durch die schlechte Approximation auf einem „falschen” Gebiet. Weil interessante Kenngrößen

wie Auftrieb und Widerstandsbeiwert (von starker Relevanz beispielsweise in der Automobilindustrie) durch Integrale über die Oberfläche des Randobjektes definiert sind, kann eine zu grobe Approximation dieser Oberfläche schnell dazu führen, daß gewisse Zielvorgaben, beispielsweise die Reduktion des Diskretisierungsfehlers auf weniger als ein Prozent, nicht mehr eingehalten werden können. Auf der anderen Seite verhalten sich hinreichend feine Triangulierungen gutartig: Ist die Triangulierung fein genug konstruiert, so treten die (unvermeidbaren) Approximationsfehler nicht unterhalb des maximal möglichen Levels auf bzw. dominieren noch nicht.

Das vierte Teilgebiet der Arbeit, die Untersuchung von *Projektionstechniken*, zeigt die prinzipielle Eignung der vorgestellten Techniken zur Randanpassung in Anwendungen aus der Strömungssimulation. Es zeigt sich allerdings die folgende, aus praktischer Sicht relevante Grenze des streng *geometrischen* Zugangs: Für das fehlerfreie Funktionieren der Verfahren ist es schon bei (moderat) komplexen Geometrien notwendig, daß das Grobgitter „gut genug" an die Details der Geometrie angepaßt ist. Aus zwei Gründen ist dies zweifelhaft: Um bereits auf Grobgitterebene alle kleinformatigen Details einer beliebig komplexen Geometrie zu erfassen, sind sehr viele Zellen nötig. Dies karikiert die Verwendung eines Mehrgitterlösers: Der weitaus größte Teil der zur Berechnung nötigen Zeit wird in einem solchen Fall zur Lösung des Grobgitterproblems verwendet. Bei näherer Betrachtung der Vorgehensweise in kommerzieller Software ergibt sich: Sowohl TRUEGRID (`http://www.truegrid.com`) als auch ICEM CFD (`http://www.ansys.com`), mit denen im Rahmen der Arbeit experimentiert wurde, lassen die Projektion von relevanten Randpunkten auf das Ergebnis einer automatischen Segmentierung der Geometrie (gewissermaßen auf den relevanten Teil der Geometrie, nicht auf große planare Teilgebiete, auf denen die Projektion eindeutig ist) durch den Benutzer manuell durchführen. Lediglich Spezialfälle, in denen die Anwendung einer Schablonentechnik möglich ist, werden automatisch verarbeitet. Hinzu kommt, daß diese Pakete nur bedingt für den Einsatz in *top down*-Mehrgitterszenarien geeignet sind: Sie unterstützen die Generierung eines feinsten Gitters, das bei geeigneter Wahl der Parameter vom Benutzer manuell durch Zusammenfassen von Zellen vergröbert werden kann. Die Anwendung eines solchen *bottom up*-Ansatzes widerspricht der *top down*-Philosophie des verwendeten Mehrgitterlösers. Zusätzlich verbietet diese Technik eine reguläre Weiterverfeinerung: Stellt der Anwender fest, daß das ursprünglich generierte Gitter nicht fein genug ist, so muß entweder der Gittergenerierungsprozeß wiederholt oder eine eigene Randanpassung des manuell verfeinerten Gitters vorgenommen werden. Dies kann potentiell die gleichen Probleme verursachen, wie sie in dieser Arbeit experimentell bei der Umströmung des Automodells festgestellt wurden.

5.5.2 Mögliche Lösungsansätze für das Projektionsproblem

Eine erste einfache Idee, die Projektion stabiler zu gestalten, wurde schon in Kapitel 3.4.3 vorgestellt: Eine stückweise Projektion, die die Qualitätssicherung des Gitters nicht erst am Ende, sondern während der Projektion durchführt, bietet auf den ersten Blick die Möglichkeit, Fehler, d.h. durch die Projektion verursachte Elementinvertierungen frühzeitiger zu erkennen und möglicherweise besser korrigieren zu können. Der dort entworfene Projektionsalgorithmus implementiert gerade diese Idee. Die genauere Betrachtung der Fahrzeugumströmung aus dem vorherigen Abschnitt zeigt aber, daß dies bereits bei diesem moderat komplexen Modell fehlschlägt: Beim Übergang vom (funktionierenden) Level 4 zu Level 5 ist keine der in der

