

Ein neuer Netzbewegungslöser für große Verformungen basierend auf der Centroidal-Voronoi-Tessellierung

vorgelegt von

M.Sc.

Witalij Wambold

geb. in Wladimir

Von der Fakultät II - Mathematik und Naturwissenschaften der Technischen
Universität Berlin zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften

- Dr. rer. nat. -

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr. rer. nat. Ulrich Pinkall
Berichter/Gutachter:	Prof. Dr. rer. nat. Günter Bärwolff
Berichter/Gutachter:	Prof. Dr. rer. nat. Volker Mehrmann
Berichter/Gutachter:	Dr. rer. nat. Andreas Gitt-Gehrke

Tag der wissenschaftlichen Aussprache: 01. August 2016

Berlin 2017

Zusammenfassung

Die vorliegende Arbeit beschäftigt sich mit einem Ansatz zur Simulation der Strömungen in zeitabhängigen Gebieten. Der integrale Teil der Arbeit besteht dabei in der Entwicklung einer Methode zur Bewegung von sogenannten Centroidal-Voronoi-Diagrammen zusammen mit ihren Definitionsgebieten. Ein großer Vorteil des entwickelten Algorithmus besteht in ständiger Erhaltung der Gitterqualität bei unbeschränkt großen Verformungen des darunterliegenden Rechengebietes. Ein weiterer Vorteil dieser Methode besteht darin, dass die Anzahl der Volumenelemente während der ganzen transienten Simulation konstant bleibt, womit Interpolation solcher Strömungskenngrößen wie Druck und Geschwindigkeit vermieden wird.

Zunächst werden Grundlagen und vier klassische Ansätze zur Konstruktion der Voronoi-Diagramme von nicht abgeschlossenen bzw. offenen Gebieten beschrieben. Danach wird ein im Rahmen dieser Arbeit entwickelter Algorithmus zur schnellen Aktualisierung des Voronoi-Diagramms eines Quaders vorgestellt, der das Rechengebiet komplett umhüllt. Ein weiterer Teil der Arbeit befasst sich mit Berücksichtigung nicht konvexer Strukturen des vordefinierten Gebietes. Wir verwenden eine zusätzliche triangulierte Fläche als Repräsentation des Gebietsrandes. Solche kritische Strukturen wie scharfe Kanten bzw. spitze Ecken werden in dem endgültigen Volumengitter abgebildet.

Im nachfolgenden und größten Teil dieser Arbeit wird gezeigt, wie die Centroidal-Voronoi-Diagramme dafür genutzt werden können, um sowohl eine Art Gitterbewegung herzustellen, als auch die Qualität der bewegenden Gitter aufrecht zu erhalten. Die Idee dieses Ansatz ist, dass man die zeitlich aufgelöste Simulation mit einem Centroidal-Voronoi-Gitter startet und nach jeder Randdeformation die Centroidal-Voronoi-Eigenschaft des darunterliegenden Gitters wiederherstellt. Danach werden die Algorithmen beschrieben, die zur Konstruktion der Centroidal-Voronoi-Diagramme verwendet werden. In diesem Kontext wird auch ein im Rahmen dieser Arbeit entstandener Algorithmus vorgestellt, der sowohl effizienter als die in [61] beschriebene Methode ist, als auch eine wesentlich höhere Konvergenzrate gegenüber dem klassischen Lloyd-Algorithmus hat. Mittels experimenteller Untersuchungen werden in diesem Teil der notwendige Genauigkeitsgrad der zur Approximation der Volumenintegrale verwendeten Quadraturformeln bestimmt und die Konvergenzrate des Lloyd-Algorithmus untersucht.

Ein weiterer Teil der Arbeit ist der Verwendbarkeit des entwickelten Algorithmus mit der Finite-Volume-Methode gewidmet. Hierdurch werden die sogenannten Navier-Stokes-Gleichungen in ALE-Formulierung sowie die geometrischen Erhaltungsgleichungen hergeleitet. Danach wird ein Algorithmus angeboten, mit dessen Hilfe die sogenannten Netzflüsse (Engl. *mesh fluxes*) so berechnet werden, dass der geometrische Erhaltungssatz erfüllt ist. Denn aufgrund der Neuerzeugung der Flächenelemente ist es unmöglich mithilfe der Verschiebung der Knoten dieser Elemente die Netzflüsse zu bestimmen, wie es auch bei den klassischen Ansätzen gemacht wird (siehe [28]).

Im Anschluss wird der Gesamtautomat anhand einer relativ komplizierten Geometrie des Propellers in einem inkompressiblen Fluid validiert.

Abstract

The present work deals with an approach concerning the simulation of flows in time-dependent domains. The integral part of the work consists in developing of a method for motion of the so-called centroidal Voronoi diagrams together with their definition areas. A great advantage of the developed algorithm is constantly maintaining the grid quality in the case of unlimited deformations of the underlying computational domain. Another advantage of this method is that the number of cells remains constant during the entire transient simulation, whereby interpolation of such flow fields as pressure and velocity is avoided.

First, the theoretical background and four classical approaches for construction of Voronoi diagrams of open domains are described. Afterwards, an algorithm developed in this work for fast update of a Voronoi diagram of a cuboid completely enveloping the computational domain will be presented. Another part of the work is concerned with non-convex structures of the predefined domain. We use an additional triangulated surface as a representation of the boundary. Such critical structures such as sharp edges or pointed corners are represented in the final volume mesh.

In the following and the largest part of this work is shown, how the centroidal Voronoi diagrams can be used, in order to move the internal mesh and to maintain the high quality of the moved mesh. The idea of this approach is that the transient simulation is started with a centroidal Voronoi mesh and the centroidal Voronoi property of the underlying mesh is restored after each deformation of the domain boundary. Thereafter, the algorithms are described, which are used for construction of the centroidal Voronoi diagrams. In this context an algorithm arising within the scope of this work is presented, which is more efficient than the in [61] described method, and has a significantly higher convergence rate than the classic Lloyd's method. By means of experimental studies the necessary degree of precision of quadrature rules is determined, which is used for approximation of the volume integrals. Furthermore the convergence rate of the Lloyd algorithm is examined.

Another part of the work is dedicated to the applicability of the developed algorithm to the finite volume method. As a result, the so-called Navier-Stokes equations in ALE formulation as well as the geometric conservation laws are derived. Then an algorithm is offered allowing such calculation of the mesh fluxes so that the geometric conservation law is satisfied. Because of the regeneration of the faces, it is impossible to use the displacement of the face nodes, in order to compute the mesh fluxes (see [28]).

Finally, the whole algorithm will be validated using a relatively complicated geometry of the propeller immersed in an incompressible flow.

Inhaltsverzeichnis

1 Einleitung	1
1.1 Motivation	1
1.2 Stand der Wissenschaft	2
1.3 Wissenschaftlicher Beitrag	3
1.4 Kapitelübersicht	5
2 Voronoi-Diagramme nicht konvexer Gebiete	7
2.1 Theoretische Hintergründe	7
2.1.1 Voronoi-Diagramme	7
2.1.2 Delaunay-Triangulierung, Dualität und Entartung	9
2.1.3 Klassische Methoden zur Voronoi- bzw. Delaunay-Zerlegung	10
2.2 Voronoi-Zerlegung des einhüllenden Quaders	12
2.2.1 Verschneiden mit Bisektoren	13
2.2.2 Bestimmung der nächsten Nachbarn	14
2.3 Verschneiden mit dem Randgitter	16
2.3.1 Überblick über den Algorithmus	17
2.3.2 Feature-Edge-Struktur	18
2.3.3 Randgitter	22
2.4 Konstruktion globaler Voronoi-Knoten	24
2.5 Problematik der Randelemente	25
3 Gitterbewegung mithilfe von CVT	29
3.1 Der Begriff Centroidal-Voronoi-Tessellierung	29
3.2 Globale Optimalität von CVT	30
3.2.1 CVT als Gradient der Fehlerfunktion	30
3.2.2 Normierte Trägheit und Ideales Polyeder	35
3.2.3 Definition der Zelldichte	37
3.3 Lloyd-Algorithmus	40
3.4 Alternativer Lloyd-Newton-Algorithmus	42
3.5 Auswertung der mehrfachen Integrale	46
3.5.1 Volumenintegrale	46
3.5.2 Flächenintegrale	53
3.6 Konvergenz	54
3.6.1 Konvergenzrate beim Lloyd-Algorithmus	55
3.6.2 Konvergenzrate beim alternativen Lloyd-Newton-Algorithmus	60
3.7 Wodurch entsteht Gitterbewegung?	63
3.7.1 Vorbereiten des initialen Netzes zur Strömungssimulation	65
3.7.2 Optimierung der Randzellen mithilfe von CCVT	67
3.7.3 Randbedingungen zur Verschiebung der Generatoren	69

3.8 Wahl der Quadraturformeln zur Auswertung der Volumen- bzw. der Flächenintegrale	70
4 Nutzung des Verfahrens mit der FVM	73
4.1 Lagrangesche und Eulersche Darstellung	73
4.2 Erhaltungssätze der Strömungsmechanik	74
4.2.1 Massenerhaltung	74
4.2.2 Impulserhaltung	75
4.3 Navier-Stokes-Gleichungen	76
4.4 Arbitrary-Lagrange-Euler-Formulierung	77
4.5 Geometrische Erhaltungssätze	78
4.6 Diskretisierung von Differentialoperatoren	79
4.6.1 Zeitdiskretisierung	80
4.6.2 Ortsdiskretisierung mittels der Finite-Volumen-Methode	83
4.6.3 SIMPLE-Algorithmus für die FVM	86
4.6.4 Gitterqualitätskriterien im Bezug auf die FVM	90
4.6.5 Motivation zur Verwendung der Centroidal-Voronoi-Gitter	91
4.7 Berechnung der zulässigen Netzflüsse	92
4.7.1 Korrektur der geschätzten Netzflüsse	92
4.7.2 Eine gute Näherung der Netzflüsse	98
4.8 Verteilung der Zelldichte bzw. der Kantenlänge	102
5 Validierung in OpenFOAM	105
5.1 Integration in OpenFOAM	105
5.2 Propeller im Wasser	105
6 Fazit und Ausblick	121
Literaturverzeichnis	122

Abbildungsverzeichnis

2.1	Voronoi-Diagramm im zwei-dimensionalen Raum. Erzeugt durch den in dieser Arbeit entwickelten Algorithmus, indem nur eine Zellschicht in der dritten Dimensionsrichtung generiert wurde.	9
2.2	Voronoi-Diagramm (durchgehende Liniensegmente) und seine duale Delaunay-Triangulierung (gestrichelte Liniensegmente) im zwei-dimensionalen Raum	9
2.3	Edge-Flip beim Verletzen der Delaunay-Bedingung. Die Kante p_0p_1 wird durch die Kante p_3p_4 ersetzt, da die Sphäre durch $p_0p_1p_2p_4$ den Punkt p_3 in ihrem Inneren enthält.	11
2.4	Schnitt eines Quaders mit einem Bisektor	13
2.5	Die Nachbarn mehrerer Stufen der weiß gefärbten Voronoi-Zelle: erster Stufe in rot, zweiter Stufe in grün und dritter Stufe in blau.	14
2.6	Abbruchbedingung bei der Bestimmung der nächsten Nachbarn. Die schwarzen Liniensegmente entsprechen den Voronoi-Kanten des alten Voronoi-Diagramms. Die schwarzen bzw. blauen Punkte entsprechen den alten bzw. neuen Generatoren. Die zu berechnende Voronoi-Region Ω_i ist durch blaue Liniensegmente abgegrenzt. Der Radius von Ω_i ist durch die blaue gestrichelte Linie dargestellt. Die roten Liniensegmente bilden eine Art Hülle um Ω_i , so dass kein Generator aus den Nachbarstufen höher als 3 einen gemeinsamen Bisektor mit Ω_i haben kann. Diese Hülle befindet sich zwischen den Nachbarn zweiter und dritter Stufen.	15
2.7	Öffnungswinkel zwischen zwei benachbarten Dreiecken	19
2.8	Schnittpunkte einer Gerade mit einem konvexen Polyeder. Das Liniensegment ist durch die Punkte p_0 und p_1 gegeben. Die Schnittebene wird mithilfe des inneren Punktes p_2 konstruiert. Der Schnitt dieser Ebene mit dem Rand der dargestellten Voronoi-Zelle erzeugt einen Polygonzug, der in orange gekennzeichnet ist. p_3 und p_4 sind die Schnittpunkte der Gerade $l(\tau)$ mit dem Polygonzug.	20
2.9	Verschneiden einer Voronoi-Zelle mit dem Randgitter. Die Sicht von außen. Die Nummern auf dem Bild entsprechen den Indizes der Voronoi-Knoten.	24
2.10	Zwei Volumina einer Voronoi-Region (in blau). Die Nachbarzelle ist in grün abgebildet. Das Randgitter hat graue Farbe.	26
2.11	Voronoi-Nachbarn mit zwei gemeinsamen Flächen. Das Randgitter ist in orange dargestellt. Die Voronoi-Regionen sind in grauer Farbe abgebildet.	27
3.1	Vergleich zwischen einem Voronoi-Gitter und einem Centroidal-Voronoi-Gitter. Die Anzahl der Zellen ist bei beiden Gittern gleich.	30
3.2	Abgeschnittenes Oktaeder. Die Oberflächen sind entweder Quadrate oder regelmäßige Sechsecke.	37
3.3	Zerlegung einer Pyramide in Tetraeder	48

3.4	CVT eines rechteckigen Prismas der Länge 2 mit einem Kantenverhältnis von 0.1 zu 1. D.h. die Kantenlänge ist eine lineare Funktion des Ortes: $\mathcal{L}(x) = 0.1 \cdot (2 - x^1) + 1 \cdot x^1$. Dabei ist x^1 die Koordinate der Längsrichtung des Prismas. Quadraturformeln des Genauigkeitsgrades von 1 bis 6. Der Lloyd-Algorithmus wurde bis zum absoluten Fixpunkt durchgeführt. Das Modell enthält 3651 Zellen.	50
3.5	Die erreichte Kantenlänge gegenüber der Ortskoordinate	51
3.6	Zerlegung der Flächenelemente in Dreiecke.	53
3.7	Orthogonales Prisma. Abmessungen: H/B/L: 1m/1m/2m	55
3.8	Konvergenz von Lloyd-Algorithmus. Acht verschiedene Gitter: von 72 bis 3651 Zellen. Modell ist in der Abb. 3.7 dargestellt.	57
3.9	Fehlerreduktion beim Lloyd-Algorithmus. Acht verschiedene Gitter: von 72 bis 3651 Zellen. Modell ist in Abb. 3.7 dargestellt.	58
3.10	Abhängigkeit der Konvergenz vom Gradienten der Zelldichte beim Lloyd-Algorithmus. Im Bezug auf die Abbildung 3.7 beträgt die Kantenlänge rechts 1 und links wie in der Legende angegeben.	59
3.11	Abhängigkeit der Konvergenz vom Parameter α beim Lloyd-Newton-Algorithmus. Inhomogene Zelldichte: Kantenlängenverhältnis von 0.1 zu 1. Das Gitter mit 1519 Elementen.	61
3.12	Durchschnittliche Konvergenzraten aus den ersten 20 Iterationen des Lloyd-Algorithmus und des Lloyd-Newton-Algorithmus bei $\alpha = 0.3$. Inhomogene Zelldichte mit Kantenlängenverhältnis von 0.5 zu 1.	62
3.13	Simulation einer frei fallenden Kugel in einem viskosen Fluid. Die Bewegungsrichtung zeigt nach rechts. Das Gesamtgitter enthält 775380 Voronoi-Zellen. Nach einer gewissen Zeit ist die Netzfeinheit am linken Rand der Kugel deutlich höher als am rechten Rand (Siehe die Abb. 3.13b).	64
4.1	Veranschaulichung der ALE-Formulierung	77
4.2	Verbindungsline der Zellenschwerpunkte im Bezug zu dem Flächenschwerpunkt. \mathbf{x}_o und \mathbf{x}_n sind die Schwerpunkte der benachbarten Zellen. \mathbf{x}_f ist der Schwerpunkt der Fläche. \mathbf{n} ist die normierte Flächennormale. D.h. $\ \mathbf{n}\ = 1$. Außerdem gilt $\mathbf{u} := \mathbf{x}_n - \mathbf{x}_o$ und $\mathbf{v} := \mathbf{x}_f - \mathbf{x}_s$	90
4.3	Berechnung der Rotationskomponente der Verschiebung. Die Punktbezeichnungen werden schwarz dargestellt. Die Vektoren werden in Farbe abgebildet.	100
4.4	Abhängigkeit der Änderungsrate der Elementgröße vom κ	104
5.1	Triangulierte Oberflächen zur Simulation des Propellers. Die Drehachse des Rotors wird etwas verlängert und ragt aus dem äußeren Zylinder heraus. Der Durchmesser der Drehachse ist 0.05278 Meter. Der größte Umfang der Rotorflügel beträgt ca. 0.22 Meter. Die Länge des Rotors beträgt 0.271 Meter. Der Umfang des äußeren Zylinders ist 0.6 Meter. Die Länge des Zylinders beträgt 1 Meter.	106
5.2	Das Geschwindigkeitsfeld bei dem unter OpenFOAM abgelegten Tutorial "propeller". Die Anzahl der Zellen beträgt 525586. Das Netz wurde mit dem Tool snappyHexMesh erzeugt. Dadurch sind die meisten Zellen Hexaeder.	108
5.3	Ungünstige Überlappung von zwei Flächenelementen. $\cos(\alpha) = \frac{\mathbf{n} \cdot \mathbf{u}}{\ \mathbf{u}\ } \approx 0$	108
5.4	Das Teilnetz des Prozessors, der AMI-Randbedingung enthält	109
5.5	Stirnfläche der Propellerflügel in der Gitterdarstellung	110
5.6	Simulation des drehenden Propellers. Y-Komponente des Geschwindigkeitsfeldes in der XY-Ebene. Die Lösung wird an den Zellmittelpunkten ausgewertet.	111

Abbildungsverzeichnis

5.7 Simulation des drehenden Propellers. Y-Komponente des Geschwindigkeitsfeldes wird an der XY-Schnittebene ausgewertet. Der Wertebereich wurde durch das kleinste Intervall beschränkt.	112
5.8 Simulation des drehenden Propellers. Die Netze aller 4 Rechenfälle.	113
5.9 Simulation des drehenden Propellers. Y-Komponente des Geschwindigkeitsfeldes in der YZ-Ebene. Die Lösung wird an den Zellmittelpunkten ausgewertet. Der vergrößerte Bereich des Rotors. Der Wertebereich wurde durch das kleinste Intervall beschränkt.	114
5.10 Simulation des drehenden Propellers. Der Betrag der Wandschubspannung. . . .	115
5.11 Rotor in der Gitterdarstellung	116
5.12 Druckkraft des Propellers	118
5.13 Teilnetz eines Prozessors vom Rechenfall 1	119

Tabellenverzeichnis

3.1 Gaußsche Quadraturformel des zweiten Grades für das Referenztetraeder	49
3.2 Das erreichte Kantenlängenverhältnis bei verschiedenen Genauigkeitsgraden der Quadraturformeln. Der Sollwert des Kantenlängenverhältnisses ist 10. Es handelt sich um die <i>Keast-Rules</i>	52
3.3 Quadraturformel des dritten Genauigkeitsgrades für das Referenzdreieck	54
5.1 Rechenfälle zur Validierung des entwickelten Netzbewegungslösers	107

Verzeichnis von Algorithmen

2.1	Algorithmus zur Bestimmung der nächsten Voronoi-Nachbarn	16
2.2	Rekursiver Algorithmus zur Bestimmung der Schnittpunkte einer Feature-Edge-Struktur mit einem Voronoi-Gitter	21
3.1	Lloyd-Algorithmus	40
3.2	Newton Verfahren zur Berechnung der CVT	42
3.3	Lloyd-Newton-Algorithmus zur Berechnung der CVT	45
3.4	Ein Algorithmus zur Netzbewegung mithilfe von CVT	63
3.5	Ein verbesserter Algorithmus zur Netzbewegung auf Basis von CVT	65
3.6	Vorbereiten des initialen Netzes zur Strömungssimulation	67
4.1	SIMPLE-Algorithmus	89
4.2	Berechnung der Verteilung der Zelldichte	104

1 Einleitung

Der rasante Anstieg von Rechen- sowie Speicherkapazitäten hat dazu beigetragen, dass heutzutage so gut wie jede technische Entwicklung nicht ohne rechnergestützte Simulation auskommen kann. Die meisten physikalischen Vorgänge lassen sich durch partielle Differentialgleichungen beschreiben, die mithilfe der Finite-Volumen-Methode oder der Finite-Elemente-Methode gelöst werden können. Die numerische Stabilität dieser Verfahren ist sehr stark von der Qualität der verwendeten örtlichen Diskretisierung abhängig. Es gibt eine Fülle von Programmen, die hoch-qualitative Netze generieren können. Die hohe Qualität des inneren Netzes bei unbeschränkt großen Verformungen des Gebietsrandes zu erhalten, bleibt jedoch ein Gegenstand der Forschung bis in unsere Zeit. Unter der Gitterqualität werden bestimmte Kriterien verstanden, durch welche die Genauigkeit der verwendeten Diskretisierungsschemata stark beeinflusst wird. Für Finite-Volumen-Methode werden diese Kriterien in dem Unterabschnitt 4.6.4 aufgestellt. Im allgemeinen sind die Gitter mit gleichseitigen bzw. mit gleichwinkligen Elementen optimal.

1.1 Motivation

Die vorliegende Doktorarbeit wurde von Volkswagen AG gefördert und in der Abteilung Komponentenentwicklung am Standort Salzgitter durchgeführt. Fachlich wurde diese Arbeit durch das Institut für Mathematik der Technischen Universität Berlin unterstützt.

Am Standort Salzgitter wird eine große Reihe von Verbrennungsmotoren für verschiedene Automobilmarken des Volkswagen Konzerns entwickelt und hergestellt. Ein umweltbewusstes Verhalten der Volkswagen AG spiegelt sich in der Optimierung auch der kleinsten Motorkomponenten von modernen Fahrzeugen wider. Vor der Anfertigung durchläuft ein Bauteil mehrere Zyklen vom Entwurf zum Prototyp. Da die Herstellungskosten für Prototypen meistens im vierstelligen Bereich liegen, werden die Bauteile nach dem Entwurf mit Hilfe von modernen Berechnungsprogrammen auf mögliche Schwachstellen untersucht.