Arbeit entwickelten und untersuchten Projektions- und Gitterglättungstechniken geeignet, die dort festgestellte negative Dominanz des „schlechten" Grobgitters zu kompensieren. Aus qualitativer Sicht ist dies, wie dort argumentiert und mit Visualisierungen des Strömungsverhaltens belegt, kein Nachteil, aus quantitativer Sicht verhindert dieses Verhalten die Analyse vollständig. Andererseits zeigen die qualitativen Untersuchungen, daß für diesen Anwendungsfall kleine Details der zu umströmenden Geometrie gar nicht aufgelöst werden müssen, weil sie das globale Strömungsbild nicht beeinflussen. Eine relativierende Betrachtung dieses Phänomens und ein Vorschlag, wie dies in ein Mehrgitterszenario zu integrieren ist, wird im nächsten Abschnitt diskutiert.

Eine weitere Idee wurde in Abschnitt 2.3 vorgestellt: Zunächst werden Elementinvertierungen zugelassen, und mit Hilfe algebraischer *mesh untangling*-Verfahren [38, 39] wird dann versucht, das resultierende Gitter zu reparieren. Die Forschung in diesem Bereich steckt jedoch noch in den Kinderschuhen.

Die Adaption des *snake*-Konzeptes durch die *umbrella*-basierte Projektion auf die Randanpassung (s. Kap. 3.4.3) scheint aus zweierlei Gründen zu Problemen zu führen: Erstens muß das Konzept auf „trennbare" aktive Konturen erweitert werden, um auch Objekte mit Genus größer null behandeln zu können. Zum anderen geht es ja nicht nur darum, ein Oberflächengitter zu erstellen, vielmehr besteht das Oberflächennetz aus Seitenflächen von Hexaedern. In der Konsequenz muß sichergestellt werden, daß durch die Randanpassung keine sich schneidenden Zellen entstehen. Dieses Problem ist allen hier beschriebenen Lösungsvorschlägen inhärent und kann auch schon bei Objekten mit Genus null auftreten, wie die folgende Skizze in 2D verdeutlicht:

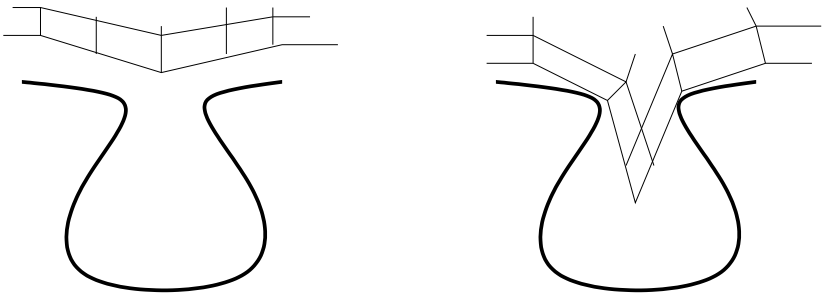


Abbildung 5.31: *Beispiel für inhärente Elementdurchdringungen. Links: „Seitenflächen-Snake" mit adjazenten Zellen vor der Anpassung. Rechts: Es entstehen sich durchdringende Elemente.*

Dieses Problem kann nicht auf der Basis einer Gitterglättung gelöst werden, stattdessen müssen fortgeschrittene Techniken zur Zellvereinigung und zur Gitterdeformation eingesetzt werden. Dieses Problem ist prinzipiell unabhängig vom aktuellen Level. Wie in 3D solche Zelldurchdringungen detektiert werden können, ist außerdem (bei nichtplanaren Hexaeder-Seitenflächen) ein nichttriviales Problem.

5.5.3 Vorschläge zur Relaxierung der Randanpassung

Erweitert man die Klasse der zur Verfügung stehenden Ansätze und verabschiedet sich von geometrischen Projektionstechniken, so bietet sich die Verwendung der im *multiresolution modelling* [18, 22, 21, 29, 41] untersuchten *parametrischen* Techniken an. Ein gegebenes Modell beliebig hoher Auflösung wird hier zu einem sogenannten Basisnetz (engl. *base mesh*) mit sehr wenigen Dreiecken oder Vierecken (engl. *patches*) segmentiert. Ausgehend von diesem Basisnetz werden nun Repräsentationen der Geometrie mit zunehmenden Detailstufen generiert (engl. *remeshing*), bis das (hochaufgelöste) Ausgangsmodell wieder erreicht ist. Abbildung 5.32 verdeutlicht das Vorgehen:

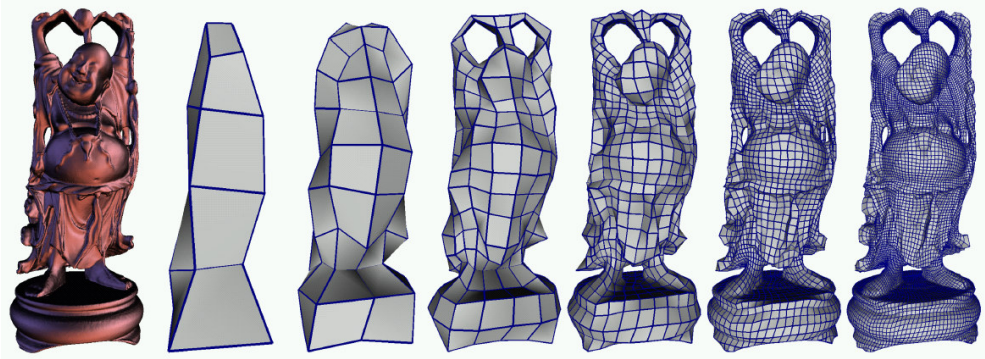


Abbildung 5.32: *Beispiel für ein* remeshing *nach [22]: Von links nach rechts: Ausgangsmodell, Basisnetz, zunehmende Detailstufen bei sich ändernder Topologie*

Der Vorteil dieses Vorgehens liegt darin, daß ausgehend vom Basisnetz für jede Teilfläche (in fortgeschrittenen Veröffentlichungen auch über die Grenzen von Teilflächen hinaus) eine isometrische Parametrisierung zur Verfügung steht. Während der *remeshing*-Phase wird typischerweise auch die sogenannte *subdivision connectivity* sichergestellt, d.h. es wird eine echte Teilmengenbeziehung zwischen Dreiecken auf verschiedenen Hierarchieebenen konstruiert. Dies vereinfacht die Parametrisierung deutlich. Die Integration in den Prozeß der Randanpassung ist also folgendermaßen möglich:

1. Berechne das Basisnetz ausgehend vom gegebenen Modell.

2. Generiere ein Hexaedergitter, identifiziere dabei die einzelnen *patches* des Basisnetzes mit Seitenflächen der Hexaeder, die auf dem Rand liegen. Stelle insbesondere sicher, daß Knoten im Basisnetz mit Knoten des Hexaedergitters korrespondieren.

3. Berechne lokale Parametrisierungen auf jeder Teilfläche.

4. In jedem Unterteilungsschritt des Mehrgitterverfahrens berechne zunächst eine nächstfeinere Geometrierepräsentation und bestimme die Koordinaten der neu hinzukommenden Knoten des Hexaedergitters durch bilineare Interpolation der zwei (bei Kantenmittelpunkten) oder vier (bei Flächenmittelpunkten) Parameterwerte der Knoten des gröberen Gitters.

Dieses Verfahren verspricht eine gute Randanpassung, es existieren jedoch prinzipielle Probleme. Der oben skizzierte Fall sich schneidender Hexaederzellen kann auch hier auftreten. Wie in Abbildung 5.32 können die verschiedenen Repräsentationen verschiedenen Geschlechts sein. Auch in diesem Fall müssen Hexaederzellen vereinigt werden. Setzt man voraus, daß alle Netze gleiche Topologie haben, so wird man wieder Grobgitter mit zu vielen Zellen generieren.

Eine weitere Möglichkeit ist die Relaxierung des kompletten Randbegriffs. Anstatt den Rand als fest gegeben zu betrachten, wird er als eine diffusive Schicht interpretiert. Dies läuft im wesentlichen auf eine nicht-diskrete Version der *multiresolution*-Idee hinaus. Wie dieser Vorschlag praktisch die dort skizzierten Probleme umgehen könnte, ist jedoch noch nicht klar.