Die jahrelang andauernde technische Entwicklung hat zu sehr eleganten Geometrien der Motorkomponenten geführt, so dass es meistens sehr viel intellektuellen Einsatz kostet, um diese Komponenten simulieren zu können. Die zurzeit als Bremskraftverstärker dienende Verdrängerpumpe hat beispielsweise einen seitlich versetzten Rotor, wodurch sehr kleine Spalte entstehen. Diese nicht symmetrische Geometrie sowie hohe Turbulenzen in der Strömung machen die Durchführung einer transienten bzw. zeitlich aufgelösten Simulation notwendig. Das zeitlich abhängige Gebiet muss zu jedem Zeitschritt in eine Menge von disjunkten Volumenelementen unterteilt werden. Diese Aufgabe kann jedoch aufgrund der beschriebenen Struktur der Verdrängerpumpe durch viele heutzutage existierende Ansätze nicht gelöst werden. Die Vereinigung der disjunkten Volumenelemente wird meistens als Netz bzw. als Gitter bezeichnet.

Eine Strategie hat sich jedoch in diesem Fall gut bewährt, bei der mehrere Netze erzeugt werden und eines davon mit dem Flügel durch den Spalt geschickt wird. Dadurch gibt es bewegliche und ruhende Netze, zwischen denen die berechneten Felder interpoliert werden müssen. Es ist allgemein bekannt, dass so eine Interpolation zu großen Verfälschungen der realen physikalischen Phänomene führen kann. Ein zweiter und ebenso wichtiger Punkt ist, dass die bereits bestehenden Ansätze monatelang menschliche Arbeit erfordern. Denn für jede Geometrie muss

ein optimales Netz manuell vorbereitet werden. Dabei bezieht sich das Adjektiv “optimal“ auf das gewählte räumliche Diskretisierungsschema. Wie bereits erwähnt wurde, sind die Netze mit gleichseitigen bzw. gleichwinkligen Volumenelementen im allgemeinen optimal. Bei dem beschriebenen Ansatz kann das Verhältnis an einem Knoten angrenzenden Kanten im Bereich des Spalts den Wert 1000 erreichen. Aus diesem Grund können die Matrizen der zu lösenden linearen Gleichungssysteme sehr schlecht konditioniert sein, wodurch die iterativen Verfahren zur Lösung dieser Systeme meist nicht konvergieren. In der Praxis müssen sehr viele Rechnungen durchgeführt werden, bis ein optimales und lauffähiges Netz gefunden wird (siehe [49]).

Man sieht hierdurch die Notwendigkeit eines automatischen und universellen Netzbewegungslösers, der zu jedem Simulationsschritt ein den bestimmten Qualitätskriterien genügendes Gitter erzeugen kann.

1.2 Stand der Wissenschaft

Bei der Simulation von Strömungen in zeitabhängigen Gebieten wurden bereits viele Ansätze entwickelt. Diese kann man in fünf wichtige Klassen einteilen.

- **Overset Grids** (siehe [56, 76]): Bei diesem Ansatz werden mehrere überlappende Netze erzeugt. Das strömungsmechanische Problem und das strukturmechanische Problem können dabei auf entsprechend speziellen Gittern gelöst werden.
- **Immersed boundary** (Quellen [13, 43, 67]): Hierbei wird eine Fläche in ein volumetrisches Gebiet getaucht. Die Schnittpunkte der Volumenelemente mit dieser Fläche repräsentieren den Rand des Gebietes. Die Position des Randes wird nicht exakt berechnet, sondern geht als Koeffizient in das zu lösende lineare Gleichungssystem ein.

Ein Vorteil der beiden oben beschriebenen Verfahren ist die Möglichkeit die Ränder komplizierter Geometrien sehr leicht abzubilden. Ein großer Nachteil ist die lediglich angehöerte Darstellung des Randes, auch wenn die entsprechende interaktive Stelle verfeinert wird.

- **Remeshing oder Adaptive Mesh Refinement** ([39, 47, 70, 71, 75]): Diese Verfahrensklasse wird meistens durch AMR abgekürzt. Im Vergleich zu davor diskutierten Verfahren bietet AMR eine wesentlich genauere Darstellung des Randes, jedoch wird dabei das innere Gebiet zum Teil neu vernetzt. Eine kritische Stelle ist die dadurch notwendige Interpolation der Flüsse zwischen unterschiedlichen Gittern.
- **Mesh Motion**: ([11, 12, 15, 18, 26, 30, 66, 73, 77, 85, 86]): Hierbei werden die inneren Knoten des Gitters als Reaktion der Verlagerung der Randknoten verschoben. Durch diesen Ansatz lassen sich nur bedingt große Randverschiebungen realisieren. Bei periodischen Bewegungen bzw. Rotationen kann dieser Nachteil mithilfe der Interpolation an den sogenannten *Arbitrary-Mesh-Interfaces* (AMI) vermieden werden (siehe [48]).
- **Precomputed Meshes** ([2, 3]): Bei dieser Strategie werden im Vorfeld Gitter nur für eine begrenzte Anzahl von Schritten der transienten Simulation erzeugt. Die Zwischenzustände können durch lineare Verformung dieser vorberechneten Gitter (d.h. durch die Anwendung des Ansatzes Mesh-Motion) erhalten werden.

Der in dieser Arbeit entwickelte Algorithmus kann sowohl der Verfahrensklasse AMR als der Kategorie Mesh-Motion zugeordnet werden, da die Anzahl der Volumenelemente erhalten wird, die Flächenelemente werden jedoch neu erzeugt. Da die Interpolation der Felder nicht benötigt

wird, steht unser Algorithmus der Verfahrensklasse Mesh-Motion wesentlich näher. Wir geben daher einen kurzen Überblick der gängigen Ansätze dieser Verfahrensklasse.

Es gibt Feder-Masse bzw. FEM basierte Mesh-Motion-Solver (siehe dafür [11, 12, 15, 18, 26, 85] bzw. [30]). Hierbei werden die Knotenverschiebungen als elastische Verformungen modelliert. Ein ebenfalls wichtiger Ansatz beruht auf der sogenannten Laplaceschen Glättung der Randverschiebungen ins Innere (siehe [66, 77]). Der meist effektivste Ansatz basiert auf der Interpolation der Randverschiebungen durch die sogenannten radialen Basisfunktionen. Ein großer Nachteil dieser Strategie besteht in der Lösung vollbesetzter linearer Gleichungssysteme, deren Größe von der Anzahl der gewählten Stützpunkte abhängt. Um die Größe des zu lösenden Gleichungssystems in Grenzen zu halten, ist eine optimale Wahl der Stützpunkte durch den Anwender notwendig. Wie bereits erwähnt, besteht ein großer Nachteil dieser und weiterer auf reiner Knotenverschiebung basierender Ansätze darin, dass nur beschränkte Randverformungen realisiert werden können.

1.3 Wissenschaftlicher Beitrag

In der vorliegenden Arbeit wird ein neuer Ansatz zur Netzbewegung für unbeschränkt große Deformationen vorgestellt, der auf der sogenannten *Centroidal-Voronoi-Tessellierung* (CVT) basiert. Eine CVT ist ein Spezialfall einer Voronoi-Tessellierung, indem die diese Voronoi-Tessellierung erzeugenden Generatoren in den geometrischen Schwerpunkten der entsprechenden Voronoi-Gebiete liegen. Der Rand wird hierbei durch eine zusätzliche triangulierte Fläche im STL-Format (Engl. Standard Tessellation Language) dargestellt. Die Netzbewegung kommt dadurch zustande, dass nach jeder Verschiebung der Randfläche die CVT-Eigenschaft aller Gebiete wiederhergestellt wird. Es empfiehlt sich, die zeitlich aufgelöste Simulation mit einem bereits erzeugten CV-Gitter zu starten, denn sonst führt die Rekonstruktion der CVT zu viel zu großen Verschiebungen des darunterliegenden Gitters.

Eine weitere besondere Eigenschaft des vorgestellten Algorithmus ist, dass bei der Wiederherstellung der CVT-Eigenschaft des darunterliegenden Gitters die Anzahl der Volumenelemente beibehalten wird. Außerdem bleibt auch die Zuordnung der Elemente zu den Feldern erhalten. Im Gegensatz zu vorhandenen Ansätzen zur Retopologisierung benötigt man hierdurch keine Interpolation der zu lösenden Felder. Es müssen lediglich die Netzflüsse (Engl. *mesh fluxes*) neu berechnet werden. Im weiteren werden die wichtigsten Punkte beschrieben, die in der vorliegenden Dissertation ausgearbeitet wurden.

- Kapitel 2:

1. Hier wird eine Methode vorgestellt, mit deren Hilfe aus einem beliebigen Gitter ein neues Voronoi-Diagramm in \mathbb{R}^3 konstruiert werden kann. Für n Generatoren bzw. Elemente beträgt die Zeitkomplexität $\mathcal{O}(n)$. Einige der bisher bekannten Algorithmen zur Konstruktion der Voronoi-Diagramme in \mathbb{R}^3 haben im *worst case* die Zeitkomplexität $\mathcal{O}(n \log(n))$ (siehe [69, 80] sowie den Unterabschnitt 2.1.3), besitzen jedoch eine große Konstante vor \mathcal{O} , wie es beispielsweise bei der in MATLAB angebotenen Variante ist. Einige dieser Algorithmen haben zwar eine „kleine“ Konstante vor \mathcal{O} , besitzen jedoch im *worst case* die Zeitkomplexität $\mathcal{O}(n^2)$ (siehe [4]). Der in dieser Arbeit entwickelte Algorithmus besitzt sowohl eine „kleine“ Konstante vor \mathcal{O} als auch die Zeitkomplexität $\mathcal{O}(n)$.

Die erreichte Verbesserung beruht darauf, dass der entwickelte Algorithmus die Informationen des vorhandenen Gitters ausnutzt, wodurch neue nächste Nachbarn jedes Voronoi-Gebietes in konstanter Rechenzeit $\mathcal{O}(1)$ bestimmt werden können.

Kapitel 1. Einleitung

2. Ein weiterer wichtiger Beitrag besteht in einem Ansatz zur rekursiven Verschneidung eines Voronoi-Diagramms in \mathbb{R}^3 mit einer geschlossenen triangulierten Fläche (STL). Für m an den Rand angrenzenden Volumenelemente und l Dreiecke des Randgitters hat dieser Algorithmus die Zeitkomplexität $\mathcal{O}(\max(m, l))$. Entscheidend ist dabei, dass bei einem festgelegten Randgitter die Zeitkomplexität des Algorithmus nicht von der gesamten Anzahl der Volumenelemente sondern nur von der Anzahl der an den Rand angrenzenden Volumenelementen abhängt. Es können sowohl konvexe als auch konkave Randflächen behandelt werden. Darüber hinaus werden die spitzen Winkel bzw. die scharfen Kanten in die endgültige Struktur der Voronoi-Zellen einfließen.

- Kapitel 3:

1. Hier wird gezeigt, dass eine *Centroidal-Voronoi-Tessellierung* (CVT) genau dann vorliegt, wenn eine Art Fehlerfunktion an der entsprechenden Stelle ihr Minimum annimmt. Es wird der Gradient dieser Funktion in geschlossener Form berechnet und bewiesen, dass dieser Gradient an der “CVT-Stelle” gleich einem Nullvektor ist. Für eine einzelne Zelle ist diese Funktion genau dann minimal, wenn die Form dieser Zelle einer Kugel ähnelt. Dadurch ist die kugelförmige Gestalt der Centroidal-Voronoi-Elemente bedingt.
2. Weiterhin wird in diesem Kapitel ein auf dem Newton-Verfahren basierender Algorithmus zur Berechnung der CVT vorgestellt, der eine höhere Konvergenzrate als der klassische Lloyd-Algorithmus hat. Dieser Algorithmus unterscheidet sich von der bereits vorhandenen Lloyd-Newton-Methode dadurch, dass die Matrix des zu lösenden linearen Gleichungssystem symmetrisch positiv definit ist. Außerdem ist diese Matrix bei geeigneter Parameterwahl irreduzibel diagonal dominant. Diese Umstände ermöglichen den Einsatz solcher effizienter Verfahren zur Lösung linearer Gleichungssysteme wie die Methode der konjugierten Gradienten und die unvollständige Cholesky-Zerlegung.
3. Eine weitere Errungenschaft der vorliegenden Arbeit besteht in einer experimentellen Bestimmung des niedrigst notwendigen Genauigkeitsgrades der Quadraturformel, die zur numerischen Auswertung der Volumen- bzw. Flächenintegrale benutzt wird. Es handelt sich dabei um die Volumenintegrale, die bei der Berechnung der mit der Zelldichte gewichteten Schwerpunkte im Lloyd-Algorithmus ausgewertet werden müssen.
4. Die in dieser Arbeit beschriebene Optimierung der an den Rand angrenzenden Volumenelemente mit Hilfe einer Art diskreten Projektion der entsprechenden Generatoren auf die Randfläche leistet ebenso einen großen Beitrag zur Forschung auf dem Gebiet Gittergenerierung.

- Kapitel 4:

1. Ein wichtiger und zum Fortschritt beitragender Teil dieses Kapitels beschäftigt sich mit der Bestimmung der Netzflüsse, bei denen das sogenannte *Volume Conservation Law* (VCL) erfüllt ist. Der im Rahmen dieser Arbeit entstandener Netzbewegungslöser lässt es nicht zu, die Netzflüsse auf klassische Weise zu bestimmen (siehe die Literatur [22, 28, 31, 32, 35]). Es wurde daher ein Algorithmus entwickelt, mit dessen Hilfe jede beliebige Schätzung der Netzflüsse so angepasst werden kann, dass das VCL erfüllt ist.

2. Im abschließenden Teil dieses Kapitels wird ein effizienter Ansatz bekannt gegeben, mit dessen Hilfe eine zweimal stetig differenzierbare Zelldichte-Funktion für ein abgeschlossenes Gebiet in \mathbb{R}^3 so berechnet werden kann, dass diese den auf dem Gebietsrand vordefinierten Werten genügt.

1.4 Kapitelübersicht

- Das Kapitel 2 beschäftigt sich mit Voronoi-Diagrammen in \mathbb{R}^3 . Am Anfang werden die vier klassischen Verfahren zur Konstruktion der Voronoi-Diagramme kurz beschrieben. Danach wird ein neuer im Laufe dieser Dissertation entstandener Algorithmus zur Konstruktion der Voronoi-Diagramme vorgestellt. Zum Schluss erfolgt die Behandlung allgemeiner nicht konvexer Gebiete.
- In dem Kapitel 3 wird die Bedeutung der sogenannten Centroidal-Voronoi-Tessellierung (CVT) für das Forschungsgebiet Gittergenerierung dargestellt. Es wird gezeigt, dass CV-Gitter im Gegensatz zu allen anderen regulären Gittern eine Art globales Optimalitätskriterium erfüllen. Ferner werden hier die Methoden beschrieben, die zur Konstruktion von CVT erforderlich sind.
- Der Zweck des Kapitels 4 ist zu berichten, welche zusätzliche Methoden noch benötigt werden, um den entwickelten Algorithmus zur Netzbewegung in Kombination mit der Finite-Volumen-Methode verwendbar zu machen. Am Anfang des Kapitels werden mithilfe einiger Literaturquellen die Navier-Stokes-Gleichungen in der sogenannten ALE-Formulierung (Arbiträre-Lagrangesche-Eulersche-Formulierung) hergeleitet. Darüber hinaus wird ein zusätzlicher geometrischer Erhaltungssatz vermittelt, der in der englischen Literatur als *Volume Conservation Law* bezeichnet wird.
- Das Kapitel 5 beschreibt Integration des entwickelten Netzbewegungslösers in OpenFOAM und vergleicht ihn mit den in diesem Paket eingebauten Ansätzen anhand einer relativ komplizierten Geometrie des Propellers. OpenFOAM ist eine frei verfügbare und in C++ implementierte Bibliothek, die ein großes Sortiment von Lösern zur Strömungssimulation anbietet.

2 Voronoi-Diagramme nicht konvexer Gebiete

In diesem Kapitel wird eine Variante der Voronoi-Zerlegung vorgestellt, die im ungünstigsten Fall die Zeitkomplexität $\mathcal{O}(N)$ besitzt. N ist hierbei die Anzahl der Voronoi-Regionen. Alle bisher bekannten Methoden zur Konstruktion von Voronoi-Diagrammen haben im günstigsten Fall die Zeitkomplexität $\mathcal{O}(N \log(N))$, im ungünstigsten Fall kann jedoch die Zeitkomplexität $\mathcal{O}(N^2)$ sein (siehe [69]). Das trifft auch bei der heutzutage effizientesten Umsetzung zu (siehe [4]).

Die Idee des hier beschriebenen Ansatzes besteht darin, dass ein Voronoi-Diagramm des definierten Gebietes schon vorhanden ist, so dass für eine gewisse Verschiebung der Generatoren dieses Voronoi-Diagramm nur noch aktualisiert werden kann. Die Voraussetzung für die Durchführbarkeit dieser Aktualisierung ist, dass der neue Generator innerhalb seiner bisherigen Voronoi-Region liegt. Diese Eigenschaft ist beim im Kapitel 3 vorgestellten Algorithmus zur Gitterbewegung erfüllt. Ein großer Vorteil des in dieser Arbeit entwickelten Algorithmus zur Aktualisierung eines Voronoi-Diagramms ist seine Durchführbarkeit auch in dem Fall, wenn das initiale Gitter kein Voronoi-Diagramm ist. Eine wichtige Voraussetzung an dieses Gitter jedoch ist, dass alle seine Flächenelemente eben sind.

Für die Darstellung des Gebietsrandes verwenden wir in dieser Arbeit ein zusätzliches Oberflächen- bzw. Randgitter, das ausschließlich aus Dreiecken besteht. Verschiebungen durch vorgegebene Randbedingungen werden auf dieses Randgitter angewandt. Durch solch eine Darstellung der Gebietsgrenzen kann die Randtopologie des Volumengitters während der transienten Simulation verändert werden. Die Position der neu entstandenen Randknoten des Volumengitters kann dadurch gewonnen werden, dass das Voronoi-Diagramm der konvexen Hülle des aktuellen Rechengebietes mit dem Randgitter verschnitten wird.

Ein weiterer Vorteil der in dieser Arbeit vorgestellten Voronoi-Zerlegung besteht darin, dass die Zeitkomplexität unabhängig von der Konzentration der Generatoren im Raum ist. Die Voraussetzung dafür ist, dass diese Konzentration sich vom alten zum neuen Gitter nur unwesentlich verändert, was bei den meisten praktischen Problemstellungen der Fall ist.

2.1 Theoretische Hintergründe

In diesem Abschnitt lehnen wir uns an [69]. In dieser Schrift werden die Voronoi-Zerlegung und die Delaunay-Triangulierung im zwei-dimensionalen Raum betrachtet. In der vorliegenden Dissertation werden die wichtigsten Definitionen und Sätze auf den drei-dimensionalen Raum übertragen.

2.1.1 Voronoi-Diagramme

Wir betrachten eine Menge $\{\mathbf{x}_i\}_{i=1}^N$, so dass $\mathbf{x}_i \in \mathbb{R}^3$ für $i = 1, 2, \dots, N$ ist.

Definition 2.1 (Bisektor). *Beim Bisektor von zwei Punkten $\mathbf{x}_1 \in \mathbb{R}^3$, $\mathbf{x}_2 \in \mathbb{R}^3$ handelt es um die Menge*

$$B(\mathbf{x}_1, \mathbf{x}_2) := \{\mathbf{x} \in \mathbb{R}^3 : \|\mathbf{x}_1 - \mathbf{x}\| = \|\mathbf{x}_2 - \mathbf{x}\|\}. \quad (2.1)$$

Kapitel 2. Voronoi-Diagramme nicht konvexer Gebiete

Der Bisektor $B(\mathbf{x}_1, \mathbf{x}_2)$ besteht aus allen Punkten $\mathbf{x} \in \mathbb{R}^3$, die sich auf der mittelsenkrechten Ebene des Liniensegments $\mathbf{x}_1\mathbf{x}_2$ befinden. Diese Ebene zerlegt den Raum \mathbb{R}^3 in zwei offene Halbräume:

$$D(\mathbf{x}_1, \mathbf{x}_2) = \{\mathbf{x} \in \mathbb{R}^3 : \|\mathbf{x}_1 - \mathbf{x}\| < \|\mathbf{x}_2 - \mathbf{x}\|\}, \quad (2.2)$$

$$D(\mathbf{x}_2, \mathbf{x}_1) = \{\mathbf{x} \in \mathbb{R}^3 : \|\mathbf{x}_2 - \mathbf{x}\| < \|\mathbf{x}_1 - \mathbf{x}\|\}. \quad (2.3)$$

Definition 2.2 (Voronoi-Region). Sei $\mathcal{X} := \{\mathbf{x}_i\}_{i=1}^N$ mit $\mathbf{x}_i \in \mathbb{R}^3$. Eine Voronoi-Region $\Omega_i \subset \mathbb{R}^3$ zugehörig zu \mathbf{x}_i ist wie folgt definiert:

$$\Omega_i = \{\mathbf{x} \in \mathbb{R}^3 \mid \|\mathbf{x}_i - \mathbf{x}\| < \|\mathbf{x}_j - \mathbf{x}\| \text{ für } j = 1, \dots, N \text{ mit } j \neq i\}. \quad (2.4)$$

Offensichtlich gilt auch:

$$\Omega_i = \bigcap_{\mathbf{x}_j \in \mathcal{X} \setminus \mathbf{x}_i} D(\mathbf{x}_i, \mathbf{x}_j). \quad (2.5)$$

Die Punkte \mathbf{x}_i bezeichnen wir als Generatoren der entsprechenden Voronoi-Regionen.

Eine durch die Definition 2.2 erzeugte Voronoi-Region ist als Durchschnitt von $N - 1$ offenen Halbräumen offen und konvex, jedoch nicht unbedingt beschränkt.