5.5.4 Generelle praktische Grenzen

Ein bekanntes Problem bei dreidimensionalen Strömungssimulationen ist – wie weiter oben erläutert – die Tatsache, daß die Generierung eines „funktionierenden" Grobgitters auf Hexaederbasis bei komplexen Geometrien selbst mit sehr teurer kommerzieller Software viel manuelle Arbeit bedeutet. Außerdem will man soviel *a priori*-Wissen wie möglich einfließen lassen und beispielsweise das Gitter an die zu erwartenden Strömungen anpassen.

Praktisch interessante quantitative Untersuchungen in 3D sind unabhängig vom in dieser Arbeit untersuchten Randanpassungsproblem auf Supercomputer bzw. Clustercomputer beschränkt. Physikalisch interessante Effekte wie Verwirbelungen treten erst bei hohen Geschwindigkeiten auf. In einer Simulation muß demzufolge mit hohen Reynoldszahlen instationär gerechnet werden. Zu lösen ist also nicht nur ein Problem mit 10^8 oder mehr Unbekannten, sondern eines für jeden Zeitpunkt. In praktischen Szenarien sind 1000 und mehr Zeitpunkte keine Seltenheit. An massiv parallelen Lösern führt hier aus mehreren Gründen kein Weg vorbei: Selbst bei blindem Glauben an das Moore'sche Gesetz (Verdopplung Halbleiter-Packungsdichte alle 18 Monate) ist die direkte Simulation auf **einem** Computer innerhalb der nächsten 5-10 Jahre illusorisch, was die Rechenzeit betrifft. Der Speicherbedarf einer solchen Rechnung steigt schnell in den Bereich mehrerer TeraByte, was nur bei verteilten Systemen in den nächsten Jahren überhaupt zur Verfügung steht.

Ein anderer Ausweg ist die Verwendung von adaptiven Verfeinerungsstrategien. Eine einfache Idee ist in der folgenden Abbildung skizziert. Zwei Nachteile sind festzustellen: Die Gitterqualität ist ohne eine anschließende Glättung schlecht, und eine bessere Randauflösung wird nicht erzielt.

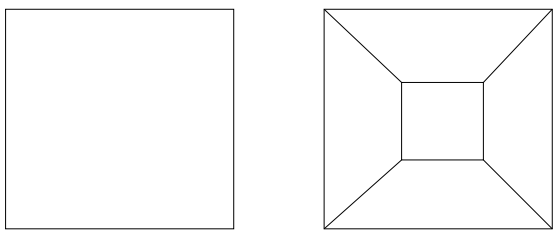


Abbildung 5.33: *Mögliche adaptive Verfeinerungsstrategie, geeignet für 2D und 3D*

Eine weiterer Vorschlag ist es, adaptiv neue Elemente zu generieren, die auf dem nächstgröberen Level keine „Eltern" haben. Ein Kriterium dazu wäre beispielsweise die Detektion starker Größenunterschiede nach einer ersten Projektion oder die Detektion invertierter Elemente. Abbildung 5.34 verdeutlicht die Idee.

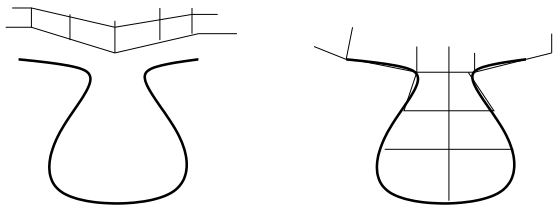


Abbildung 5.34: *Alternative Verfeinerungsstrategie im Kontrast zu 5.31 unter Erzeugung neuer Elemente ohne „Eltern", geeignet für 2D und 3D*

Die fehlenden Hierarchiebeziehungen stehen jedoch im Widerspruch zum Mehrgitteralgorithmus. Mittels einer nachträglichen Vergröberung, d.h. durch das rekursive Zusammenfassen von Zellen, könnte die Gitterhierarchie wiederhergestellt werden. Im Umkehrschluß kann dies jedoch ein initial aus wenigen Zellen bestehendes Grobgitter stark aufblähen. Anstatt die Gitterhierarchie nachträglich zu verändern, können auch Methoden zur *domain decomposition* [50] angewendet werden, mit denen die Verarbeitung solcher Teilgebiete außerhalb der Mehrgitteriteration durchgeführt werden kann.