Definition 2.3 (Voronoi-Diagramm). Seien $\mathcal{X} := \{\mathbf{x}_i\}_{i=1}^N$ mit $\mathbf{x}_i \in \mathbb{R}^3$ und $\Omega_i \subset \mathbb{R}^3$ für $i = 1, 2, \dots, N$ die zugehörigen Voronoi-Regionen. Das Voronoi-Diagramm zu der Menge \mathcal{X} ist wie folgt definiert:

$$V(\mathcal{X}) := \mathbb{R}^3 \setminus \bigcup_{i=1}^N \Omega_i. \quad (2.6)$$

Das ist die Menge der Punkte, die zu keinem \mathbf{x}_i gehören, sondern den gleichen Abstand zu mehreren \mathbf{x}_i bilden. Ein Voronoi-Diagramm in \mathbb{R}^3 besteht aus Flächen, Kanten und Knoten. Vorausgesetzt es gibt höchstens 4 Generatoren, die auf einer gemeinsamen Sphäre liegen. Dann gelten folgende Aussagen:

- Jeder Punkt einer Voronoi-Fläche hat genau zwei nächste Nachbarn, d.h. bildet den gleichen Abstand zu genau zwei Generatoren.
- Jeder Punkt einer Voronoi-Kante hat genau drei nächste Nachbarn.
- Ein Voronoi-Knoten hat genau vier nächste Nachbarn.

Im folgenden werden einige wichtige Sätze aus [69] ohne Beweise vorgestellt. Diese Sätze werden in [69] für \mathbb{R}^2 formuliert, und lassen sich leicht auf \mathbb{R}^3 übertragen.

Satz 2.1. Ein Voronoi-Diagramm $V(\mathcal{X})$ ist immer zusammenhängend, wenn die Generatoren aus \mathcal{X} nicht auf einer gemeinsamen Gerade bzw. Ebene liegen.

Satz 2.2. Ein Voronoi-Diagramm von N Generatoren in \mathbb{R}^3 hat $\mathcal{O}(N)$ viele Flächen, Kanten und Knoten.

Satz 2.3. Die Konstruktion eines Voronoi-Diagramms mit N Generatoren in \mathbb{R}^3 hat die Zeitkomplexität $\mathcal{O}(N \log(N))$.

Die Abbildung 2.1 zeigt ein Voronoi-Diagramm und seine Generatoren im zwei-dimensionalen Raum.

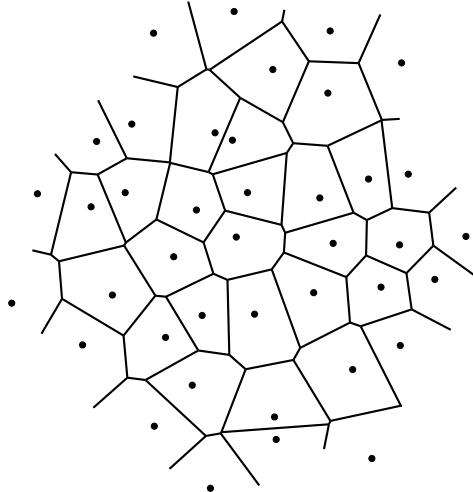


Abbildung 2.1: Voronoi-Diagramm im zwei-dimensionalen Raum. Erzeugt durch den in dieser Arbeit entwickelten Algorithmus, indem nur eine Zellschicht in der dritten Dimensionsrichtung generiert wurde.

2.1.2 Delaunay-Triangulierung, Dualität und Entartung

Definition 2.4 (Delaunay-Triangulierung). *Sei $\mathcal{X} := \{\mathbf{x}_i\}_{i=1}^N$ die Menge von Punkten aus \mathbb{R}^3 und sei $V(\mathcal{X})$ ihr Voronoi-Diagramm. Zwei Punkte $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$ werden durch das Liniensegment $\mathbf{x}_i\mathbf{x}_j$ miteinander verbunden, wenn ihre Voronoi-Regionen V_i und V_j eine gemeinsame Fläche haben. Die durch diese Definition entstehende Menge von Liniensegmenten heißt Delaunay-Triangulierung $DT(\mathcal{X})$ der Punktmenge \mathcal{X} .*

Die Abbildung 2.2 zeigt einen wichtigen Zusammenhang der Delaunay-Triangulierung und des Voronoi-Diagramms.

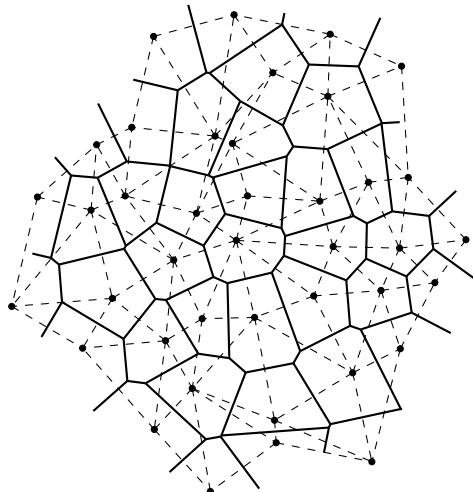


Abbildung 2.2: Voronoi-Diagramm (durchgehende Liniensemente) und seine duale Delaunay-Triangulierung (gestrichelte Liniensemente) im zwei-dimensionalen Raum

Lemma 2.1. *Zwei Generatoren $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$ sind genau dann durch eine Delaunay-Kante miteinander zu verbinden, wenn es eine Sphäre gibt, die nur \mathbf{x}_i und \mathbf{x}_j auf ihrem Rand enthält und keinen anderen Generator aus \mathcal{X} in ihrem Inneren. Der Mittelpunkt dieser Sphäre liegt auf der Voronoi-Fläche zwischen \mathbf{x}_i und \mathbf{x}_j .*

Kapitel 2. Voronoi-Diagramme nicht konvexer Gebiete

Beweis. Nach Definition ist $x_i x_j$ genau dann eine Delaunay-Kante, wenn es im Voronoi-Diagramm $V(\mathcal{X})$ eine gemeinsame Fläche f_{ij} zwischen den Generatoren x_i und x_j gibt. Für jeden Punkt $x \in \mathbb{R}^3$ im Inneren von f_{ij} sind x_i und x_j die einzigen nächsten Nachbarn in \mathcal{X} . Damit enthält die Sphäre durch x_i und x_j im Mittelpunkt x keinen Generator aus $\mathcal{X} \setminus \{x_i, x_j\}$ im Inneren oder auf dem Rand. \square

Satz 2.4 (Delaunay-Bedingung). *Sei $\mathcal{X} := \{x_i\}_{i=1}^N$ die Menge von Punkten aus \mathbb{R}^3 und sei $V(\mathcal{X})$ ihr Voronoi-Diagramm. Die Sphäre durch die Knoten jedes Tetraeders aus $DT(\mathcal{X})$ enthält keine weiteren Delaunay-Knoten aus $DT(\mathcal{X})$ in ihrem Inneren.*

Beweis. Der Satz 2.4 kann analog wie das Lemma 2.1 bewiesen werden. Der Mittelpunkt der Sphäre durch die Knoten jedes Tetraeders liegt dabei exakt auf einem Voronoi-Knoten. \square

Da zwei Voronoi-Regionen höchstens eine gemeinsame Fläche haben können, entspricht dieser eindeutig eine Delaunay-Kante im dualen Graph der Delaunay-Triangulierung. Aufgrund von Lemma 2.1 können sich die Delaunay-Kanten nicht kreuzen. Damit ist jede Delaunay-Triangulierung zusammenhängend. Es gelten folgende Zusammenhänge zwischen einem Voronoi-Diagramm und ihrer dualen Delaunay-Triangulierung:

- Jedem Generator entspricht ein Delaunay-Knoten.
- Für jede Voronoi-Fläche gibt es genau eine Delaunay-Kante.
- Jeder Voronoi-Kante entspricht eine Delaunay-Fläche.
- Zu jedem Voronoi-Knoten gehört ein Delaunay-Dreieck.

Die Delaunay-Triangulierung im \mathbb{R}^3 ist auch eindeutig, enthält im Allgemeinen nicht nur Tetraeder. Ein hexaedrisches Gitter kann auch als ein Voronoi-Diagramm aufgefasst werden, die dualen Elemente sind jedoch Hexaeder. Diese Hexaeder lassen sich in Tetraeder zerlegen. Dennoch gibt es dafür mehrere Wege. Dabei können sowohl fünf als auch sechs Tetraeder entstehen. Das bedeutet, die ausschließlich aus Tetraedern bestehende Delaunay-Triangulierung ist im Allgemeinen nicht eindeutig. Die Voraussetzung für die Eindeutigkeit ist, dass es keine Sphäre gibt, die durch mehr als vier Delaunay-Knoten verläuft. Die Delaunay-Triangulierungen, die diese Bedingung nicht erfüllen, beschreibt man als entartet bzw. degeneriert. Die Elemente einer solchen Delaunay-Triangulierung können beliebige Polyeder sein. In diesen Fällen gibt es keinen eindeutigen Zusammenhang zwischen einem Voronoi-Diagramm und seiner Delaunay-Triangulierung.

Die entarteten Delaunay-Diagramme sind bei einer zufälligen Erzeugung der Generatoren äußerst selten. Das Problem der Entartung wird meistens durch ein zusätzliches Rauschen der Generatoren gelöst (vergleiche [52]).

2.1.3 Klassische Methoden zur Voronoi- bzw. Delaunay-Zerlegung

Es gibt Methoden zur Konstruktion von Voronoi-Diagrammen, die auf die Erzeugung seiner dualen Delaunay-Triangulierung zurückführen. Wie der Satz 2.3 besagt, beträgt die untere Schranke der Zeitkomplexität zur Konstruktion eines Voronoi-Diagramms mit N Generatoren $\mathcal{O}(N \log(N))$. Die Konstruktion der Delaunay-Triangulierung als dualer Graph muss die gleiche Zeitkomplexität besitzen. D.h. unabhängig vom gewählten Algorithmus kann keine bessere Zeitkomplexität erreicht werden. Alle in diesem Unterabschnitt beschriebenen Algorithmen haben demnach im günstigsten Fall die Zeitkomplexität $\mathcal{O}(N \log(N))$ und die Speicherkomplexität $\mathcal{O}(N)$ (Siehe [69]).

Inkrementelle Konstruktion

Der erste Ansatz besteht in dem inkrementellen Hinzufügen neuer Delaunay-Knoten in das bereits bestehende Delaunay-Diagramm. Die Schwierigkeit besteht dabei in der Bestimmung des Tetraeders aus dem alten Gitter, in welchem der neue Delaunay-Knoten liegt. Durch Erzeugung einer speziellen Baumstruktur wird im Mittel die oben erwähnte Zeitkomplexität erreicht. Beim Vorliegen des enthaltenden Tetraeders sind alle seine Knoten mit dem hinzufügten Knoten durch entsprechende Delaunay-Kanten zu verbinden. Danach ist die Delaunay-Bedingung, wie im Satz 2.4 formuliert, für alle Nachbarn zu überprüfen. Beim Verletzen wird ein sogenannter Edge-Flip durchgeführt, wonach diese Bedingung für das entsprechende Tetraeder erfüllt ist. Die Abbildung 2.3 zeigt, was ein Edge-Flip auslöst. Diese Operation ist eindeutig, da es nur zwei Möglichkeiten gibt, eine Pyramide mit einer viereckigen Grundfläche in Tetraeder zu zerlegen. Klein zeigt in [69], dass der Gesamtalgorithmus im *worst case* die Zeitkomplexität $\mathcal{O}(N^2)$ besitzt. Dieser Algorithmus ist in der frei verfügbaren Bibliothek CGAL umgesetzt (siehe [4]).

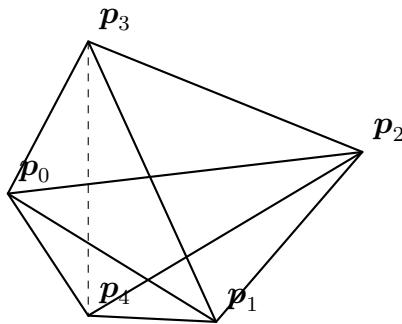


Abbildung 2.3: Edge-Flip beim Verletzen der Delaunay-Bedingung. Die Kante p_0p_1 wird durch die Kante p_3p_4 ersetzt, da die Sphäre durch $p_0p_1p_2p_4$ den Punkt p_3 in ihrem Inneren enthält.

Sweep

In [69] ist der Sweep-Algorithmus für die zwei-dimensionale Voronoi-Zerlegung beschrieben. Es handelt sich dabei um eine Gerade, die durch alle gegebenen Generatoren entlang einer Koordinate wandert. Dazu müssen alle Generatoren nach dieser Koordinate sortiert werden. Die Sortierung hat im günstigsten Fall die Zeitkomplexität $\mathcal{O}(N \log(N))$. Zwischen der Sweep-Gerade und den angrenzenden Generatoren entstehen parabelförmige Bisektoren. Wichtig ist dabei, dass der durch die Sweep-Gerade bereits verstrichene Teil des Voronoi-Diagramms sich nicht mehr ändert. Die Interaktion dieses Algorithmus beschränkt sich nur auf die Generatoren, die als nächste Nachbarn zu der Sweep-Gerade gelten. In [69] wird gezeigt, dass der Sweep-Algorithmus im *worst case* die Zeitkomplexität $\mathcal{O}(N \log(N))$ und die Speicherkomplexität $\mathcal{O}(N)$ besitzt. Offensichtlich lässt sich dieser Algorithmus auf den drei-dimensionalen Raum dadurch übertragen, dass anstelle einer Sweep-Gerade eine Sweep-Ebene verwendet wird.

Divide and Conquer

Die Idee besteht hier im Aufteilen des Gebiets in Untergebiete, die möglichst gleiche Anzahl von Generatoren enthalten. Man kann zeigen, dass es möglich ist, eine Menge von N Generatoren in zwei disjunkte Mengen so aufzuteilen, dass jede Untergruppe mindestens $\frac{1}{4}N$ groß ist. Dazu müssen Generatoren nach allen Koordinaten sortiert werden. Jede Untergruppe wird auch aufgeteilt. Offensichtlich muss die kleinste Untergruppe mindestens einen Generator besitzen, so dass keine direkte Voronoi-Zerlegung der Untermengen notwendig ist. Die Hauptaufgabe besteht dann im Zusammenfügen von Untermengen zu Obermengen. Der Vorteil liegt darin, dass

alle Kanten, die in den Untermengen enthalten sind, auch in der Obermenge vorkommen. Die Kanten in den Untermengen sind jedoch zu lang, da diese über den Rand dieser Mengen hinausreichen. Beim Zusammenfügen wird damit ein Bisektor berechnet, der zusätzliche Kanten zwischen neuen Nachbarn aus beiden Untermengen erzeugt. Die hinausreichenden Kanten werden dabei abgeschnitten. Die Zeitkomplexität ist schon durch die Sortierung nach unten beschränkt und beträgt $\mathcal{O}(N \log(N))$. In Anlehnung an [69] beträgt die Speicherkomplexität $\mathcal{O}(N)$.

Geometrische Transformation

Die Idee dieses Ansatzes besteht in der Erweiterung des darunterliegenden Raumes um eine Dimension $t = x^2 + y^2 + z^2$. Damit erhalten die ursprünglichen Generatoren die Koordinaten (x_i, y_i, z_i, t_i) . Man kann zeigen, dass die Konstruktion einer konvexen Hülle in dem erweiterten Raum äquivalent zur Konstruktion der Delaunay-Triangulierung im ursprünglichen Raum ist. Die Erzeugung der konvexen Hülle hat die Zeitkomplexität $\mathcal{O}(N \log(N))$. Die Datenstrukturen der konvexen Hülle im erweiterten Raum entsprechen vollständig den Datenstrukturen im ursprünglichen Raum, so dass nach der Berechnung kein weiterer Rechenaufwand entsteht, um die Daten zu extrahieren. Bei den Generatoren muss lediglich die vierte Dimension entfernt werden. Dieser Algorithmus wird vom kommerziellen Programm MATLAB angeboten.

Verbesserung einer arbiträren Triangulierung durch Edge-Flip-Operationen

Hierbei wird eine beliebige Triangulierung durch sukzessive Edge-Flip-Operationen wie beim Ansatz „Inkrementelle Konstruktion“ in eine Delaunay-Triangulierung überführt. Man kann zeigen, dass dieser Ansatz zwar auf jede Triangulierung in \mathbb{R}^2 erfolgreich angewendet werden kann, ist jedoch für Triangulierungen in \mathbb{R}^3 im allgemeinen nicht durchführbar (siehe [80]).

2.2 Voronoi-Zerlegung des einhüllenden Quaders

Die Definitionen aus dem Abschnitt 2.1 behalten hier ihre Gültigkeit. Ein einziger Unterschied zu der Definition des Voronoi-Diagramms besteht darin, dass in dieser Arbeit keine unendlich entfernte Voronoi-Knoten zugelassen sind. Es findet keine Zerlegung von \mathbb{R}^3 statt, sondern eines abgeschlossenen Quaders, der alle Generatoren enthält. Dieser wird im weiteren als einhüllender Quader bezeichnet.

Um das Voronoi-Diagramm zu gegebenen Generatoren dieses Quaders zu berechnen, verwenden wir den im Internet frei-verfügbaren Code von C. Rycroft (siehe dazu [72]). Diese Bibliothek berechnet das Voronoi-Diagramm direkt, ohne seine duale Struktur Delaunay-Zerlegung zu bestimmen. Hierbei werden auch die entarteten Voronoi-Diagramme berücksichtigt, da die Existenz der Dualität nicht mehr notwendig ist. Die Knoten eines Voronoi-Diagramms werden bei dieser Bibliothek als Schnittpunkte der Bisektoren bestimmt. Jede Zelle wird separat berechnet und gespeichert.

Zur Bestimmung der nächsten Nachbarn wird der einhüllende Quader in eine Menge von gleich großen Quadern unterteilt, wodurch die Suche nach den nächsten Nachbarn für einen Generator in etwa konstanter Zeitkomplexität geschehen kann. Dies gilt jedoch nur dann, wenn jeder dieser Quadern etwa gleich viele Generatoren enthält. Bei den meisten praktischen Anwendungen ist es oft nicht der Fall, weil bestimmte Gebiete des simulierten Modells wesentlich feiner aufgelöst werden müssen.

Aus diesem Grund wird in der vorliegenden Arbeit zur Bestimmung der nächsten Nachbarn ein alternativer Ansatz verwendet (siehe den Unterabschnitt 2.2.2), deren Zeitkomplexität für einen Generator sowohl konstant als auch unabhängig von der Homogenität der räumlichen Ver-

teilung der Generatoren ist. Aus der oben erwähnten Bibliothek verwenden wir nur die Funktion zur Bestimmung der Schnittpunkte von Bisektoren.

Im folgenden wird beschrieben, wie in dieser Arbeit jede Zelle Ω_i berechnet wird. Zuerst werden die Koordinaten des einhüllenden Quaders berechnet, so dass sowohl alle Generatoren als auch das Randgitter im Quader enthalten sind. Damit dieser Quader auch im numerischen Sinne alle Punkte enthält, muss dieser vergrößert werden. Wir verwenden den Vergrößerungsfaktor 1.01 für jede Ortsrichtung. Jede Voronoi-Zelle wird mit den Abmessungen des einhüllenden Quaders initialisiert, wonach diese mit den Bisektoren der nächsten Nachbarn mithilfe der Bibliothek von C. Rycroft geschnitten wird. Die nächsten Nachbarn werden hierbei mit dem im Unterabschnitt 2.2.2 beschriebenen Algorithmus bestimmt.

2.2.1 Verschneiden mit Bisektoren

Die Datenstruktur in der voro++-Bibliothek ist so konzipiert, dass für jede Zelle nur Knoten und ihre Verbindungen (Kanten) mit anderen Knoten gespeichert werden. Diese Verbindungen werden im Gegenuhrzeigersinn durchgelaufen, wenn man auf die Zelle von außen schaut. Daraus lassen sich auch Flächen rekonstruieren. Jeder Fläche entspricht eine Nachbarzelle. Die Indizes der Nachbarn werden in einem zusätzlichen Feld gespeichert, dessen Länge gleich der Anzahl der Verbindungen jedes Knotens ist. Die Indizes der Nachbarn lassen sich auch wie die Flächen rekonstruieren. Bei der Verschneidung werden neue Verbindungen immer im Gegenuhrzeigersinn erzeugt, so dass die Datenstruktur beibehalten wird.

Die Abbildung 2.4 zeigt, wie ein Quader mit einem Bisektor verschchnitten wird. Für den Knoten p_7 ergibt sich folgende Reihenfolge der Verbindungen: p_8, p_3, p_6 . Diesen Verbindungen entsprechen die Nachbarflächen: F_1, F_2, F_3 . Beim Verschneiden wird die neue Fläche dadurch erzeugt, dass neue Knoten (c_1, c_2, c_3) in folgender Reihenfolge erzeugt werden. Die Verbindungen des außenstehenden Knotens p_7 werden nacheinander besucht, und die Schnittpunkte dieser Verbindungen mit dem Bisektor berechnet. Wenn man mit der Verbindung (p_7-p_3) startet, ergibt sich für die neuen Knoten die Reihenfolge: c_2, c_3, c_1 . Nach dem Verschneiden wird der Knoten p_7 samt seinen Verbindungen entfernt.

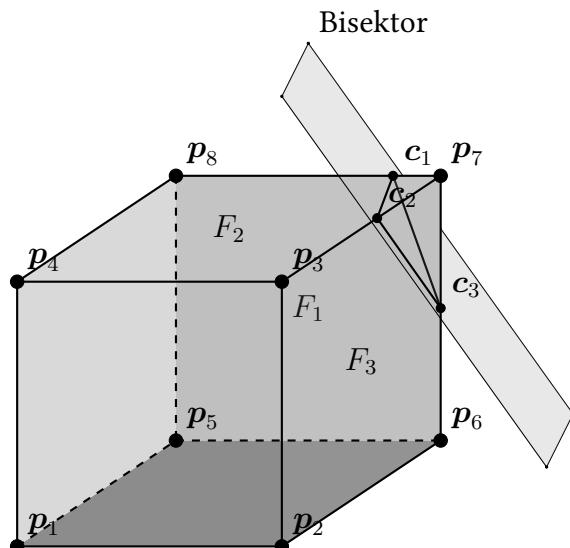


Abbildung 2.4: Schnitt eines Quaders mit einem Bisektor

2.2.2 Bestimmung der nächsten Nachbarn

Meistens wird die Bestimmung des nächsten Nachbarn einer Punktmenge $\{\mathbf{x}_i\}_i^N$ in der Literatur auf die Berechnung des Voronoi-Diagramms von $\{\mathbf{x}_i\}_i^N$ zurückgeführt (siehe [69]). Dabei bezeichnet N die Anzahl der Generatoren. Mit Hilfe dieses Voronoi-Diagramms kann der nächste Nachbar mit einer Zeitkomplexität $\mathcal{O}(\log(N))$ bestimmt werden. Beim inkrementellen Hinzufügen der Generatoren in das bereits bestehende Voronoi-Diagramm ist damit eine Zeitkomplexität von $\mathcal{O}(\log(N)N)$ zu erwarten. Dies entspricht auch dem Satz 2.3, dass $\mathcal{O}(\log(N)N)$ die minimale Zeitkomplexität ist, um ein Voronoi-Diagramm zu erzeugen. Bei einem großen N kann somit der Rechenaufwand immerhin nichtlinear steigen. In dieser Arbeit wird eine andere Strategie verfolgt.

Wir benutzen die Information über die Nachbarn des Gitters an dem alten Zeitschritt. Jede Zelle wird als ein Quader initialisiert und rekursiv mit den Bisektoren der alten Nachbarn geschnitten. Da die Nachbarschaftsbeziehungen sich vom Schritt zu Schritt ändern, müssen mehrere Stufen der alten Nachbarn besucht werden.

Definition 2.5. Unter den Nachbarn der $(n + 1)$ -ten Stufe sind die direkten Nachbarn der n -ten Stufe zu verstehen, die jedoch nicht zu den Stufen n und $(n - 1)$ gehören. Die Nachbarn erster Stufe einer Voronoi-Region sind ihre direkten Nachbarn.

Die Abbildung 2.5 zeigt mehrere Stufen der Nachbarn einer Voronoi-Zelle im zwei-dimensionalen Raum.

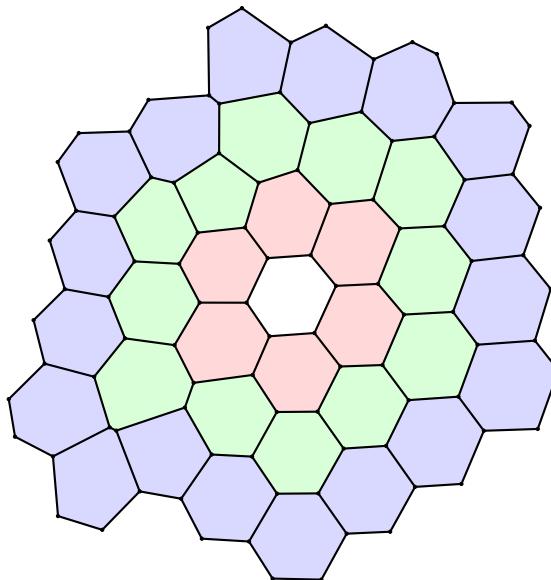


Abbildung 2.5: Die Nachbarn mehrerer Stufen der weiß gefärbten Voronoi-Zelle: erster Stufe in rot, zweiter Stufe in grün und dritter Stufe in blau.

Im Falle der entarteten Voronoi-Zellen, d.h. ein Knoten gehört zu mehr als 4 Zellen, sind die Nachbarn einer Stufe nicht nur über die Flächen zu bestimmen, sondern auch über die Knoten. Da wir davon ausgehen, ein beliebiges Gitter zu haben, kann es sich auch um ein tetraedrisches Gitter handeln. In diesem Fall gibt es viele Nachbarn, die nur über einen Knoten angrenzen, dennoch keine gemeinsame Fläche haben.

Für die Durchführbarkeit des in dieser Arbeit entwickelten Algorithmus zur Aktualisierung einer Voronoi-Zerlegung sind zwei Bedingungen notwendig:

1. Jede Stufe der Nachbarn muss die niedrigere Stufe komplett umhüllen. Für die Stufe n bedeutet dies, dass jede Verbindungsline von jedem inneren Punkt jedes Elements der

Stufe $(n - 1)$ zu jedem inneren Punkt jedes Elements der Stufe $(n + 1)$ eine von Null verschiedene Länge hat. Mit anderen Worten muss die kleinste „Breite“ der Stufe n immer strikt größer Null sein.

2. Neue Generatoren dürfen ihre alte Voronoi-Regionen nicht verlassen. Es wird sich zeigen, dass diese Bedingung bei den in dieser Arbeit verwendeten Algorithmen zur Verschiebung der Generatoren erfüllt ist.

Definition 2.6 (Radius einer Voronoi-Zelle). *Der Radius einer Voronoi-Zelle bezeichnet das Maximum der doppelten Abstände zwischen dem Generator und den Voronoi-Knoten dieser Zelle.*

Nun formulieren wir einen Satz, auf dem der Zerlegungsalgorithmus basiert.

Satz 2.5. *Sei ein Voronoi-Diagramm in \mathbb{R}^3 gegeben. Sei auch eine Menge der Generatoren gegeben, die in ihren alten Voronoi-Regionen liegen. Gäbe es eine Stufe der Nachbarn n , deren Bisektoren keine Schnittpunkte mit der zu berechnenden Voronoi-Zelle Ω_i haben. Wenn der Abstand aller Voronoi-Knoten dieser Nachbarn zum Generator von Ω_i größer als der Radius von Ω_i ist, dann kann kein Bisektor einer höheren Stufe der Nachbarn als n einen Schnittpunkt mit Ω_i haben.*

Beweis. Wir zeigen zuerst die Gültigkeit des Satzes 2.5 anhand eines Voronoi-Diagramms in \mathbb{R}^2 . Die Abbildung 2.6 zeigt solch ein Voronoi-Diagramm.

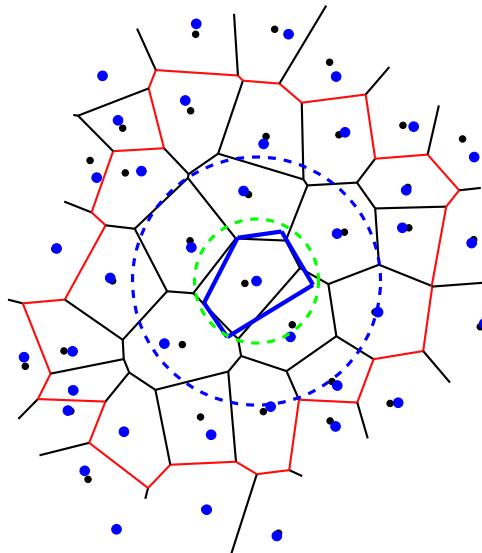


Abbildung 2.6: Abbruchbedingung bei der Bestimmung der nächsten Nachbarn. Die schwarzen Liniensegmente entsprechen den Voronoi-Kanten des alten Voronoi-Diagramms. Die schwarzen bzw. blauen Punkte entsprechen den alten bzw. neuen Generatoren. Die zu berechnende Voronoi-Region Ω_i ist durch blaue Liniensegmente abgegrenzt. Der Radius von Ω_i ist durch die blaue gestrichelte Linie dargestellt. Die roten Liniensegmente bilden eine Art Hölle um Ω_i , so dass kein Generator aus den Nachbarstufen höher als 3 einen gemeinsamen Bisektor mit Ω_i haben kann. Diese Hölle befindet sich zwischen den Nachbarn zweiter und dritter Stufen.

Angenommen die Bisektoren der zweiten Stufe der Nachbarn haben keine Schnittpunkte mit der Zelle Ω_i . Sei es auch so, dass der Abstand aller Voronoi-Knoten dieser Nachbarn zum Generator von Ω_i größer als der Radius von Ω_i sind. Dann gibt es offenbar eine Hölle wie in Abbildung 2.6, die die äußeren Knoten der Nachbarn der zweiten Stufe miteinander verbindet. Es ist offensichtlich, dass der Abstand von jedem Punkt dieser Hölle zum Generator von Ω_i auch größer als der Radius von Ω_i ist. Mit der Voraussetzung, dass die neuen Generatoren ihre alten

Kapitel 2. Voronoi-Diagramme nicht konvexer Gebiete

Voronoi-Regionen nicht verlassen, folgt die Behauptung. Wir gehen außerdem davon aus, dass das alte Gitter zusammenhängend ist, was bei einem Voronoi-Diagramm immer der Fall ist. Der Beweis lässt sich auf \mathbb{R}^3 leicht übertragen. Die Hülle der letzten Nachbarstufe stellt dabei eine Menge von Flächen dar. \square

Erstaunlicherweise gilt der Satz 2.5 nicht nur für Voronoi-Diagramme, sondern auch für ein beliebiges polyedrisches Gitter. Das Gitter muss zusammenhängend sein, und nur konvexe Volumenelemente besitzen. Wichtig ist auch, dass die Flächenelemente eben sein müssen. Der Algorithmus 2.1 zur Bestimmung der nächsten Voronoi-Nachbarn lässt sich aus dem Satz 2.5 direkt ableiten.

Algorithmus 2.1 Algorithmus zur Bestimmung der nächsten Voronoi-Nachbarn

Sei ein zusammenhängendes Gitter aus konvexen Polyedern mit ebenen Flächenelementen gegeben. Bestimme die Nachbarn erster Stufe betrachte diese als aktuelle Stufe. Um die Voronoi-Zelle Ω_i , die zum Generator x_i gehört, zu berechnen, führe folgende Schritte durch:

do

Schneide die Zelle mit den Bisektoren der aktuellen Nachbarn.

if Es gibt Schnittpunkte mit den Bisketoren **then**

Berechne den Radius von Ω_i .

else

Bestimme die Abstände aller Voronoi-Knoten zu dem Generator von Ω_i .

if Alle Abstände sind größer als der Radius von Ω_i . **then**

Die Schleife terminiert.

else

Bestimme die Nachbarn nächster Stufe und betrachte diese als aktuelle Stufe.

end if

end if

while

Bemerkung 2.1 (Zeitkomplexität):

Die Durchführbarkeit des hier beschriebenen Algorithmus ist gegeben, wenn die neuen Generatoren innerhalb ihrer alten Voronoi-Regionen liegen. In der Praxis hat es sich herausgestellt, dass höchstens vier Nachbarstufen des alten Gitters besucht werden müssen, um die nächsten Nachbarn einer Voronoi-Region im neuen Voronoi-Diagramm zu bestimmen. Durchschnittlich reichen dabei nur drei Nachbarstufen aus. Für den entwickelten Algorithmus erhält man damit eine Zeitkomplexität $\mathcal{O}(N)$, wobei N die Anzahl der Generatoren bezeichnet.

Bemerkung 2.2:

Wie am Anfang dieses Abschnittes bereits erwähnt wurde, werden von dem in dieser Arbeit entwickelten Algorithmus zur Voronoi-Zerlegung des einhüllenden Quaders auch die entarteten Voronoi-Diagramme unterstützt.

2.3 Verschneiden mit dem Randgitter

Bei der Verformung eines Gebietes ist es nicht ausgeschlossen, dass einige Volumenelemente neue Randflächen erhalten. Aus diesem Grund kann der Rand zu jedem Zeitpunkt nicht als eine Menge von Flächenelementen dargestellt werden. Es ist daher unmöglich die Verschiebung der Punkte zu definieren, die nur zu bestimmten Zeitpunkten der transienten Simulation existieren.

Durch die Verwendung eines zusätzlichen Randgitters, das nur aus Dreiecken besteht, ist die oben beschriebene Problematik ausgeschlossen. Bei der Deformation behält dieses Randgitter seine Topologie. Die Randverschiebung lässt sich damit leicht auf dieses Randgitter anwenden, welches aus mehreren geschlossenen bzw. „wasserdichten“ Flächenverbunden bestehen kann.

2.3.1 Überblick über den Algorithmus

Das Verschneiden des Volumengitters des einhüllenden Quaders mit dem Randgitter erzeugt die endgültige Zerlegung des definierten Rechengebietes. Dieses algorithmische Problem lässt sich in folgende Schritte unterteilen:

1. Identifikation der Volumenelemente, die ein *feature edge* oder ein *feature point* enthalten. Es handelt sich dabei um scharfe Kanten, die in dem endgültigen Volumengitter auch abgebildet werden müssen. Ein Verzicht auf diese Strukturen würde auf ein unnatürliches Abrunden des Randes führen. Die Feature-Edge-Struktur lässt sich aus dem Randgitter ableiten, indem man die Winkel zwischen zwei Dreiecken mit einem vorgegebenen Wert vergleicht. Beim Überschreiten dieses Wertes ist ein Feature-Edge gegeben. Ein Feature-Point ist gegeben, wenn drei Feature-Edges sich in einem Punkt treffen.

Nachdem die Feature-Edge-Struktur erzeugt wurde, wird diese mit dem Gitter des einhüllenden Quaders verschnitten. Genauer genommen werden die Schnittpunkte der Voronoi-Flächen mit der Feature-Edge-Struktur bestimmt, wodurch die Menge der Schnittpunkte der Voronoi-Kanten mit dem Randgitter vervollständigt wird.

2. Abhängig davon, ob es Restriktionen für die Generatoren der Randzellen gibt, können die Schnittpunkte der Voronoi-Flächen mit der Feature-Edge-Struktur dafür benutzt werden, um die Position der entsprechenden Generatoren zu berechnen.

Im Unterabschnitt 3.7.2 wird beschrieben, wie die Generatoren der Randelemente so berechnet werden, dass die auf dem Rand liegenden Flächenelemente optimiert werden. Wenn die Position eines Generators verändert wird, muss die Voronoi-Zerlegung der entsprechenden Voronoi-Zelle und ihrer Nachbarn neu erzeugt werden. Danach muss der Schritt 1. wiederholt werden, um die Schnittpunkte mit der Feature-Edge-Struktur zu aktualisieren. Diese Schritte werden so lange wiederholt, bis es keine Randzellen gibt, bei denen sich die Feature-Edge-Struktur ändert.

3. In diesem Schritt wird das Voronoi-Diagramm des einhüllenden Quaders mit dem Randgitter verschnitten. Es handelt sich dabei um einen rekursiven Prozess. Für jeden geschlossenen Flächenverbund muss zumindest eine Voronoi-Kante gefunden werden, die einen Schnittpunkt mit einem Dreieck dieses Verbunds hat. Analog wie im Unterabschnitt 2.2.1 werden die Schnittpunkte der Voronoi-Kanten der entsprechenden Zelle bestimmt. Zusätzlich werden die Schnittpunkte mit der Feature-Edge-Struktur berücksichtigt. Diese befinden sich auf den Voronoi-Flächen. Die Information aus dem Schritt 1 wird hier dafür benutzt, um die auf den Voronoi-Flächen liegenden Schnittpunkte mit der Feature-Edge-Struktur korrekt miteinander zu verbinden. Um zu bestimmen, welcher Teil der Zelle im Volumen bleibt, wird am Schnittpunkt die Normale des Randgitters ausgewertet. Das Skalarprodukt der geschnittenen Kante mit dieser Normale liefert die fehlende Information.

Während der Bestimmung der Schnittpunkte der Voronoi-Kanten mit der Randfläche, werden die Nachbarn gespeichert, in denen die soeben geschnittene Voronoi-Kante auch enthalten ist. Sobald das Verschneiden einer Zelle mit der Randfläche abgeschlossen ist, springen wir zu der Nachbarzelle, die eine geschnittene Voronoi-Kante enthält. Die Prozedur

wird dann abgebrochen, wenn es keine Zellen mehr gibt, bei denen die gespeicherten Voronoi-Kanten noch nicht abgeschnitten sind.

Wir haben oben angenommen, dass es für jeden geschlossenen Flächenverbund eine Voronoi-Kante gibt, die einen Schnittpunkt mit einem zu diesem Flächenverbund zugehörigen Dreieck hat. Um so eine Kante zu bestimmen, prüfen wir die Voronoi-Kanten der Randzellen aus dem alten Gitter auf die Überschneidung mit jedem geschlossenen Flächenverbund. Während der Gitterbewegung ändern sich die Randzellen nur geringfügig, so dass die Anzahl zu prüfender Voronoi-Kanten gering ist.

In den nächsten Unterabschnitten werden wir die Schritte 1. und 3. etwas genauer beschreiben.

2.3.2 Feature-Edge-Struktur

Wie schon früher erwähnt wurde, wird die Feature-Edge-Struktur anhand der Krümmung des Randgitters erstellt. Es handelt sich dabei um den Winkel an der Kante zwischen zwei benachbarten Dreiecken. Seien \mathbf{n}_0 und \mathbf{n}_1 die Normalen zweier benachbarter Dreiecke. Für einen vordefinierten Winkel $\alpha \in [0, 90^\circ]$ ist eine Feature-Edge durch folgende Bedingung zu bestimmen:

$$\mathbf{n}_0 \cdot \mathbf{n}_1 < \cos \alpha, \quad (2.7)$$

wobei die Normalen die Länge 1 haben. Im Allgemeinen kann $\mathbf{n}_0 \cdot \mathbf{n}_1$ für ein Randgitter im Intervall $[-1, 1]$ liegen. Bei einem gültigen Randgitter sollte jedoch der Wert -1 nicht angenommen werden. Der größten bzw. der kleinsten Krümmung entspricht der Wert -1 bzw. 1 . D.h. bei dem Wert 1 ist die Fläche an der darunterliegenden Kante eben. Darüber hinaus ist es wichtig zu wissen, ob es sich genau um eine konkave oder konvexe Kante handelt. Seien \mathbf{c}_0 und \mathbf{c}_1 die geometrischen Schwerpunkte der an der Kante $\mathbf{p}_0\mathbf{p}_1$ anliegenden Dreiecke t_0 und t_1 . Sei auch \mathbf{w} die einheitliche Flächennormale des Dreiecks t_0 . Der Öffnungswinkel zwischen t_0 und t_1 kann damit wie folgt berechnet werden:

$$\mathbf{w} = \mathbf{p}_1 - \mathbf{p}_0 \quad (2.8)$$

$$\mathbf{e}_x = \frac{(\mathbf{c}_0 - \mathbf{p}_0) \cdot \mathbf{w}}{\mathbf{w} \cdot \mathbf{w}} \mathbf{w} - (\mathbf{c}_0 - \mathbf{p}_0) \quad (2.9)$$

$$\mathbf{e}_x = \frac{\mathbf{e}_x}{\|\mathbf{e}_x\|} \quad (2.10)$$

$$\mathbf{f} = (\mathbf{c}_1 - \mathbf{p}_0) - \frac{(\mathbf{c}_1 - \mathbf{p}_0) \cdot \mathbf{w}}{\mathbf{w} \cdot \mathbf{w}} \mathbf{w} \quad (2.11)$$

$$\mathbf{f} = \frac{\mathbf{f}}{\|\mathbf{f}\|} \quad (2.12)$$

$$p_x = \mathbf{f} \cdot \mathbf{e}_x \quad (2.13)$$

$$p_y = \mathbf{f} \cdot \mathbf{n}_0 \quad (2.14)$$

$$\beta = \text{atan2}(p_y, p_x). \quad (2.15)$$

Wenn β positiv ist, liegt eine konkave Kante vor. Sonst geht es um eine konvexe Kante. Den Extremfall $\beta = \pi$ soll es nicht geben, sonst müssen die benachbarten Dreiecke komplett aufeinander liegen. Die Abbildung 2.7 veranschaulicht die Rechenschritte (2.8)-(2.15). β ist der Winkel zwischen \mathbf{e}_x und \mathbf{f} im Koordinatensystem $(\mathbf{e}_x, \mathbf{n}_0, \frac{\mathbf{w}}{\|\mathbf{w}\|})$. Anhand der beschriebenen Schritte wird ein Feld erzeugt, dessen Dimension gleich der Anzahl der Kanten in dem Randgitter ist. Ein

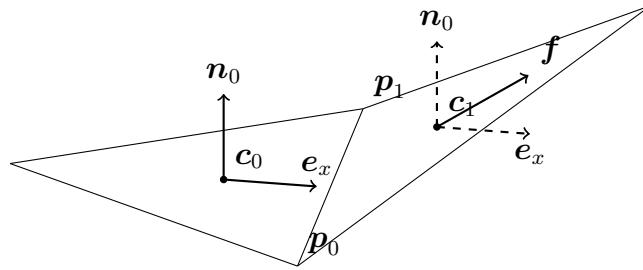


Abbildung 2.7: Öffnungswinkel zwischen zwei benachbarten Dreiecken

Eintrag dieses Feldes kann folgende Werte annehmen:

$$\begin{cases} -1 & \text{konkave Kante} \\ 1 & \text{konvexe Kante} \\ 0 & \text{keine Feature-Edge} \end{cases} \quad (2.16)$$

Ein weiterer Bestandteil der Feature-Edge-Struktur ist ein Feld, das die Verbindungen der Knoten des Randgitters zu den Kanten enthält. Jede Kante kennt ihren Start- bzw. Endpunkt. Mit Hilfe dieser Struktur können wir rekursiv durch alle Feature-Edges laufen. Um die Rekursion zu starten, benötigen wir einen Schnittpunkt einer Feature-Edge mit einer Voronoi-Fläche. Dies kann effizient mit Hilfe des alten Gitters bestimmt werden, indem man die Suchmenge nur auf die alten Randelemente begrenzt. Der Kerngedanke der rekursiven Prozedur kann wie folgt formuliert werden:

Man springe immer zur nächsten Feature-Edge und bestimme die Schnittpunkte mit allen Voronoi-Flächen, die auf dem Weg liegen. Sobald man keine noch nicht besuchte Nachbarkante mehr hat, bricht die Prozedur ab. Da die Feature-Edge-Struktur auch die sogenannten Feature-Points hat, gibt es Verzweigungen in dieser Struktur. Bei der Rekursion entscheidet man sich für den beliebigen Zweig und speichert die restlichen Zweige. Im Falle des Abbruchs wird die Rekursion an den gespeicherten Zweigen fortgesetzt. Diese Technik ist sehr effizient, so lange die Anzahl der zusammenhängenden Graphen der Feature-Edges klein ist. In den meisten Geometrien ist diese Voraussetzung gegeben.