5.6 Danksagung

Ich möchte mich abschließend bei allen Personen bedanken, die zu dieser Arbeit beigetragen haben: An erster Stelle stehen hier natürlich meine beiden Betreuer, Prof. Dr. Turek und Prof. Dr. Müller, die sich immer Zeit für meine Fragen, Probleme und meinen Diskussionsbedarf genommen haben.
Ohne die Mitarbeiter am LS3, insbesondere Dr. Jaroslav Hron, wäre gerade die Implementierung zur Arbeit nicht in endlicher Zeit möglich gewesen. Vielen Dank!

Literaturverzeichnis

[1] ABRAMOWSKI, S. und MÜLLER, H.: *Geometrisches Modellieren*. BI-Wissenschaftsverlag, 1992.

[2] ANDERSON, J. D.: *Computational Fluid Dynamics: The Basics with Applications*. McGraw Hill, 6 Aufl., 1995. ISBN 0070016852.

[3] AUTODESC INC.: *DXF Reference 2004*. Manual, Autodesc Inc., Feb. 2003. `http://www.autodesc.com`.

[4] BAREQUET und KUMAR: *Repairing CAD Models*. In: YAGEL und HAGEN (Hrsg.): *IEEE Visualization '97*, `http://cs.jhu.edu/~barquet`, 1997. IEEE.

[5] BENKO, MARTIN und VARADY: *Algorithms for Reverse Engeneering Boundary Representation Models*. Computer Aided Design, 33(11):839–851, 2001.

[6] BLACKER: *Meeting the Challenge for Automated Conformal Hexahedral Meshing*. In: *Proceedings of the 11th International Roundtable, Sandia National Laboratories*, `http://img.sandia.gov`, 2000.

[7] BLENDER: *Blender 3D*. Techn. Ber., Blender, 2003. `http://www.blender3d.org`.

[8] BRAACK, M. und RICHTER, T.: *Solutions of 3D Navier-Stokes benchmark problems with adaptive finite elements*. Preprint, Institute of Applied Mathematics, University of Heidelberg, INF 294, 69120 Heidelberg, Germany, Apr. 2004. Submitted to Elsevier Science.

[9] BRAESS, D.: *Finite Elemente*. Springer-Verlag Berlin-Heidelberg New York, 3 Aufl., 2003. 3., korrigierte und ergänzte Auflage, ISBN 3-540-00122-0.

[10] CAMPAGNA, KOBBELT und SEIDEL: *Directed Edges - A Scalable Representation for Triangle Meshes*. Journal of Graphics Tools, 3(4):1–12, 1998.

[11] CAREY, G. F.: *Computational Grids: Generation, Adaptation, and Solution Strategies*. Taylor und Francis, 1997.

[12] CHANG: *A Survey of Geometric Data Structures for Ray Tracing*. Doktorarbeit, Department of Computer and Information Science, Polytechnic University, Brooklyn, New York, 2001.

[13] COHEN, L. und COHEN, I.: *Finite element methods for active contour models and ballons for 2-d and 3-d images*. IEEE Transactions on Pattern Analysis and Machine Intelligence, S. 1131–1147, 1993.

[14] DAVIS: *A column pre-ordering strategy for the unsymmetric-pattern multifrontal method*. Technical Report TR-03-006, -, 2003. Submitted to ACM Trans. Math. Software.

[15] DONEA, J. und HUERTA, A.: *Finite Element Methods for Flow Problems*. Wiley, Mai 2003. ISBN 0-471-49666-9.

[16] FARIN, G.E.: *Kurven und Flächen im Computer Aided Geometric Design*. Vieweg, 1994. ISBN 3528165421,.

[17] FELLNER, W.D.: *Computer Grafik*. Teubner, 1992. 2.Auflage, ISBN 3411151226 .

[18] FLOATER, M.S.: *Parameterization and smooth approximation of surface triangulations*. Computer Aided Geometric Design, 14(3):231–250, 1997.

[19] FOLEY, J. D., VAN DAM, A., FEINER, S. und HUGHES: *Computer Graphics- Principles and Practice (2nd Edition)*. Addison-Wesley, 1996.

[20] GROSSMANN, CH. und ROOS, H.-G.: *Numerik partieller Differentialgleichungen*. Teubner, 1994. 2.Auflage .

[21] GUSKOV, VIDIMCE, SWELDENS und SCHRÖDER: *Normal Meshes*. In: *Proceedings of the 27th annual conference on computer graphics and interactive techniques*, Bd. 27. ACM Press, Addison-Wesley Publishing Co., 2000. ISBN 1-58113-208-5.