Im weiteren wird beschrieben, wie die Schnittpunkte einer Kante mit den Flächen einer Voronoi-Zelle berechnet werden können. Diese Aufgabe wird auf die Berechnung der Schnittpunkte einer durch diese Kante verlaufenden Gerade $\mathbf{l}(\tau) := \mathbf{a} + \tau\mathbf{b}$ mit Flächen einer Voronoi-Zelle zurückgeführt. Seien \mathbf{p}_0 und \mathbf{p}_1 die Punkte, die eine Feature-Edge erzeugen. Dann ist:

$$\begin{aligned} \mathbf{a} &= \mathbf{p}_0, \\ \mathbf{b} &= \mathbf{p}_1 - \mathbf{p}_0. \end{aligned}$$

Jedem Schnittpunkt entspricht ein Parameter τ_i . Anhand von τ_i wird entschieden, ob der jeweilige Schnittpunkt auf der zu untersuchten Kante liegt. Da die Voronoi-Regionen des einhüllenden Quaders konvex sind, kann es höchstens zwei Schnittpunkte einer Gerade mit Flächen einer dieser Regionen geben. Es sind dabei drei Fälle zu unterscheiden. Es gibt

- genau zwei Schnittpunkte;
- nur einen Schnittpunkt, wenn die Gerade außerhalb der Voronoi-Region liegt und eine Voronoi-Kante oder einen Voronoi-Knoten berührt;
- keine Schnittpunkte.

Kapitel 2. Voronoi-Diagramme nicht konvexer Gebiete

Für die Berechnung der Schnittpunkte einer Geraden $l(\tau)$ mit der Oberfläche einer Voronoi-Zelle Ω könnte man sich folgende Strategie überlegen. Man trianguliere diese Oberfläche $\partial\Omega$ und berechne die Schnittpunkte von $l(\tau)$ mit allen Dreiecken von $\partial\Omega$. Dabei ist die Punktmenge jedes Dreiecks durch

$$t(\alpha, \beta) := \{w + \alpha u + \beta v : 0 \leq \alpha \leq 1, 0 \leq \beta \leq 1, \alpha + \beta \leq 1\} \quad (2.17)$$

gegeben, wobei $u, v \in \mathbb{R}^3$ seine aufspannenden Vektoren sind. Um einen Schnittpunkt von $l(\tau)$ mit $t(\alpha, \beta)$ zu berechnen, muss das Gleichungssystem

$$\alpha u + \beta v - \tau b = a - w \quad (2.18)$$

gelöst werden. Wenn die Parameter die Bedingungen $0 \leq \alpha \leq 1, 0 \leq \beta \leq 1, \alpha + \beta \leq 1$ erfüllen, haben wir das Dreieck gefunden, das einen Schnittpunkt mit $l(\tau)$ enthält.

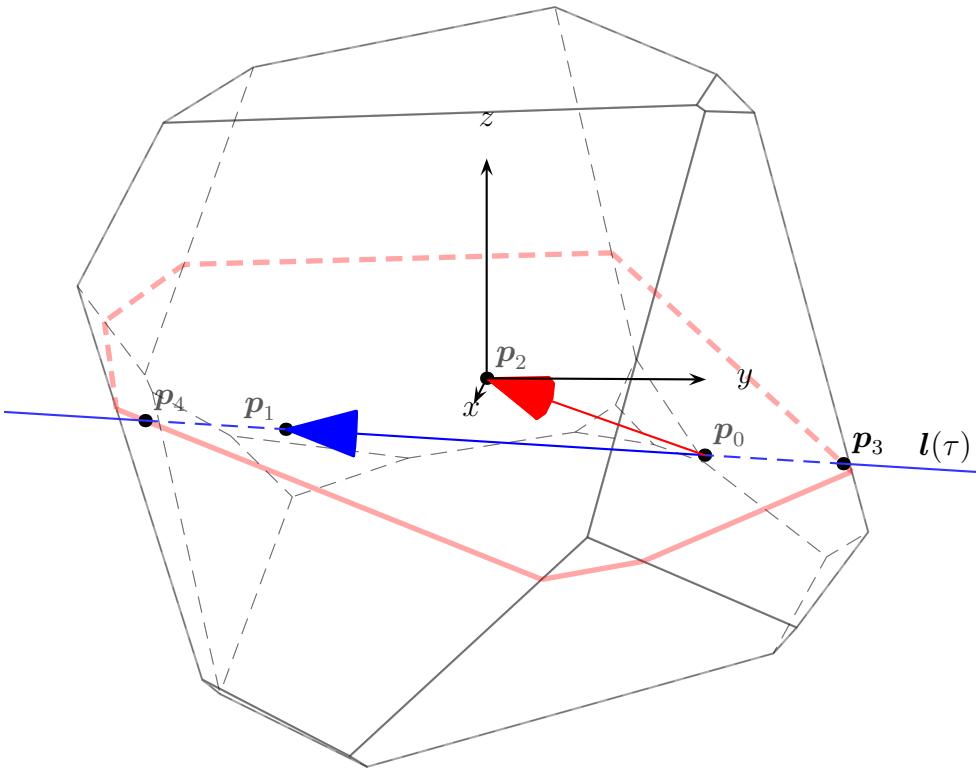


Abbildung 2.8: Schnittpunkte einer Geraden mit einem konvexen Polyeder. Das Liniensegment ist durch die Punkte p_0 und p_1 gegeben. Die Schnittebene wird mithilfe des inneren Punktes p_2 konstruiert. Der Schnitt dieser Ebene mit dem Rand der dargestellten Voronoi-Zelle erzeugt einen Polygonzug, der in orange gekennzeichnet ist. p_3 und p_4 sind die Schnittpunkte der Gerade $l(\tau)$ mit dem Polygonzug.

Eine Voronoi-Region hat durchschnittlich 12 Flächen. Jede Voronoi-Fläche hat durchschnittlich 6 Kanten. Eine sechs-kantige Voronoi-Fläche kann effizient in 4 Dreiecke zerlegt werden. D.h. die Oberfläche einer Voronoi-Zelle enthält durchschnittlich 48 Dreiecke, für die das Gleichungssystem in (2.18) gelöst werden muss. Dies ist mit einem großen Rechenaufwand verbunden. Dabei ist zu beachten, dass eine Voronoi-Zelle mehrere Feature-Edges enthalten kann, wenn das Randgitter feiner als das Volumengitter aufgelöst ist.

In dieser Arbeit wird ein wesentlich effizienterer Ansatz für die Bestimmung der Schnittpunkte von $\mathbf{l}(\tau)$ und dem Rand einer Voronoi-Region $\partial\Omega$ vorgestellt. Es wird innerhalb Ω ein Hilfspunkt \mathbf{p}_2 gesucht. Durch $\mathbf{l}(\tau)$ und \mathbf{p}_2 wird eine Ebene E gelegt. Danach wird $\partial\Omega$ mit dieser Ebene geschnitten. Dadurch entsteht ein ebener, geschlossener und konvexer Polygonzug. Die Schnittpunkte dieses Polygonzugs mit $\mathbf{l}(\tau)$ stellen die Lösung dar. Der Schnitt von E mit $\partial\Omega$ kann genau so, wie in dem Unterabschnitt 2.2.1 beschrieben ist, sehr effizient durchgeführt werden.

Die Abbildung 2.8 veranschaulicht den beschriebenen Ansatz. Die Idee dieses Ansatzes liegt offensichtlich in einer Art Reduktion der Suchmenge. Die in diesem Unterabschnitt beschriebenen Rechenschritte sind im Algorithmus 2.2 zusammengefasst.

Algorithmus 2.2 Rekursiver Algorithmus zur Bestimmung der Schnittpunkte einer Feature-Edge-Struktur mit einem Voronoi-Gitter

Initialisiere ein binäres Feld, das die Einträge für die bereits besuchten Feature-Edges enthält.

do

Bestimme eine Feature-Edge E , deren Start- bzw. Endpunkt in einer Voronoi-Region Ω liegt.

while do

Den in der Zelle Ω liegenden Punkt bezeichnen wir mit \mathbf{p}_0 . \mathbf{p}_1 ist der zweite Punkt, durch den die Feature-Edge begrenzt ist.

Berechne Schnittpunkte der durch die Feature-Edge verlaufenden Gerade $\mathbf{l}(\tau) := \mathbf{a} + \tau\mathbf{b}$ mit Ω , wobei $\mathbf{a} := \mathbf{p}_0$ und $\mathbf{b} := \mathbf{p}_1 - \mathbf{p}_0$ sind.

Projiziere \mathbf{p}_0 und \mathbf{p}_1 auf $\mathbf{l}(\tau)$. Für einen Punkt \mathbf{p} führt die Projektion zu: $\tau = \frac{\mathbf{b} \cdot (\mathbf{p} - \mathbf{a})}{\mathbf{b} \cdot \mathbf{b}}$. Für \mathbf{p}_0 und \mathbf{p}_1 erhält man damit τ_0 und τ_1 .

Berechne $\tau_{max} = \max(\tau_0, \tau_1)$

if $\tau_{max} \geq 1$ **then**

Speichere diesen Schnittpunkt für Ω und für den Nachbarn Ω_{next} , der zu dem Schnittpunkt zugehörige Fläche enthält.

Setze $\Omega = \Omega_{next}$.

else

Markiere E als besucht. Gehe durch die Verbindung von \mathbf{p}_1 und suche nach einer nicht besuchten Feature-Edge E_{next} . Speichere die restlichen nicht besuchten Feature-Edges von \mathbf{p}_1 und den Index der Zelle Ω in den dynamischen Listen L_e und L_c .

if Der Punkt \mathbf{p}_1 hat mehr als zwei Verbindungen. **then**

Addiere \mathbf{p}_1 als Feature-Point zu Ω .

end if

if Wenn alle Verbindungen von \mathbf{p}_1 besucht sind. **then**

Hole E_{next} und Ω aus den Listen L_e und L_c .

end if

Setze $E = E_{next}$.

end if

end while

while Es gibt noch nicht besuchte Feature-Edges.

Bemerkung 2.1 (Zeitkomplexität des Algorithmus 2.2):

Wenn das Randgitter feiner als das Volumengitter aufgelöst ist, enthält jede Voronoi-Zelle mehrere Feature-Edges. Für jede Feature-Edge müssen Schnittpunkte mit der entsprechenden Zelle

Kapitel 2. Voronoi-Diagramme nicht konvexer Gebiete

berechnet werden. Für M Feature-Edges erhält man dann die Zeitkomplexität $\mathcal{O}(M)$. Wenn das Randgitter größer als das Volumengitter aufgelöst ist, dann muss für jede Voronoi-Zelle die Schnittpunkte mit der entsprechenden Feature-Edge berechnet werden. Für N Voronoi-Zellen, die eine Feature-Edge enthalten, ist die Zeitkomplexität $\mathcal{O}(N)$. Zusammenfassend erhält man die Zeitkomplexität $\mathcal{O}(\max(N, M))$.

2.3.3 Randgitter

In diesem Unterabschnitt wird beschrieben, wie in dieser Arbeit die Schnittpunkte der Voronoi-Kanten mit einer triangulierten Fläche (Randgitter) bestimmt werden. Für jeden geschlossenen Flächenverbund muss zumindest eine Voronoi-Kante gefunden werden, die einen Schnittpunkt mit diesem Flächenverbund hat. Mit dieser Voronoi-Kante und einer diese Kante enthaltenden Voronoi-Region Ω_i kann die im Unterabschnitt 2.3.1 beschriebene rekursive Prozedur gestartet werden. Bei dieser Prozedur werden die Schnittpunkte der Voronoi-Kanten von V_i mit dem Randgitter berechnet und mit ihren zugehörigen Voronoi-Regionen in einem Feld abgespeichert. Darüber hinaus wird zu jeder Voronoi-Kante auch der entsprechende Dreiecksindex des Randgitters gespeichert.

Wir beschäftigen uns nun damit, wie jede Voronoi-Region mit dem Randgitter verschnitten wird. Im weiteren Verlauf dieses Unterabschnittes beziehen sich die Begriffe Voronoi-Fläche, Voronoi-Kante und Voronoi-Knoten auf eine einzige Voronoi-Region. Vor dem Verschneiden werden aus dem Speicher die geschnittene Voronoi-Kante E_0 , der Schnittpunkt P_0 , das Dreieck D_0 und die zugehörige Voronoi-Region Ω_i geholt.

Wir erklären die wichtigsten Schritte des Algorithmus anhand der Abbildung 2.9a. Jede zu Ω_i zugehörige Voronoi-Fläche kann durch zwei Voronoi-Kanten eindeutig zugeordnet werden. Jede Voronoi-Kante beispielsweise $E(11, 10)$ wird durch zwei Voronoi-Knoten 11 und 10 beschrieben. Angenommen $E(11, 10)$ ist die Voronoi-Kante, die den gespeicherten Schnittpunkt $P_0 = P(11, 10)$ mit dem Randgitter hat. $P(11, 10)$ ist der erste Schnittpunkt, mit dem das Verschneiden initialisiert wird. Der nächste Schnittpunkt $P(2, 10)$ liegt auf der Kante $E(2, 10)$. Die Voronoi-Fläche, die $E(11, 10)$ und $E(2, 10)$ enthält, bezeichnen wir durch $F(11, 10, 0)$.

Die Idee des Algorithmus besteht darin, dass wir von $P(11, 10)$ zu $P(2, 10)$ entlang des Randgitters gehen, so dass dabei alle die Voronoi-Fläche $F(11, 10, 0)$ schneidenden Dreiecke und damit auch alle diese Fläche schneidenden Feature-Edges besucht werden. Die Schnittpunkte mit den Feature-Edges werden als Feature-Points in die Voronoi-Region eingefügt. Nach dem Abschluss des Schnittes von Ω_i mit dem Randgitter werden diese Feature-Points, wie es später beschrieben wird, miteinander verbunden. Ergebnis des Schnittes $F(11, 10, 0)$ mit dem Randgitter ist ein ebener und offener Polygonzug PZ . Jeder Knoten von PZ ist ein Schnittpunkt eines Dreiecks des Randgitters mit der durch $F(11, 10, 0)$ verlaufenden Ebene. Ausnahmen sind der Start- bzw. der Endknoten von PZ (siehe die Abb. 2.9a). Diese Knoten stimmen mit $P(11, 10)$ und $P(2, 10)$ überein. Einige Knoten von PZ sind die bereits erwähnten Feature-Points.

Damit besteht das weitere Problem in der Bestimmung des Polygonzugs PZ . Wir fangen mit dem ersten Dreieck D_0 an. Der Schnitt des Dreiecks D_0 mit der Ebene durch $F(11, 10, 0)$ liefert das Segment S_0 , das durch die Punkte P_s und P_e begrenzt ist. Nun muss entschieden werden, welcher dieser Punkte der nächste Knoten des zu berechnenden Polygonzugs PZ ist. Diesen nächsten Knoten bezeichnen wir als K_i . Beim Dreieck D_0 ist es K_0 . Beim D_0 , muss K_0 in dem durch die Kanten $E(11, 10)$ und $E(11, 0)$ aufgespannten Unterraum liegen. Mit der Kenntnis von K_0 können wir auch das nächste Dreieck des Randgitters bestimmen, dessen Segment zu PZ gehört. Dieses Dreieck muss die Kante des Randgitters enthalten, auf der K_0 liegt.

Angenommen wir haben das nächste Dreieck D_1 bestimmt. Wir berechnen den Schnitt von D_1 mit der Ebene durch $F(11, 10, 0)$. Dadurch erhält man wieder zwei Punkte P_s und P_e . Ei-

ner dieser Punkte ist der nächste Knoten K_1 des Polygonzugs PZ . Einer dieser Punkte stimmt jedoch mit K_0 überein. Durch einen einfachen Vergleich von P_s und P_e mit K_0 können wir K_1 bestimmen. Somit springen wir immer wieder zum nächsten Dreieck, solange das aktuelle Liniensegment durch P_s und P_e die Voronoi-Kante $E(2, 10)$ nicht schneidet. Wenn das der Fall ist, liegt der Schnittpunkt der Voronoi-Kante $E(2, 10)$ mit dem Randgitter vor. Eine wichtige Voraussetzung für die Durchführbarkeit des entwickelten Algorithmus ist, dass jede Kante des Randgitters höchstens zu zwei Dreiecken gehört. Danach gehen wir zur nächsten Voronoi-Fläche, die von $F(11, 10, 0)$ verschieden ist und die geschnittene Kante $E(2, 10)$ enthält.

Nach dem Bestimmen aller Schnittpunkte, müssen die Feature-Points noch miteinander verbunden werden. Hierzu verwenden wir die Feature-Edge-Struktur, die in dem Unterabschnitt 2.3.2 beschrieben wurde. In dieser Struktur sind die Verbindungen der Feature-Points zueinander für jede Voronoi-Zelle gespeichert.

Die Abbildung 2.9b zeigt eine durch die hier beschriebene Methode abgeschnittene Voronoi-Zelle. Ein großer Vorteil des hier beschriebenen Algorithmus besteht darin, dass dieser bei allen beliebigen Konstellationen zwischen dem Randgitter und den Voronoi-Regionen durchführbar ist. Eine Voronoi-Kante kann beispielsweise mehrmals durch eine Fläche geschnitten werden. Außerdem werden alle Feature-Edges und Feature-Points in der endgültigen Zelle abgebildet.

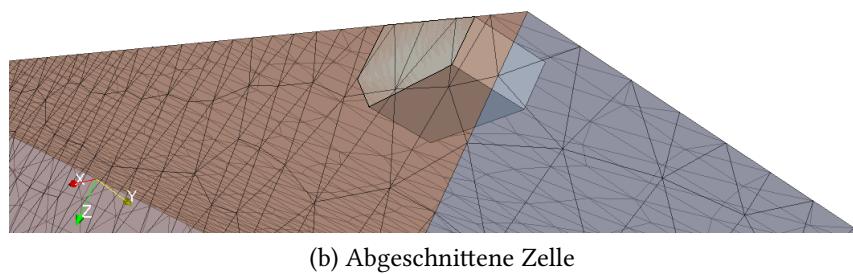
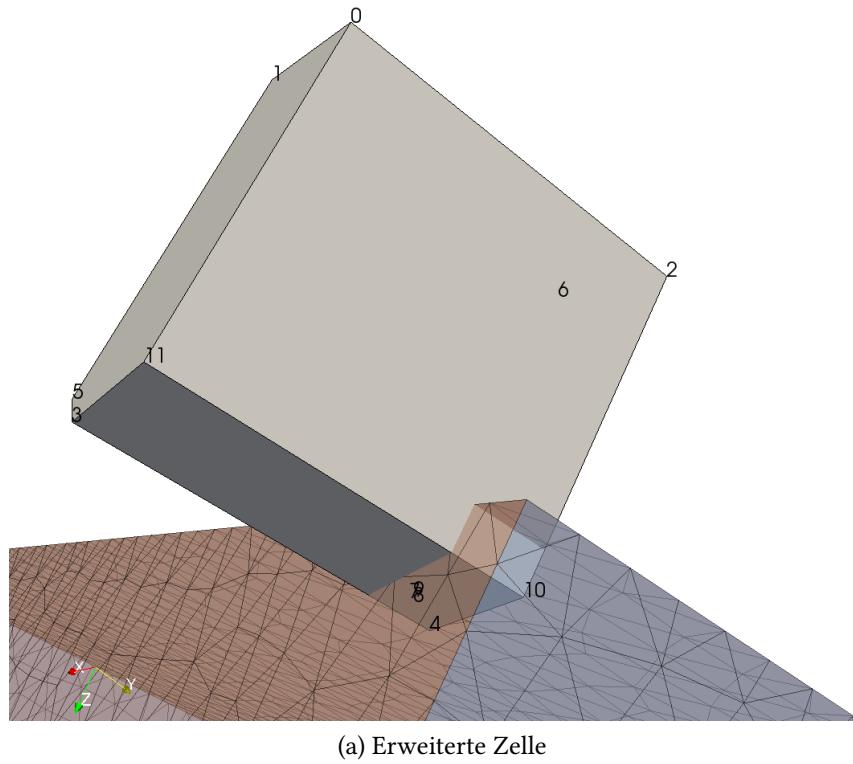


Abbildung 2.9: Verschneiden einer Voronoi-Zelle mit dem Randgitter. Die Sicht von außen. Die Nummern auf dem Bild entsprechen den Indizes der Voronoi-Knoten.

2.4 Konstruktion globaler Voronoi-Knoten

In den Abschnitten 2.2, 2.3 wurde beschrieben, wie das Rechengebiet in eine Menge von paarweise disjunkten Voronoi-Regionen unterteilt wird. Wie im Unterabschnitt 2.2.1 beschrieben,

werden diese Voronoi-Regionen als Menge von geordneten Knoten-zu-Knoten-Verbindungen gespeichert, wodurch sich die Flächen rekonstruieren lassen. Die Indizes der Knoten beziehen sich auf die einzelnen Voronoi-Regionen und haben daher eine lokale Bedeutung. Aus diesem Grund wird eine Methode benötigt, die aus diesen Regionen ein globales Gitter konstruiert. Die globalen Voronoi-Flächen können leicht erzeugt werden, da diese zu genau zwei Voronoi-Regionen gehören. Um die globalen Voronoi-Knoten zu kreieren, gehen wir durch die Knoten der zugehörigen lokalen Voronoi-Flächen und bestimmen eine Korrelation zwischen diesen Knoten. Diese Voronoi-Flächen enthalten die Indizes der lokalen Voronoi-Knoten. Wir vergeben den zugehörigen lokalen Voronoi-Knoten in beiden Zellen den einheitlichen globalen Knotenindex. Bei den noch nicht besuchten lokalen Voronoi-Knoten wird der globale Knotenindex erhöht. Nachdem diese Prozedur abgeschlossen ist, besitzt jeder der lokalen Voronoi-Knoten einen globalen Knotenindex. Danach wird ein Feld für die Positionen der neuen globalen Knoten erzeugt, das die Länge des aktuellen globalen Knotenindexes hat. Man geht durch die lokalen Voronoi-Regionen und ordnet den globalen Knoten ihre entsprechenden Positionen zu.

Diese Strategie hat Erfolg, solange die zugehörigen lokalen Voronoi-Flächen gleich sind. Als Ergebnis der Rundungsfehler kann es dazu kommen, dass es Voronoi-Nachbarn gibt, deren lokale auf dem Bisektor liegende Voronoi-Flächen ungleich sind, d.h. sowohl die Anzahl als auch die Position der Voronoi-Knoten in den beiden Flächen kann sich unterscheiden. Bei unterschiedlicher Anzahl der Knoten greifen wir auf folgende Strategie zurück. Aus der Voronoi-Zelle mit der höheren Anzahl der Voronoi-Knoten werden die redundanten Knoten anhand folgender Bedingung entfernt. Die erzeugte Korrelation zwischen zwei Mengen von Knoten liefert für jede Menge eine Liste mit dem Knotenindex der anderen Menge. Darüber hinaus werden auch die Abstände zwischen zwei zugehörigen Knoten berechnet und gespeichert. In der längeren Liste werden die doppelten Einträge gesucht und der Eintrag mit dem größeren Abstand zum Entfernen markiert. Dies wird solange wiederholt, bis es keine mehrfach auftretenden Einträge gibt. Anschließend werden aus der größeren Knotenmenge die Knoten an den markierten Stellen entfernt.

Nach der Entfernung sind die zugehörigen Voronoi-Flächen gleich. Da ein lokaler Voronoi-Knoten zu mindestens drei lokalen Voronoi-Flächen gehört, können bei der Entfernung dieser Knoten auch andere lokale Voronoi-Flächen modifiziert werden. Diese Flächen müssen auch mit den zugehörigen lokalen Voronoi-Flächen verglichen werden, und gegebenenfalls korrigiert werden.

Die Rundungsfehler können nicht nur zu unterschiedlichen Voronoi-Flächen führen, sondern auch zu den Fällen, wo ein Nachbar die zugehörige Voronoi-Fläche nicht besitzt. Solche freie Voronoi-Flächen sind sehr klein und werden auch entfernt. D.h. diese werden entweder zu einer Kante oder zu einem Knoten zusammengeführt. Als Zielkante wird die längste Kante in der Fläche ausgewählt. Wenn die Länge der längsten Kante eine Toleranz unterschreitet, wird auch diese entfernt, so dass anstelle einer Fläche nur ein Knoten verbleibt.

Jede Entfernung eines Voronoi-Knotens erzeugt unebene Voronoi-Flächen. Die Unebenheiten sind jedoch vernachlässigbar, da die unterschiedlichen Flächen von zwei benachbarten Voronoi-Zellen ausschließlich durch Rundungsfehler verursacht werden.

2.5 Problematik der Randelemente

In den vorherigen Abschnitten wurde beschrieben, wie das Voronoi-Diagramms eines Teilgebiets von \mathbb{R}^3 in dieser Arbeit bestimmt wird, wenn eine Tessellierung dieses Teilgebiets bereits existiert. Zu den in diesem Teilgebiet verteilten Generatoren wird das zugehörige Voronoi-Diagramm so erzeugt, dass die Vereinigung seiner Voronoi-Regionen den die Generatoren ein-

Kapitel 2. Voronoi-Diagramme nicht konvexer Gebiete

hüllende Quader liefert. Beim Verschneiden dieses Voronoi-Diagramms mit dem das Definitionsgebiet begrenzenden Randgitter gibt es jedoch einige kritische Situationen, die im folgenden aufgelistet sind.

1. Die Randzellen können sehr kleine Randflächen besitzen. Dies verschlechtert die Qualität des erzeugten Gitters.
2. Im Bereich konkaver Randkanten gibt es Voronoi-Zellen, die nicht zusammenhängend sind. D.h. es entstehen mehrere disjunkte Volumina, die zu dem gleichen Generator gehören.
3. Nicht konvexe Voronoi-Regionen auf dem Rand können einen Nachbarn doppelt besitzen. D.h. zwischen zwei Nachbarn kann es zwei gemeinsame Flächen geben, die vor dem Verschneiden mit dem Randgitter Teil einer einzigen Voronoi-Fläche waren.

Für die zwei letzten Punkte werden wir uns Beispiele anschauen. Die Abbildung 2.10 zeigt zwei Volumina, die zu einem Generator gehören (blaues Gebiet). Darüber hinaus enthält in grün abgebildete Zelle zwei unterschiedliche Flächen, die jedoch dem gleichen Nachbarn entsprechen.

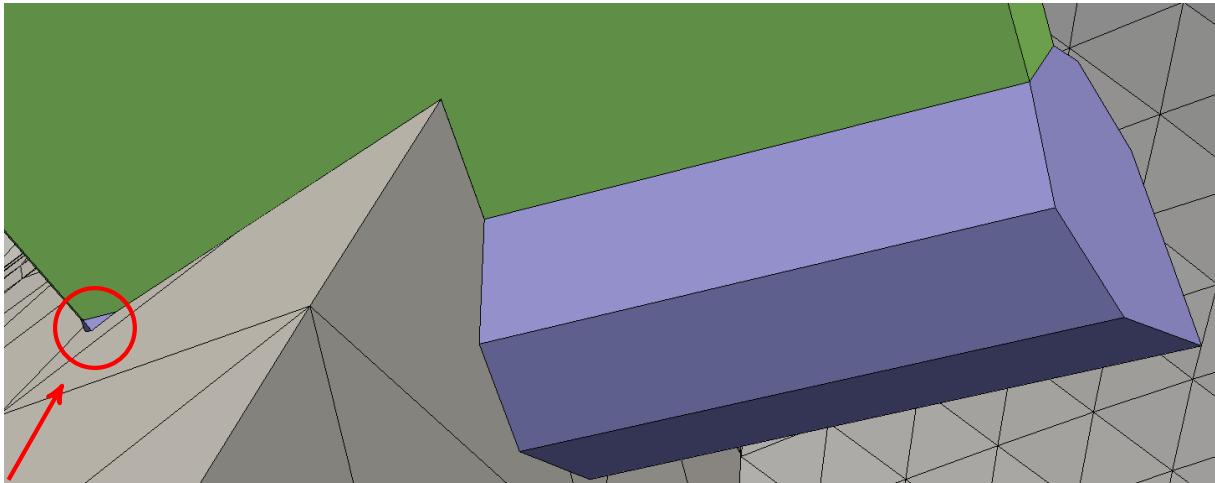
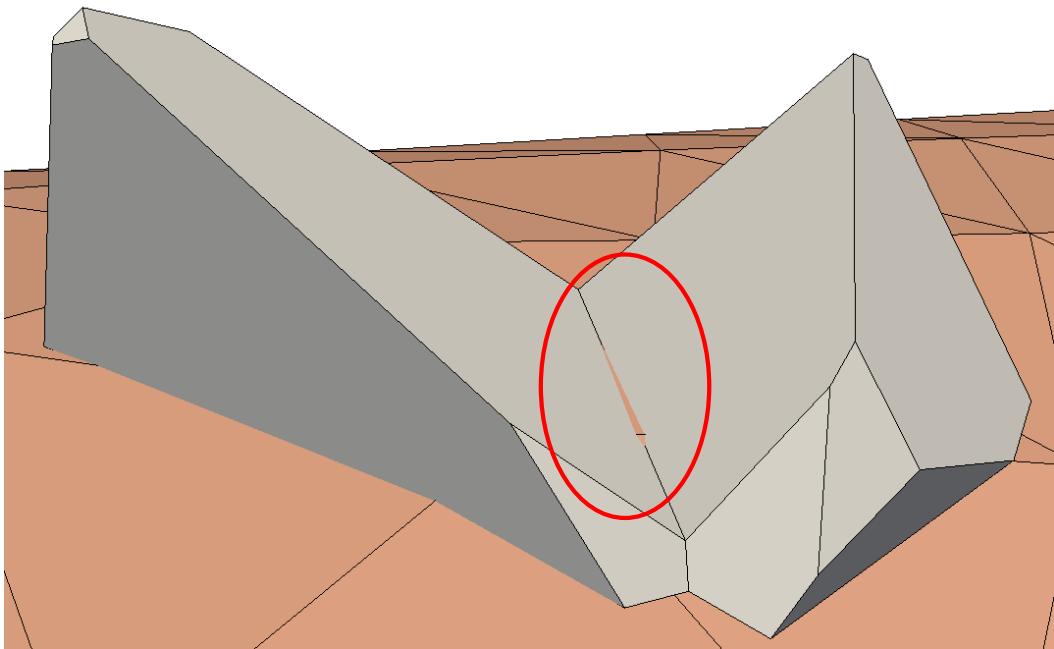


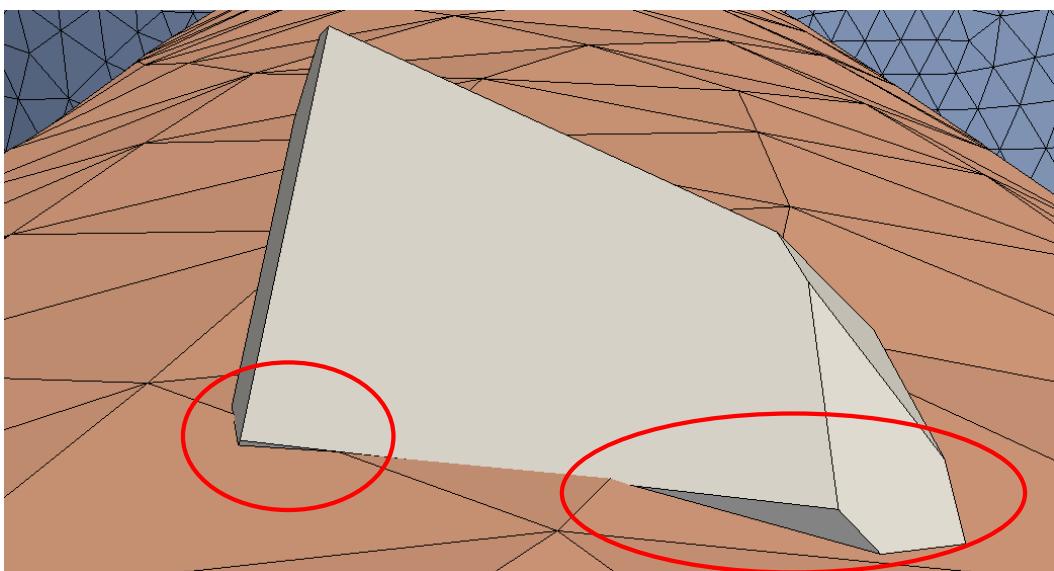
Abbildung 2.10: Zwei Volumina einer Voronoi-Region (in blau). Die Nachbarzelle ist in grün abgebildet. Das Randgitter hat graue Farbe.

Die Abbildung 2.11 zeigt zwei Zellen, die zwei gemeinsame Flächen besitzen. Vor dem Schnitt mit dem Randgitter war dies nur eine gemeinsame Fläche. Man kann sicherlich noch viele andere Beispiele finden, bei denen nicht konvexe Gebiete auch zu nicht konvexen Zellen führen. So kann beispielsweise eine Voronoi-Region in mehr als zwei Volumina zerlegt werden. Man kann sich auch einen Fall vorstellen, bei dem eine Voronoi-Region von einer konischen Randfläche durchstochen wird, so dass darin ein Hohlraum entsteht.

Die beschriebene Problematik röhrt offensichtlich aus der Tatsache her, dass die Anzahl der Zellen an den konkaven Stellen unzureichend ist. Spaltung einer Voronoi-Region (Abb. 2.10) bzw. einer Voronoi-Fläche (Abb. 2.11) könnte dadurch vermieden werden, dass man das separierte Volumen bzw. die abgetrennte Fläche einem zusätzlichen Generator zuordnet. Dieser würde den entsprechenden Teil des Raumes an sich binden. Es besteht eine gewisse Wahrscheinlichkeit, dass nach einem Hinzufügen eines Generators ein unerwünschter Fall zwar vermieden wird, jedoch dadurch ein weiteres und ähnliches Problem an einer anderen Stelle entsteht. Nach dem Erweitern der gespaltenen Zellen durch zusätzliche Generatoren muss eine weitere Prüfung erfolgen, bei der wiederum die gespaltenen Volumen- bzw. Flächenelemente identifiziert und mit



(a) Zwei Voronoi-Regionen, die über zwei disjunkte Flächen benachbart sind.



(b) Zwei Flächen einer Voronoi-Region, die jedoch zum gleichen Bisektor gehören.

Abbildung 2.11: Voronoi-Nachbarn mit zwei gemeinsamen Flächen. Das Randgitter ist in orange dargestellt. Die Voronoi-Regionen sind in grauer Farbe abgebildet.

Kapitel 2. Voronoi-Diagramme nicht konvexer Gebiete

zusätzlichen Generatoren erweitert werden. Dies wird solange wiederholt, bis alle Volumen- bzw. Flächenelemente zusammenhängend sind. Die Praxis hat gezeigt, dass dieses iterative Vorgehen endlich ist. Im Regelfall benötigt man nicht mehr als drei Iterationen. Bei jeder Iterationen reicht es auch, nur lokal das Voronoi-Diagramm neu zu erzeugen.

Bemerkung 2.1:

Es empfiehlt sich nicht, zusätzliche Generatoren während einer CFD-Simulation hinzuzufügen. Denn hierdurch wäre es notwendig die Strömungskenngrößen wie Druck und Geschwindigkeit zu interpolieren. Im Unterabschnitt 3.7.3 werden Randbedingungen für Generatoren beschrieben, mit dessen Hilfe sich Änderungen an den Randflächen der Volumenelemente bei Verschiebungen des Randgitters vermeiden lassen.

3 Gitterbewegung mithilfe von CVT

In Anlehnung an [19, 59, 60, 61, 62, 64] wird in diesem Kapitel der Begriff Centroidal-Voronoi-Tessellierung (CVT) diskutiert, der aus den englischen Literaturquellen stammt. Die äquivalente deutsche Bezeichnung ist das Centroidal-Voronoi-Diagramm, die jedoch in der deutschen Literatur selten verwendet wird.

Es wird außerdem eine fundamentale Eigenschaft der Centroidal-Voronoi-Diagramme gezeigt, die darin besteht, dass diese Gitter eine Art globale Optimalität besitzen. Darüber hinaus werden mathematischen Methoden diskutiert, die zur Konstruktion der CVT verwendet werden können. Im Anschluss wird beschrieben, wie die Gitterbewegung in dieser Arbeit umgesetzt worden ist.

3.1 Der Begriff Centroidal-Voronoi-Tessellierung

Wir beschränken uns auf Centroidal-Voronoi-Tessellierungen im \mathbb{R}^3 . Für den allgemeinen Fall kann man in [64] nachlesen. Zuerst führen wir einige Definitionen auf. Wir betrachten dabei Tessellierungen abgeschlossener Gebiete.

Definition 3.1 (Tessellierung). *Sei $\Omega \subset \mathbb{R}^3$ abgeschlossen, die Menge $\{\Omega_i\}_{i=1}^N$ mit $\Omega_i \subseteq \Omega$ wird als Tessellierung von Ω bezeichnet, wenn $(\Omega_i \setminus \partial\Omega_i) \cap (\Omega_j \setminus \partial\Omega_j) = \emptyset$ für $i \neq j$ und $\cup_{i=1}^N \Omega_i = \Omega$.*

Definition 3.2 (Voronoi-Tessellierung). *Seien die Voraussetzungen wie in Definition 3.1 erfüllt. Sei $\{\mathbf{x}_i\}_{i=1}^N$ mit $\mathbf{x}_i \in \Omega$. Ein Voronoi-Gebiet $\hat{\Omega}_i \subseteq \Omega$ zugehörig zu \mathbf{x}_i ist wie folgt definiert:*

$$\hat{\Omega}_i \setminus \partial\hat{\Omega}_i = \{\mathbf{x} \in \Omega \mid \|\mathbf{x}_i - \mathbf{x}\| < \|\mathbf{x}_j - \mathbf{x}\| \text{ für } j = 1, \dots, N \text{ mit } j \neq i\}. \quad (3.1)$$

Die Voronoi-Tessellierung ist eine Tessellierung $\{\hat{\Omega}_i\}_{i=1}^N$ von Ω , wobei für jedes $\hat{\Omega}_i$ die Gleichung (3.1) erfüllt ist. Die Punkte \mathbf{x}_i nennt man Generatoren zu den Voronoi-Gebieten $\hat{\Omega}_i$.

Definition 3.3 (Centroidal-Voronoi-Tessellierung). *Seien die Voraussetzungen wie in Definition 3.2 erfüllt. Es seien außerdem eine Voronoi-Tessellierung $\{\hat{\Omega}_i\}_{i=1}^N$ von Ω und eine stetig differenzierbare Zeldichte-Funktion $\rho(\mathbf{x})$, die auf Ω definiert ist, gegeben. Ein Centroidal-Voronoi-Gebiet Ω_i^* ist ein Voronoi-Gebiet, dessen Generator mit seinem Schwerpunkt übereinstimmt, d.h.:*

$$\mathbf{x}_i = \int_{\Omega_i} \rho(\mathbf{x}) \mathbf{x} d\mathbf{x} \Bigg/ \int_{\Omega_i} \rho(\mathbf{x}) d\mathbf{x}, \quad (3.2)$$

wobei $\int_{\Omega_i} d\mathbf{x}$ als ein Dreifachintegral über Ω_i nach allen drei Ortsrichtungen x^1, x^2 und x^3 zu verstehen ist. Im weiteren soll der Index oben immer die entsprechende Ortsrichtung betonen. Eine Centroidal-Voronoi-Tessellierung ist eine Voronoi-Tessellierung mit der Eigenschaft, dass jedes Gebiet $\hat{\Omega}_i$ die Gleichung (3.2) erfüllt. Die Zeldichte bezeichnet eine Größe, die die Anzahl der Zellen pro Volumeneinheit wiedergibt. Im Unterabschnitt 3.2.3 wird eine präzisere Definition der Zeldichte angeboten.

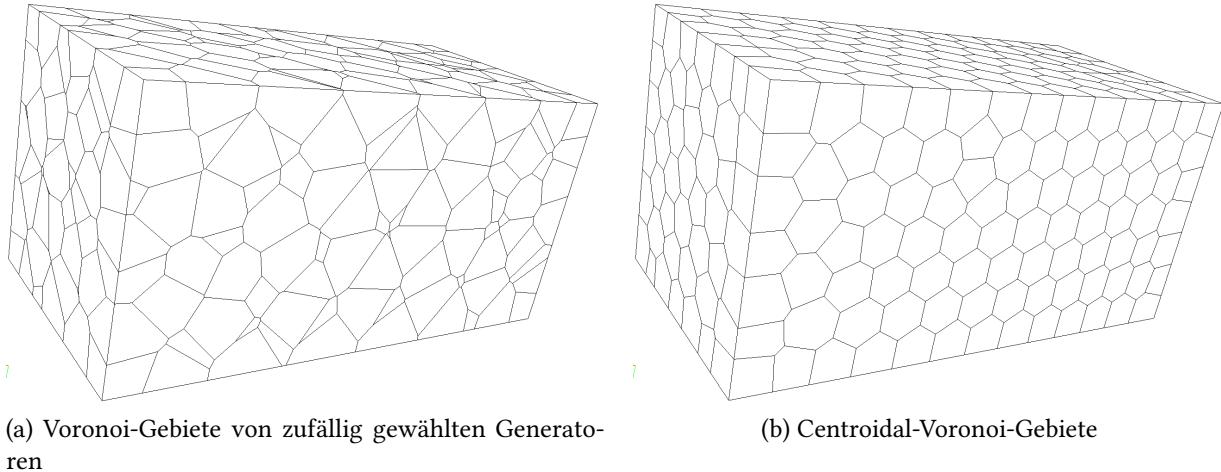


Abbildung 3.1: Vergleich zwischen einem Voronoi-Gitter und einem Centroidal-Voronoi-Gitter. Die Anzahl der Zellen ist bei beiden Gittern gleich.

Anhand der Abbildung 3.1 wird es deutlich, dass ein CV-Gitter wesentlich regulärer als ein Voronoi-Gitter aus zufällig erzeugten Generatoren ist. Die CV-Gitter haben im Allgemeinen viele Vorteile gegenüber allen möglichen diskret erzeugten polyedrischen Gittern, deren Qualität auf Basis lokaler Optimierungsschritte erreicht wird. Die CV-Gitter besitzen eine Art globale Optimalität, was im Abschnitt 3.2 untersucht wird. Diese globale Optimalität besteht darin, dass bei einer vorgegebenen Zelldichteverteilung ein CV-Gitter unter allen möglichen Gittern für alle Volumenelemente die bestmöglichen Qualitätskriterien erreicht (Siehe den Unterabschnitt 4.6.4). Diese Gitterqualitätskriterien beeinflussen die Approximationsordnung der Diskretisierungsschemata, die bei der Annäherung der örtlichen Differentialoperatoren mittels der Finite-Volumen-Methode verwendet werden. Für weitere Information wird der Leser auf das Kapitel 4 verwiesen.

3.2 Globale Optimalität von CVT

In diesem Abschnitt zeigen wir, dass die Centroidal-Voronoi-Gitter eine gewisse Optimalität besitzen. Sei $\Omega \subset \mathbb{R}^3$ abgeschlossen. Sei ferner

$$\mathcal{M} := \{(\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N)^T \quad | \quad \mathbf{y}_i \in \Omega, \text{ für } i = 1, 2, \dots, N\}. \quad (3.3)$$

Wir definieren die Funktion

$$\mathcal{F} : \mathcal{M} \mapsto \mathbb{R}, \quad \mathcal{F}(\mathcal{X}) := \sum_{i=1}^N \int_{\Omega_i} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^2 d\mathbf{x} \quad (3.4)$$

wobei $\rho(\mathbf{x})$ die Zelldichte, $\mathcal{X} := (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)^T \in \mathcal{M}$ ein Vektor mit Generatoren und $\{\Omega_i\}_{i=1}^N$ die zugehörigen Voronoi-Gebiete bezeichnen. Die Funktion in (3.4) wird in der Literatur als Fehlerfunktion der CVT bezeichnet.

3.2.1 CVT als Gradient der Fehlerfunktion

Wir zeigen in diesem Unterabschnitt, dass bei einem fest definierten Gebiet $\Omega \in \mathbb{R}^3$ die Funktion in (3.4) genau dann minimal ist, wenn die Generatoren \mathbf{x}_i für $i = 1, 2, \dots, N$ mit Schwerpunkten

ihrer zugehörigen Gebiete übereinstimmen. Hier werden einige Sätze formuliert und bewiesen. Die hier formulierten Sätze und Lemmata werden auch in weiteren Abschnitten benötigt.