[22] GUSKOV, I., KHODAKOVSKY, A., SCHRÖDER, P. und SWELDENS, W.: *Hybrid Meshes: Multiresolution using regular and irregular refinement*. Proceedings of SOCG 2002, 2002.

[23] HACKBUSCH: *Iterative Lösung großer schwachbesetzer Gleichungssysteme*. Teubner Verlag, März 2002. ISBN 351912372X.

[24] HACKBUSCH, W.: *Multi-Grid Methods and Applications*. Springer, 1985. Berlin-Heidelberg, ISBN 3540127615.

[25] HACKBUSCH, W.: *Theorie und Numerik elliptischer Differentialgleichungen, 2. Auflage*. Stuttgart: Teubner, 1996.

[26] HERMANSSON und HANSBO: *A variable diffusion method for mesh smoothing*. Preprint, Chalmers Finite Element Center, Chalmers University of Technology, Göteborg Sweden, 2002.

[27] HIRSCH, C.: *Numerical computation of internal and external flows, Band 1*. Wiley, 2000. ISBN 0-471-92385-0.

[28] HIRSCH, C.: *Numerical computation of internal and external flows, Band 2*. Wiley, 2000. ISBN 0-471-92385-0.

[29] HORMANN: *Theory and Applications of Parametrizing Triangulations*. Doktorarbeit, Universitaet Erlangen, Institut für Informatik, 2001.

[30] HOSCHEK, J. und LASSER, D.: *Grundlagen der geometrischen Datenverarbeitung*. Teubner, 2002. ISBN 3528165421 .

[31] HYSOM, D. und POTHEN, A.: *Efficient Parallel Computation of ILU(k) Preconditioners*. Preprint VA 23681-2199, NASA Langley Research Center Hampton, Institute for Computer Applications in Science and Engineering Mail Stop 132C, Mai 2000. `http://www.icase.edu/library/reports/rdp/2000/2000-23RDP.tex.refer.html`.

[32] HYSOM, D. und POTHEN, A.: *Level-based Incomplete LU Factorization: Graph Model and Algorithms*. Preprint UCRL-JC-150789, U.S. Department of Energy, Nov. 2002. `http://www.cs.odu.edu/~pothen/papers.html;to` appear in: SIAM Journal On Matrix Analysis and Applications.

[33] JOHN, V.: *Higher order finite element methods and multigrid solvers in a benchmark problem for the 3D Navier-Stokes equations*. Int. J. Numer. Meth. Fluids, 40:775–798, 2002. DOI 10.1002/fld.377.

[34] KASS, M., WITKIN, A. und TERZOPOULUS, D.: *Snakes: Active contour models*. Int. J. Computer Vision, S. 321–331, 1988.

[35] KELLY, S.: *Element Shape Testing*, Bd. 9 th edition. Eigenpublikation von AnSyS Inc., 1998. chapter 13 in Ansys Theory Reference; www.ansys.com .

[36] KILIAN, S.: *ScaRC als verallgemeinerter Mehrgitter- und Gebietszerlegungsansatz für parallele Rechnerplattformen*. Logos Verlag, Berlin, 2002. ISBN 3-8325-0092-8.

[37] KNUPP: *Martix norms & condition numbers*. In: *Proceedings of the 8th International Meshing Roundtable*, `http://imr.sandia.gov`, 1999.

[38] KNUPP: *Algebraic mesh quality metrics*. SIAM J. Sci. Comput., 23(1):193–218, 2001.

[39] KNUPP: *Hexahedral Mesh Untangling & Algebraic Mesh Quality Metrics*. In: *Proceedings of the 9th International Meshing Roundtable*, Bd. 17 d. Reihe *Engineering with Computers*, `http://imr.sandia.gov`, Okt. 2001. Springer, London.

[40] KOBBELT, L.: *Iterative Erzeugung glatter Interpolanten*. Doktorarbeit, Universität Karlsruhe, Verlag Shaker, Aachen, 1995, Dez. 1994.

[41] KOBBELT, L. P., VORSATZ, J., LABSIK, U. und SEIDEL, H.-P.: *A shrink Wrapping Approach to Remeshing Polygonal Surfaces*. In: BRUNET, P. und SCOPIGNO, R. (Hrsg.): *EUROGRAPHICS '99*, Bd. 18, 1999.