Satz 3.1. Seien $f : \mathbb{R}^3 \mapsto \mathbb{R}$, $\mathbf{x} \mapsto f(\mathbf{x})$ und $\mathbf{x} : \mathbb{R} \mapsto \mathbb{R}^3$, $u \mapsto \mathbf{x}(u)$ einmal stetig differenzierbare Funktionen. Sei weiterhin $\Omega : \mathbb{R} \mapsto \mathbb{R}^3$, $u \mapsto \Omega(u)$ ein stetiges Volumenelement, dessen Grenzen von u abhängig sind. Dann gilt

$$\frac{\partial}{\partial u} \int_{\Omega(u)} f(\mathbf{x}(u)) d\mathbf{x} = \int_{\Omega(u)} \frac{\partial}{\partial u} f(\mathbf{x}(u)) d\mathbf{x}. \quad (3.5)$$

Beweis.

$$\begin{aligned} \frac{\partial}{\partial u} \int_{\Omega(u)} f(\mathbf{x}(u)) d\mathbf{x} &= \lim_{\Delta u \rightarrow 0} \frac{\int_{\Omega(u+\Delta u)} f(\mathbf{x}(u+\Delta u)) d\mathbf{x} - \int_{\Omega(u)} f(\mathbf{x}(u)) d\mathbf{x}}{\Delta u} \\ &= \lim_{\Delta u \rightarrow 0} \frac{\int_{\Omega(u)} f(\mathbf{x}(u+\Delta u)) d\mathbf{x} + \underbrace{\int_{\Omega(u+\Delta u) \setminus \Omega(u)} f(\mathbf{x}(u+\Delta u)) d\mathbf{x}}_{\rightarrow 0} - \int_{\Omega(u)} f(\mathbf{x}(u)) d\mathbf{x}}{\Delta u} \\ &= \lim_{\Delta u \rightarrow 0} \frac{\int_{\Omega(u)} f(\mathbf{x}(u+\Delta u)) - f(\mathbf{x}(u)) d\mathbf{x}}{\Delta u} \\ &= \lim_{\Delta u \rightarrow 0} \frac{\Omega(u)}{\Delta u} \\ &= \int_{\Omega(u)} \frac{\partial}{\partial u} f(\mathbf{x}(u)) d\mathbf{x} \end{aligned}$$

Das Volumenintegral $\int_{\Omega(u+\Delta u) \setminus \Omega(u)} f(\mathbf{x}(u+\Delta u)) d\mathbf{x}$ strebt gegen Null, da das Integrationsgebiet $\Omega(u+\Delta u) \setminus \Omega(u)$ für $\Delta u \rightarrow 0$ ebenso gegen Null strebt. \square

Lemma 3.1. Seien $f : \mathbb{R}^3 \mapsto \mathbb{R}$, $\mathbf{x} \mapsto f(\mathbf{x})$ und $\mathbf{x} : \mathbb{R} \mapsto \mathbb{R}^3$, $u \mapsto \mathbf{x}(u)$ einmal stetig differenzierbare Funktionen. Dann gilt:

$$\nabla \cdot \left(f(\mathbf{x}(u), u) \frac{\partial \mathbf{x}}{\partial u} \right) = \frac{\partial f(\mathbf{x}(u), u)}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial u}, \quad (3.6)$$

wobei $(\nabla \cdot)$ der Divergenz-Operator ist.

Beweis. Seien die Voraussetzungen wie in dem Lemma 3.1 gegeben. Sei weiterhin $\mathbf{e}^i := (0, \underset{i}{1}, 0)^T$ der i -te Basisvektor. Der Divergenz-Operator lässt sich wie folgt schreiben:

$$\nabla \cdot \left(f(\mathbf{x}(u), u) \frac{\partial \mathbf{x}}{\partial u} \right) = \sum_{l=1}^3 \frac{\partial}{\partial x^l} (\mathbf{e}^l)^T \left(f(\mathbf{x}, u) \frac{\partial \mathbf{x}}{\partial u} \right) = \sum_{l=1}^3 \frac{\partial}{\partial x^l} \left(f(\mathbf{x}, u) \frac{\partial x^l}{\partial u} \right) \quad (3.7)$$

$$= \sum_{l=1}^3 \frac{\partial f(\mathbf{x}, u)}{\partial x^l} \frac{\partial x^l}{\partial u} + f(\mathbf{x}, u) \frac{\partial^2 x^l}{\partial x^l \partial u} \quad (3.8)$$

Durch eine einmalige Anwendung des Satzes von Schwarz über partielle Ableitungen erhält man aus (3.8):

$$f(\mathbf{x}, u) \frac{\partial^2 x^l}{\partial x^l \partial u} = f(\mathbf{x}, u) \frac{\partial^2 x^l}{\partial u \partial x^l} = f(\mathbf{x}, u) \frac{\partial}{\partial u} 1 = 0 \quad (3.9)$$

Ein Einsetzen von (3.9) in (3.8) liefert:

$$\nabla \cdot \left(f(\mathbf{x}, u) \frac{\partial \mathbf{x}}{\partial u} \right) = \sum_{l=1}^3 \frac{\partial f(\mathbf{x}, u)}{\partial x^l} \frac{\partial x^l}{\partial u} = \frac{\partial f(\mathbf{x}, u)}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial u}. \quad (3.10)$$

□

Das nächste Lemma wurde in [61] formuliert. Aufgrund seiner Wichtigkeit, wird dieses hier aufgeführt.

Lemma 3.2. *Sei $\mathbf{x} \in \partial\Omega_{ij} \subset \mathbb{R}^3$ ein Punkt auf dem Flächenelement $\partial\Omega_{ij} \subset \mathbb{R}^3$, das der Bisektor von den Generatoren \mathbf{x}_i und \mathbf{x}_j ist. Dann gilt für die Ableitungen $\frac{\partial \mathbf{x}}{\partial x_i^k}$ und $\frac{\partial \mathbf{x}}{\partial x_j^k}$:*

$$\frac{\partial \mathbf{x}}{\partial x_i^k} \cdot \mathbf{n} = \frac{x^k - x_i^k}{\|\mathbf{x}_j - \mathbf{x}_i\|}, \quad \frac{\partial \mathbf{x}}{\partial x_j^k} \cdot \mathbf{n} = \frac{x_j^k - x^k}{\|\mathbf{x}_j - \mathbf{x}_i\|}, \quad \forall \mathbf{x} \in \partial\Omega_{ij}. \quad (3.11)$$

Beweis. Gelten Voraussetzungen wie in Lemma 3.2, dann muss laut der Definition der Voronoi-Diagramme folgende Gleichung erfüllt sein:

$$\left(\mathbf{x} - \frac{\mathbf{x}_i + \mathbf{x}_j}{2} \right) \cdot (\mathbf{x}_j - \mathbf{x}_i) = 0, \quad \forall \mathbf{x} \in \partial\Omega_{ij}. \quad (3.12)$$

Die Ableitungen dieser Gleichung nach x_i^k und x_j^k liefern folgende Identitäten:

$$\left(\frac{\partial \mathbf{x}}{\partial x_i^k} - \frac{1}{2} \mathbf{e}^k \right) \cdot (\mathbf{x}_j - \mathbf{x}_i) + \left(\mathbf{x} - \frac{\mathbf{x}_i + \mathbf{x}_j}{2} \right) \cdot (-\mathbf{e}^k) = 0 \quad (3.13)$$

$$\Rightarrow \frac{\partial \mathbf{x}}{\partial x_i^k} \cdot (\mathbf{x}_j - \mathbf{x}_i) = x^k - x_i^k \quad (3.14)$$

$$\left(\frac{\partial \mathbf{x}}{\partial x_j^k} - \frac{1}{2} \mathbf{e}^k \right) \cdot (\mathbf{x}_j - \mathbf{x}_i) + \left(\mathbf{x} - \frac{\mathbf{x}_i + \mathbf{x}_j}{2} \right) \cdot \mathbf{e}^k = 0 \quad (3.15)$$

$$\Rightarrow \frac{\partial \mathbf{x}}{\partial x_j^k} \cdot (\mathbf{x}_j - \mathbf{x}_i) = x_j^k - x^k \quad (3.16)$$

Eine Division der Gleichungen (3.14), (3.16) durch $\|\mathbf{x}_j - \mathbf{x}_i\|$ führt zu der Behauptung. □

Satz 3.2 (Zerlegung eines Integrals in eine Summe der Integrale über disjunkte Gebiete). *Angenommen ein abgeschlossenes Gebiet $\Omega \in \mathbb{R}^d$ mit $d \in \mathbb{N}$ lässt sich in eine Menge disjunkter Untergebiete $\{\Omega_i\}_{i=1}^M$ mit $\Omega_i \subset \Omega$ zerlegen. D.h. $\bigcap_{i=1}^M (\Omega_i \setminus \partial\Omega_i) = \emptyset$ und $\bigcup_{i=1}^M \Omega_i = \Omega$. Dann gilt für eine stetige Funktion $g : \Omega \mapsto \mathbb{R}$, $\mathbf{x} \mapsto g(\mathbf{x})$:*

$$\int_{\Omega} g(\mathbf{x}) d\mathbf{x} = \sum_{i=1}^M \int_{\Omega_i} g(\mathbf{x}) d\mathbf{x}. \quad (3.17)$$

Satz 3.3. *Sei $\Omega \subset \mathbb{R}^3$ abgeschlossen. Sei weiterhin $\{\Omega_i\}_{i=1}^N$ eine Voronoi-Tessellierung von Ω , wobei Ω_i das Voronoi-Gebiet ist, das zum Generator $\mathbf{x}_i \in \Omega$ gehört. Ferner seien $\mathcal{X} := (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)^T \subset \mathcal{M}$ der Vektor mit Generatoren und $\rho(\mathbf{x})$ eine einmal stetig differenzierbare Funktion, die auf Ω definiert ist. Für die Funktion in (3.4)*

$$\mathcal{F} : \mathcal{M} \mapsto \mathbb{R}, \quad \mathcal{F}(\mathcal{X}) = \sum_{i=1}^N \int_{\Omega_i} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^2 d\mathbf{x} \quad (3.18)$$

gilt:

$$\nabla \mathcal{F}(\mathcal{X}) = -2 \begin{pmatrix} \int_{\Omega_1} \rho(\mathbf{x})(\mathbf{x} - \mathbf{x}_1) d\mathbf{x} \\ \int_{\Omega_2} \rho(\mathbf{x})(\mathbf{x} - \mathbf{x}_2) d\mathbf{x} \\ \vdots \\ \int_{\Omega_n} \rho(\mathbf{x})(\mathbf{x} - \mathbf{x}_N) d\mathbf{x} \end{pmatrix}. \quad (3.19)$$

Beweis. Seien die Voraussetzungen wie in dem Satz 3.3 gegeben. Es ist zu zeigen, dass

$$\frac{\partial}{\partial x_i^k} \mathcal{F}(\mathcal{X}) = -2 \int_{\Omega_i} \rho(\mathbf{x})(x^k - x_i^k) d\mathbf{x}. \quad (3.20)$$

Sei \mathcal{N}_i die Menge der Nachbarindizes des i -ten Generators. Von dem Generator \mathbf{x}_i sind eigenes Gebiet Ω_i und die benachbarten Gebiete $\{\Omega_j\}_{j \in \mathcal{N}_i}$ abhängig. Damit können wir schreiben:

$$\frac{\partial}{\partial x_i^k} \mathcal{F}(\mathcal{X}) = \frac{\partial}{\partial x_i^k} \sum_{l=1}^N \int_{\Omega_l} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_l\|^2 d\mathbf{x} \quad (3.21)$$

$$= \frac{\partial}{\partial x_i^k} \int_{\Omega_i} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^2 d\mathbf{x} + \sum_{j \in \mathcal{N}_i} \frac{\partial}{\partial x_i^k} \int_{\Omega_j} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_j\|^2 d\mathbf{x} \quad (3.22)$$

Mit Anwendung des Satzes 3.1 erhalten wir aus (3.22):

$$\frac{\partial}{\partial x_i^k} \mathcal{F}(\mathcal{X}) = \underbrace{\int_{\Omega_i} \frac{\partial}{\partial x_i^k} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^2 d\mathbf{x}}_{(a)} + \underbrace{\sum_{j \in \mathcal{N}_i} \int_{\Omega_j} \frac{\partial}{\partial x_i^k} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_j\|^2 d\mathbf{x}}_{(b)} \quad (3.23)$$

Wir betrachten zuerst den Ausdruck (a) aus (3.23). Um die Ableitung nach x_i^k zu bestimmen, wird folgende Hilfsfunktion eingeführt:

$$g : \mathbb{R}^3 \times \mathbb{R}^3 \mapsto \mathbb{R}, \quad (\mathbf{u}, \mathbf{v}) \mapsto g(\mathbf{u}, \mathbf{v}). \quad (3.24)$$

Die Ableitung dieser Funktion nach v^k sieht folgendermaßen aus:

$$\frac{\partial}{\partial v^k} g(\mathbf{u}, \mathbf{v}) = \begin{pmatrix} \frac{\partial g}{\partial \mathbf{u}} & \frac{\partial g}{\partial \mathbf{v}} \end{pmatrix} \begin{pmatrix} \frac{\partial \mathbf{u}}{\partial v^k} \\ \frac{\partial \mathbf{v}}{\partial v^k} \end{pmatrix} = \frac{\partial g}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial v^k} + \frac{\partial g}{\partial \mathbf{v}} \frac{\partial \mathbf{v}}{\partial v^k} = \frac{\partial g}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial v^k} + \frac{\partial g}{\partial v^k} \quad (3.25)$$

Für $\mathbf{u} := \mathbf{x}$, $\mathbf{v} := \mathbf{x}_i$, $g(\mathbf{u}, \mathbf{v}) := \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^2$ und Anwendung des Lemmas 3.1 erhalten wir:

$$\frac{\partial}{\partial x_i^k} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^2 = \frac{\partial \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^2}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial x_i^k} + \frac{\partial \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^2}{\partial x_i^k} \quad (3.26)$$

$$= \underbrace{\nabla \cdot \left(\rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^2 \frac{\partial \mathbf{x}}{\partial x_i^k} \right)}_{(c)} - \underbrace{2\rho(\mathbf{x})(x^k - x_i^k)}_{(d)} \quad (3.27)$$

Eine Integration von (c) mit anschließender Anwendung des Gaußschen Satzes liefert:

$$\int_{\Omega_i} \nabla \cdot \left(\rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^2 \frac{\partial \mathbf{x}}{\partial x_i^k} \right) d\mathbf{x} = \int_{\partial \Omega_i} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^2 \frac{\partial \mathbf{x}}{\partial x_i^k} \cdot \mathbf{n} d\mathbf{x}, \quad (3.28)$$

wobei \mathbf{n} die äußere Flächennormale des Teilgebietes Ω_i ist. Nach einer Integration von (d) und mit (3.28) erhalten wir für (a) :

$$(a) = \int_{\partial\Omega_i} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^2 \frac{\partial \mathbf{x}}{\partial x_i^k} \cdot \mathbf{n} d\mathbf{x} - 2 \int_{\Omega_i} \rho(\mathbf{x}) (x^k - x_i^k) d\mathbf{x}. \quad (3.29)$$

Das Oberflächenintegral über $\partial\Omega_i$ in (3.29) kann man als Summe der Oberflächenintegrale über $\partial\Omega_{ij}$ mit $j \in \mathcal{N}_i$ darstellen. Damit erhalten wir aus (3.29)

$$(a) = \sum_{j \in \mathcal{N}_i} \int_{\partial\Omega_{ij}} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^2 \frac{\partial \mathbf{x}}{\partial x_i^k} \cdot \mathbf{n} d\mathbf{x} - 2 \int_{\Omega_i} \rho(\mathbf{x}) (x^k - x_i^k) d\mathbf{x} \quad (3.30)$$

Mit Anwendung des Lemmas 3.2 erhalten wir aus (3.30)

$$(a) = \sum_{j \in \mathcal{N}_i} \int_{\partial\Omega_{ij}} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^2 \frac{x^k - x_i^k}{\|\mathbf{x}_j - \mathbf{x}_i\|} d\mathbf{x} - 2 \int_{\Omega_i} \rho(\mathbf{x}) (x^k - x_i^k) d\mathbf{x} \quad (3.31)$$

Da (b) von \mathbf{x}_i unabhängig ist, erhalten wir analog:

$$(b) = \int_{\Omega_j} \nabla \cdot \left(\rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_j\|^2 \frac{\partial \mathbf{x}}{\partial x_i^k} \right) d\mathbf{x} = \int_{\partial\Omega_j} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_j\|^2 \frac{\partial \mathbf{x}}{\partial x_i^k} \cdot \mathbf{n} d\mathbf{x} \quad (3.32)$$

Das Oberflächenintegral aus der Gleichung (3.32) kann man als eine Summe der Oberflächenintegrale über die einzelnen Flächenelemente darstellen. Es gilt also:

$$(b) = \sum_{l \in \mathcal{N}_j} \int_{\partial\Omega_{jl}} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_j\|^2 \frac{\partial \mathbf{x}}{\partial x_i^k} \cdot \mathbf{n} d\mathbf{x}, \quad (3.33)$$

wobei \mathcal{N}_j die Menge der Nachbarindizes des j -ten Volumenelements sind. $\partial\Omega_{jl}$ ist entsprechend das Flächenelement zwischen dem j -ten und dem l -ten Volumenelementen. Da es sich in der Gleichung (3.33) um ein Oberflächenintegral handelt, bedeutet der Ausdruck $\frac{\partial \mathbf{x}}{\partial x_i^k} \cdot \mathbf{n}$ die Positionsänderung der Flächenpunkte \mathbf{x} wenn sich x_i^k ändert. Dabei geht es um die Änderung in Richtung der äußeren Flächennormale \mathbf{n} . Obwohl die Punkte der Flächenelemente $\partial\Omega_{jl}$ in (3.33) von x_i^k abhängig sind, bleiben diese jedoch in der gleichen Ebene, wenn die Generatoren \mathbf{x}_j und \mathbf{x}_l mit $l \neq i$ fixiert sind. D.h. $\frac{\partial \mathbf{x}}{\partial x_i^k} \cdot \mathbf{n}$ ist für $\{\mathbf{x} \in \partial\Omega_{jl} \mid l \in \mathcal{N}_j \wedge l \neq i\}$ gleich Null. Mit Berücksichtigung von Lemma 3.2 können wir damit schreiben:

$$(b) = \int_{\partial\Omega_{ji}} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_j\|^2 \frac{x_i^k - x_j^k}{\|\mathbf{x}_j - \mathbf{x}_i\|} d\mathbf{x} = - \int_{\partial\Omega_{ji}} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_j\|^2 \frac{x_i^k - x_j^k}{\|\mathbf{x}_j - \mathbf{x}_i\|} d\mathbf{x}. \quad (3.34)$$

Wir setzen (a) aus (3.31) und (b) aus (3.34) in (3.23) ein und erhalten:

$$\frac{\partial}{\partial x_i^k} \mathcal{F}(\mathcal{X}) = -2 \int_{\Omega_i} \rho(\mathbf{x}) (x^k - x_i^k) d\mathbf{x} \quad (3.35)$$

$$+ \underbrace{\sum_{j \in \mathcal{N}_i} \left(\int_{\partial\Omega_{ij}} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^2 \frac{x^k - x_i^k}{\|\mathbf{x}_j - \mathbf{x}_i\|} d\mathbf{x} - \int_{\partial\Omega_{ji}} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_j\|^2 \frac{x_i^k - x_j^k}{\|\mathbf{x}_j - \mathbf{x}_i\|} d\mathbf{x} \right)}_{(e)} \quad (3.36)$$

Es bleibt nun zu zeigen, dass (e) verschwindet. Das Integrationsgebiet $\partial\Omega_{ij}$ entspricht dem Bi-sektor bzw. dem Flächenelement zwischen den Generatoren \mathbf{x}_i und \mathbf{x}_j . Das gleiche gilt für das Integrationsgebiet $\partial\Omega_{ji}$, d.h. $\partial\Omega_{ij} = \partial\Omega_{ji}$. Damit erhalten wir aus (3.36)

$$(e) = \sum_{j \in \mathcal{N}_i \setminus \partial\Omega_{ij}} \int_{\partial\Omega_{ij}} \rho(\mathbf{x}) \frac{\mathbf{x}^k - \mathbf{x}_i^k}{\|\mathbf{x}_j - \mathbf{x}_i\|} (\|\mathbf{x} - \mathbf{x}_i\|^2 - \|\mathbf{x} - \mathbf{x}_j\|^2) d\mathbf{x}. \quad (3.37)$$

Der Ausdruck $(\|\mathbf{x} - \mathbf{x}_i\|^2 - \|\mathbf{x} - \mathbf{x}_j\|^2)$ muss gleich Null sein, da die Abstände von einem Punkt $\mathbf{x} \in \partial\Omega_{ij}$ zu den Generatoren \mathbf{x}_i und \mathbf{x}_j laut Definition des Voronoi-Diagramms gleich sind. Für alle inneren Volumenelemente stimmt damit die Behauptung. Eine Region Ω_i am Gebietsrand $\partial\Omega$ beinhaltet Randflächen $\partial\Omega_{il} \subset \partial\Omega$ mit $l \in \mathcal{B}_i$, wobei \mathcal{B}_i die Menge der Indizes der Randflächen bezeichnet. In diesem Fall erhält man aus (3.28) und (3.29):

$$\frac{\partial}{\partial x_i^k} \mathcal{F}(\mathcal{X}) = -2 \int_{\Omega_i} \rho(\mathbf{x})(x^k - x_i^k) d\mathbf{x} + \sum_{l \in \mathcal{B}_i} \int_{\partial\Omega_{il}} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^2 \frac{\partial \mathbf{x}}{\partial x_i^k} \cdot \mathbf{n} d\mathbf{x}. \quad (3.38)$$

Der Ausdruck $\frac{\partial \mathbf{x}}{\partial x_i^k} \cdot \mathbf{n}$ muss Null sein, da die Position des Randes $\partial\Omega$ von den Generatoren unabhängig ist. Somit gilt die Behauptung auch für alle Elemente am Rand. \square

3.2.2 Normierte Trägheit und Ideales Polyeder

In diesem Unterabschnitt gewinnen wir die Einsicht in die Tatsache, warum die Gitter mit regelmäßigen Tetraedern oder Hexaedern im Sinne CVT instabil sind. Instabil bedeutet hierbei die Eigenschaft dieser Gitter schon bei kleinsten Störungen der Generatoren und bei weiterer Minimierung der Funktion in (3.4) mithilfe der in den Abschnitten 3.3, 3.4 beschriebenen Algorithmen in Gittern aus allgemeinen CVT-Elementen zu resultieren. Die Ursache dafür ist, dass die Gitter mit regelmäßigen Tetraedern oder Hexaedern eine Art lokales Minimum der Funktion in (3.4) bilden.