[42] LOHNER, R., MORGAN, K. und ZIENKIEWICZ, O. C. : *Accuracy estimates and adaptive refinements in finite element computations*, Kap. Adaptive grid refinement for the compressible Euler equations. Wiley, 1986.

[43] LUBER und HASTREITER: *Script of Lecture Scientific Visualization*. Universität Erlagen, 1999.

[44] LÜRIG, C., KOBBELT, L. und ERTL, T.: *Deformable surfaces for feature based indirect finite volume rendering*. In: *Computer Graphics International*, IEEE Proceedings, 1998.

[45] MEISTER, A.: *Numerik linearer Gleichungssysteme*. Vieweg Verlag, 1999. ISBN 3528031352.

[46] MUKHERJEE: *A hybrid, variational 3D smoother for orphaned shell meshes*. In: *Proceedings of the 11th International Roundtable, Sandia National Laboratories*, `http://imr.sandia.gov`, 2002.

[47] NEUENSCHWANDER, W. M.: *Elastic Deformable Contour and Surface Models for 2-D and 3-D Image Segmentation*. Doktorarbeit, Eidgenössische Techn. Hochschule Zürich, 1995.

[48] OWEN, STATEN, CANANN und SAIGAL: *Advancing Front Quadrilateral Meshing Using Triangle Transformations*. In: *Proceedings of the 7th International Meshing Roundtable*, Bd. 7, `http://ims.sandia.gov`, 1998.

[49] PINNAU, R.: *Höhere Numerik II - Theorie und Numerik elliptischer Differentialgleichungen*. Vorlesungsskript, TU Darmstadt, 2001.

[50] QUARTERONI, A. und VALLI, A.: *Domain decomposition methods for partial differential equations*. Oxford, Clarendon Press, 1999. ISBN 0-19-850178-1.

[51] REVELLES, URENA und LASTRA : *An efficient parametric algorithm for Octree Traversal*. In: *The 8th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media 2000*, `http://wscg.zcu.cz/wscg2000/wscg2000.htm`, Feb. 2000.

[52] SCHÄFER, M. und TUREK, S.: *Benchmark Computations of Laminar Flow Around a Cylinder*. Techn. Ber., Universität Heidelberg, 1996. SFB 359.

[53] SCHWARZ, H.R.: *Numerische Mathematik*. Teubner, 1997.

[54] TAKANASHI, I., MURAKI, S., DOI, A. und KAUFMAN, A.: *3D Active Net for Volume Extraction*. In: *SPIE Electronic Imaging '98*, Nr. 3298 in *Proceedings SPIE*, 1998.

[55] TERZOPOULUS, D.: *Regularization of inverse visual problems involving discontinuities*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1986.

[56] TERZOPOULUS, D., WITKIN, A. und KASS, M.: *Symmetrie-seeking models and 3d object construction*. Int. J. Computer Vision, S. 211–221, 1987.

[57] TUREK, S.: *Efficient solvers for incompressible flow problems: An algorithmic and computational approach*. Springer, 1999.

[58] TUREK, S.: *Numerische Methoden für Partielle Differentialgleichungen*. Vorlesungsskript, 2001.

[59] TUREK, S. und BECKER, CH.: *FEATFLOW - Finite element software for the incompressible Navier–Stokes equations*. User Manual, Universität Dortmund, 1999.

[60] TUREK, S., HARIG, J und SCHREIBER, P.: *FEAT3D - Finite element analysis tools in 3 dimensions*. User Manual, Universität Heidelberg, 1994.

[61] VAN DER VORST, H. A.: *Bi-CGStab: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*. SIAM J. Sci. Stat. Comp., 13(2):631–644, 1992.

[62] WESSELING, P.: *An Introduction to Multigrid Methods*. John Wiley & Sons Ltd., 1992. `http://www.mgnet.org/mgnet-books-wesseling.html` .

[63] WESSELING, P.: *Principles of computational fluid dynamics*. Springer Berlin, 2001. ISBN 3-540-67853-0.

[64] WINSLOW, A.: *Numerical solution of the quasi-linear Poisson-equation in a nonuniform triangle mesh*. J. Comp. Phys., 1:149–172, 1967.

[65] YSERENTANT: *Old and New Convergence Proofs for Multigrid Methods*. Acta Numerica, S. 1–44, 1992.

Index