Der Begriff normierte Trägheit (auf englisch: normalized inertia) wurde von Gersho in [8] eingeführt. Für ein Polyeder P wird die normierte Trägheit wie folgt definiert:

$$l(P) = \frac{\int_P \|\mathbf{x} - \hat{\mathbf{x}}\|^2 d\mathbf{x}}{(|P|)^{5/3}}, \quad (3.39)$$

wobei $\hat{\mathbf{x}}$ der geometrische Schwerpunkt von P und $|P|$ das Volumen von P sind. Die Normierung hat die Eigenschaft, dass $l(\gamma P) = l(P)$ für $\gamma > 0$, wobei $\gamma P = \{\gamma \mathbf{x} | \mathbf{x} \in P\}$. Mit anderen Worten, wenn P mit γ skaliert wird, bleibt die normierte Trägheit unverändert. Wir berechnen diese Größe für ein regelmäßiges (gleichseitiges) Tetraeder, ein regelmäßiges Hexaeder, ein abgeschnittenes Oktaeder (Abbildung 3.2) und eine Kugel. Da die Skalierung die normierte Trägheit unverändert lässt, betrachten wir ein gleichseitiges Tetraeder, der durch folgende Punkte definiert ist:

$$\mathbf{p}_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \mathbf{p}_2 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \mathbf{p}_3 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \mathbf{p}_4 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

Das Integral $\int_{tet} \|\mathbf{x} - \hat{\mathbf{x}}\|^2 d\mathbf{x}$ können wir mit Hilfe des Transformationssatzes berechnen:

Satz 3.4 (Transformationssatz (Quelle [41])). Es sei $\Omega \subset \mathbb{R}^d$ eine offene Menge und $\Phi : \Omega \mapsto \Phi(\Omega) \subset \mathbb{R}^d$ ein Diffeomorphismus. Dann ist die Funktion f auf $\Phi(\Omega)$ genau dann integrierbar, wenn die Funktion $x \mapsto f(\Phi(x)) \left| \det \left(\frac{\partial \Phi(y)}{\partial y} \right) \right|$ auf Ω integrierbar ist. In diesem Fall gilt:

$$\int_{\Phi(\Omega)} f(\mathbf{x}) d\mathbf{x} = \int_{\Omega} f(\Phi(\mathbf{y})) \left| \det \left(\frac{\partial \Phi(\mathbf{y})}{\partial \mathbf{y}} \right) \right| d\mathbf{y}, \quad (3.40)$$

wobei $\frac{\partial \Phi(\mathbf{y})}{\partial \mathbf{y}}$ die Jacobi-Matrix und $\det \left(\frac{\partial \Phi(\mathbf{y})}{\partial \mathbf{y}} \right)$ die Funktionaldeterminante von Φ ist.

Die Integrationsgrenzen für ein Referenztetraeder mit Eckpunkten

$$\tilde{\mathbf{p}}_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \tilde{\mathbf{p}}_2 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \tilde{\mathbf{p}}_3 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \tilde{\mathbf{p}}_4 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

lassen sich leicht bestimmen:

$$0 \leq y^3 \leq 1 - y^1 - y^2, \quad 0 \leq y^2 \leq 1 - y^1, \quad 0 \leq y^1 \leq 1. \quad (3.41)$$

Die Abbildung $\Phi(\mathbf{y})$ vom Referenztetraeder zum regelmäßigen Tetraeder ist wie folgt gegeben:

$$\Phi(\mathbf{y}) = A\mathbf{y} + \mathbf{p}_1, \text{ mit } A = (\mathbf{p}_2 - \mathbf{p}_1 \quad \mathbf{p}_3 - \mathbf{p}_1 \quad \mathbf{p}_4 - \mathbf{p}_1). \quad (3.42)$$

Für das regelmäßige Tetraeder erhält man damit:

$$\int_{tet} \|\mathbf{x} - \hat{\mathbf{x}}\|^2 d\mathbf{x} = \int_0^1 \int_0^{1-y^1} \int_0^{1-y^1-y^2} \|\mathbf{A}\mathbf{y} + \mathbf{p}_1 - \hat{\mathbf{x}}\|^2 |\det(A)| dy^3 dy^2 dy^1 = \frac{1}{20}. \quad (3.43)$$

Das Volumen des durch die Punkte $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4$ Tetraeders lässt sich durch die Anwendung des Transformationssatzes analog berechnen, und ist gleich $\frac{1}{3}$. Für die normierte Trägheit erhält man damit:

$$l_{tet} = \frac{1/20}{(1/3)^{5/3}} \approx 0.3120. \quad (3.44)$$

Gersho [8] hat die normierte Trägheit für ein Hexaeder, ein abgeschnittenes Oktaeder (Abbildung 3.2) und eine Kugel angegeben:

$$l_{hex} = 0.25 \quad (3.45)$$

$$l_{cutoct} = \frac{19}{64\sqrt[3]{2}} \approx 0.235629 \quad (3.46)$$

$$l_{sph} = \frac{\int_{sph} \|\mathbf{x} - \hat{\mathbf{x}}\|^2 d\mathbf{x}}{(\Omega_{sph})^{5/3}} = \frac{\frac{4}{5}\pi R^5}{\left(\frac{4}{3}\pi R^3\right)^{5/3}} \approx 0.230901. \quad (3.47)$$

Gersho behauptet auch, dass das abgeschnittene Oktaeder aus allen möglichen Polyedern in \mathbb{R}^3 die kleinste normierte Trägheit besitzt. Dies wurde in dieser Arbeit validiert. Die normierte Trägheit wurde für ein homogenes CV-Gitter mit 637 Elementen berechnet. Es wurden minimaler, maximaler und durchschnittlicher Werte berechnet:

$$l_{cvt}^{min} \approx 0.235780, \quad l_{cvt}^{max} \approx 0.242182, \quad \bar{l}_{cvt} \approx 0.238446. \quad (3.48)$$

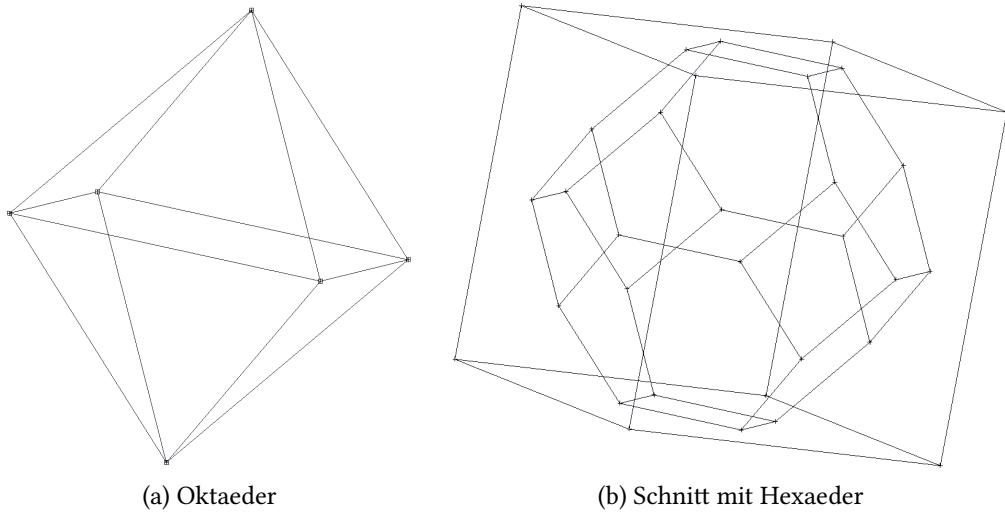


Abbildung 3.2: Abgeschnittenes Oktaeder. Die Oberflächen sind entweder Quadrate oder regelmäßige Sechsecke.

Wie wir anhand (3.48) und (3.46) feststellen können, ist der kleinste Wert der normierten Trägheit bei dem untersuchten CV-Gitter etwas größer als die normierte Trägheit des abgeschnittenen Oktaeders. Hier sehen wir eine praktische Bestätigung der Gersho's Hypothese über die Eigenschaft des abgeschnittenen Oktaeders, bei einer Tessellierung eines Gebietes das optimale Polyeder unter allen Centroidal-Voronoi-Elementen im Bezug auf die normierte Trägheit zu sein. Als einzelne geometrische Figur besitzt jedoch die Kugel die kleinste normierte Trägheit.

Bemerkung 3.1:

Bei einer CVT mit einer homogenen Zelldichte ρ strebt jedes Volumenelement so viel wie möglich Nachbarn an zu haben, um damit einer „Kugel“ zu ähneln. Damit wird ein Volumenelement gemeint, das unendlich viele Flächenelemente hat, und die Abstände von dem geometrischen Schwerpunkt zu den Voronoi-Knoten alle gleich sind. Es ist jedoch nicht möglich ein Gebiet mit solchen komplett „kugelförmigen“ Volumenelementen zu tessellieren.

Die Gersho's Hypothese ([8]) lautet: Wenn die Anzahl der Generatoren N groß genug ist, erreicht die Funktion $\sum_i \int_{\Omega_i} \|x - x_i\|^2 dx$ ihr Minimum genau dann, wenn alle Polyeder Ω_i gleiche Form annehmen. Die Form des abgeschnittenen Oktaeders ist dabei optimal. Die Randelemente bilden hierbei eine Ausnahme, da ihre Form vom Rand abhängig ist.

Bemerkung 3.2:

Die Bemerkung 3.1 liefert die Erklärung, warum die Gitter mit regelmäßigen Tetraedern oder Hexaedern im Sinne CVT instabil sind. Die CVT entstehen durch die Minimierung der Funktion $\sum_i \int_{\Omega_i} \|x - x_i\|^2 dx$, und diese nimmt ihr globales Minimum genau dann an, wenn alle Voronoi-Regionen dem abgeschnittenen Oktaeder möglichst ähnlich sind. Bei den Gittern mit regelmäßigen Tetraedern bzw. Hexaedern kann es sich nur um ein lokales Minimum der Funktion $\sum_i \int_{\Omega_i} \|x - x_i\|^2 dx$ handeln, denn die Form des abgeschnittenen Oktaeders ist optimal.

3.2.3 Definition der Zelldichte

Die Definition der Zelldichte basiert auf der Gersho's Hypothese ([8]). Wie in der Bemerkung 3.1 angegeben ist, führt diese Hypothese zu der sogenannten Gleichverteilung des Fehlers. Bei

einer homogenen Zelldichte erhält man damit:

$$\int_{\Omega_i} \|\mathbf{x} - \mathbf{x}_i\|^2 d\mathbf{x} \approx \beta, \text{ mit } \beta \in \mathbb{R} \text{ und für } i = 1, 2, \dots, N.$$

Das heißt, dass jedes Element den gleichen Beitrag zum globalen Fehler

$$\sum_{i=1}^N \int_{\Omega_i} \|\mathbf{x} - \mathbf{x}_i\|^2 d\mathbf{x}$$

hat. Wir suchen so eine Zelldichte $\rho(\mathbf{x})$, dass

$$\int_{\Omega_i} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^2 d\mathbf{x} \stackrel{!}{=} \beta, \quad \forall i \in \{1, 2, \dots, N\}. \quad (3.49)$$

Bei inhomogener Zelldichte würde somit jedes Volumenelement ebenfalls gleich zum globalen Fehler

$$\sum_{i=1}^N \int_{\Omega_i} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^2 d\mathbf{x} \quad (3.50)$$

beitragen. In [59] wurde die Zelldichte als $|\Omega|^{-5/3}$ definiert. Dies wollen wir hier etwas ausführlicher begründen. Seien r_i^{min} der minimale und r_i^{max} der maximale Radius des Elements Ω_i . D.h. es gilt:

$$r_i^{min} := \min_{\mathbf{x} \in \partial\Omega_i} \|\mathbf{x} - \mathbf{x}_i\|, \quad \forall i \in \{1, 2, \dots, N\} \quad (3.51)$$

$$r_i^{max} := \max_{\mathbf{x} \in \partial\Omega_i} \|\mathbf{x} - \mathbf{x}_i\|, \quad \forall i \in \{1, 2, \dots, N\}. \quad (3.52)$$

Seien weiterhin ρ_i^{min} das Minimum und ρ_i^{max} das Maximum von $\rho(\mathbf{x})$ in Ω_i . Damit soll gelten:

$$\rho_i^{min} := \min_{\mathbf{x} \in \Omega_i} \rho(\mathbf{x}), \quad \forall i \in \{1, 2, \dots, N\}, \quad (3.53)$$

$$\rho_i^{max} := \max_{\mathbf{x} \in \Omega_i} \rho(\mathbf{x}), \quad \forall i \in \{1, 2, \dots, N\}. \quad (3.54)$$

Mit Benutzung von (3.47), (3.51)-(3.54) erhalten wir aus (3.49):

$$\int_{\Omega_i} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^2 d\mathbf{x} \geq \int_{\Omega_i} \rho_i^{min} \|\mathbf{x} - \mathbf{x}_i\|^2 d\mathbf{x} \geq \rho_i^{min} \frac{4}{5} \pi (r_i^{min})^5, \quad (3.55)$$

$$\int_{\Omega_i} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^2 d\mathbf{x} \leq \int_{\Omega_i} \rho_i^{max} \|\mathbf{x} - \mathbf{x}_i\|^2 d\mathbf{x} \leq \rho_i^{max} \frac{4}{5} \pi (r_i^{max})^5. \quad (3.56)$$

Wir suchen so eine Zelldichte-Funktion, damit das Integral in (3.49) für jedes Element Ω_i durch ein Intervall beschränkt ist. D.h. $\forall i \in \{1, 2, \dots, N\}$ muss folgendes gelten:

$$\beta_{min} \leq \int_{\Omega_i} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^2 d\mathbf{x} \leq \beta_{max}, \quad (3.57)$$

mit $\beta_{min} \in \mathbb{R}$ und $\beta_{max} \in \mathbb{R}$. Mit Berücksichtigung von (3.55), (3.56) und (3.57) können wir β_{min} und β_{max} so definieren, dass für jedes $i \in \{1, 2, \dots, N\}$ folgende Identitäten gelten.

$$\beta_{min} := \rho_i^{min} \frac{4}{5} \pi (r_i^{min})^5 \quad \Rightarrow \quad \rho_i^{min} = \frac{\beta_{min}}{\frac{4}{5} \pi (r_i^{min})^5}, \quad (3.58)$$

$$\beta_{max} := \rho_i^{max} \frac{4}{5} \pi (r_i^{max})^5 \quad \Rightarrow \quad \rho_i^{max} = \frac{\beta_{max}}{\frac{4}{5} \pi (r_i^{max})^5}. \quad (3.59)$$

Damit haben wir für die Zelldichte $\rho_i(\mathbf{x})$ für $i \in \{1, 2, \dots, N\}$ eine untere und eine obere Schranke gefunden:

$$\frac{\beta_{min}}{\frac{4}{5} \pi (r_i^{min})^5} \leq \rho_i(\mathbf{x}) \leq \frac{\beta_{max}}{\frac{4}{5} \pi (r_i^{max})^5}. \quad (3.60)$$

Damit können wir für $i = 1, 2, \dots, N$ die Zelldichte wie folgt definieren:

$$\rho_i(\mathbf{x}) := \frac{\beta_i(\mathbf{x})}{\frac{4}{5} \pi (r_i(\mathbf{x}))^5}, \quad (3.61)$$

$$\beta_i(\mathbf{x}) : \mathbb{R} \mapsto [\beta_{min}, \beta_{max}], \quad \mathbf{x} \mapsto \beta_i(\mathbf{x}), \quad (3.62)$$

$$r_i(\mathbf{x}) : \mathbb{R} \mapsto [r_i^{min}, r_i^{max}], \quad \mathbf{x} \mapsto r_i(\mathbf{x}). \quad (3.63)$$

Aus (3.61)-(3.63) ist es ersichtlich, dass $\beta_i(\mathbf{x})$ durch die Terme β_{min} und β_{max} beschränkt ist, die für alle Elemente gleich sind. Damit $\rho(\mathbf{x})$ für verschiedene Zellen auch verschiedene Werte annehmen kann, muss der entscheidende Teil dieser Funktion in $(r_i(\mathbf{x}))^{-5}$ enthalten sein. $\beta_i(\mathbf{x})$ stellt nur einen geringen Teil der Streuung der Funktion $\rho_i(\mathbf{x})$ im i -ten Element dar. Nach der Gersho's Hypothese streben alle Elemente möglichst gleiche Form anzunehmen, wenn deren Anzahl groß wird. D.h. wir können schreiben:

$$\beta_{min} \approx \beta \approx \beta_{max}. \quad (3.64)$$

Damit folgt aus (3.61) und (3.64):

$$\rho_i(\mathbf{x}) \approx \frac{\beta}{\frac{4}{5} \pi (r_i(\mathbf{x}))^5}. \quad (3.65)$$

Wenn wir (3.65) in (3.50) einsetzen, erhalten wir:

$$\sum_{i=1}^N \int_{\Omega_i} \frac{\beta}{\frac{4}{5} \pi (r_i(\mathbf{x}))^5} \|\mathbf{x} - \mathbf{x}_i\|^2 d\mathbf{x} = \frac{\beta}{\frac{4}{5} \pi} \sum_{i=1}^N \int_{\Omega_i} \frac{1}{(r_i(\mathbf{x}))^5} \|\mathbf{x} - \mathbf{x}_i\|^2 d\mathbf{x}. \quad (3.66)$$

Da der Skalar $\frac{\beta}{\frac{4}{5} \pi}$ die Minimalstelle der Funktion in (3.66) nicht verändert, kann man schreiben:

$$\rho_i(\mathbf{x}) \approx \frac{1}{(r_i(\mathbf{x}))^5}, \quad \text{mit } r_i(\mathbf{x}) \in [r_i^{min}, r_i^{max}] \text{ für } i = 1, 2, \dots, N. \quad (3.67)$$

Aus der Gleichung (3.67) folgt, dass die Streuung von $\rho_i(\mathbf{x})$ in jedem Element Ω_i nur durch den Unterschied zwischen dem kleinsten r_i^{min} und dem größten r_i^{max} Radien dieses Elements bestimmt ist. Damit haben wir einen Zusammenhang (3.67) zwischen der Zelldichte $\rho(\mathbf{x})$ und dem Längenmaß $r(\mathbf{x})$ hergestellt, der darauf basiert, dass die Approximation in (3.49) möglichst gut ist.

Angenommen, wir würden die Zelldichte als die Anzahl der Elemente pro Volumeneinheit definieren, dann würde folgendes gelten:

$$\int_{\Omega_i} \frac{1}{r_i^3} \|\mathbf{x} - \mathbf{x}_i\|^2 d\mathbf{x} \approx \frac{r_i^5}{r_i^3} = r_i^2. \quad (3.68)$$

Das heißt, dass die Elemente mit höherem Volumeninhalt wesentlich mehr zum globalen Fehler in (3.50) beitragen würden. Als Folge könnte eine starke Verzerrung der kleineren Volumenelemente zugunsten der größeren Volumenelemente eine Reduktion des Fehlers in (3.50) bewirken. Im optimalen Fall ist es hingegen erwünscht, dass jedes Volumenelement möglichst kleine normierte Trägheit besitzt (siehe Unterabschnitt 3.2.2).

Bemerkung 3.3:

Wie wir in diesem Unterabschnitt festgestellt haben, hat eine Multiplikation der Zelldichte $\rho(\mathbf{x})$ mit einem positiven Skalar keinen Einfluss auf die Minimalstelle der Funktion in (3.50). Das Gleiche gilt auch für die Kantenlänge $r(\mathbf{x})$. D.h. entscheidend ist nur der relative Unterschied der Kantenlänge im gesamten Modell.

3.3 Lloyd-Algorithmus

In diesem Abschnitt beschäftigen wir uns mit dem sogenannten Lloyd-Algorithmus zur Konstruktion von CVT. In [64] ist eine sehr ausführliche Übersicht über weitere Techniken zur Konstruktion von CVT gegeben. Der Lloyd-Algorithmus ist eine iterative Vorschrift, bei der pro Schritt die Schwerpunkte aktueller Volumenelemente den neuen Generatoren zugewiesen werden. Danach erfolgt neue Voronoi-Tessellierung. Diese Schritte sind im Algorithmus 3.1 zusammengefasst. Im weiteren wollen wir zeigen, dass der Lloyd-Algorithmus als eine Fixpunkt-

Algorithmus 3.1 Lloyd-Algorithmus

Seien $\Omega \subset \mathbb{R}^3$ abgeschlossen und $\mathcal{M} := \{(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)^T \mid \mathbf{x}_i \in \Omega, \text{ für } i = 1, 2, \dots, N\}$. Sei weiterhin $\rho(\mathbf{x}) : \mathcal{M} \mapsto \mathbb{R}$ die Zelldichte. Wähle einen Startvektor $\mathcal{X}^0 \in \mathcal{M}$. Setze $n = 0$.

do

1. Konstruiere das Voronoi-Diagramm von Ω zum gegebenen \mathcal{X}^n .
2. Setze neue Generatoren zu den gewichteten Schwerpunkten, d.h:

$$\mathbf{x}_i^{n+1} = \left(\int_{\Omega_i(\mathcal{X}^n)} \rho(\mathbf{x}) x dx \right) \left/ \int_{\Omega_i(\mathcal{X}^n)} \rho(\mathbf{x}) d\mathbf{x} \right. \quad (3.69)$$

3. $n = n + 1$.

while $\|\mathcal{X}^{n+1} - \mathcal{X}^n\| > \text{Toleranz}$

iteration formuliert werden kann. Anhand der Definition 3.2 kann man feststellen, dass für einen festen Rand $\partial\Omega$ jedes Ω_i von \mathbf{x}_i und seinen Nachbarn abhängt. Das heißt Ω_i hängt von $\mathcal{X} \in \mathcal{M}$ ab, wobei

$$\mathcal{M} := \{(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)^T \mid \mathbf{x}_i \in \Omega, \text{ für } i = 1, 2, \dots, N\}. \quad (3.70)$$

Für jeden Schritt des Algorithmus 3.1 können wir eine Abbildung definieren:

$$T : \mathcal{M} \mapsto \mathcal{M}, \quad \mathcal{X}^n \mapsto \mathcal{X}^{n+1}, \text{ so dass } T_i(\mathcal{X}^n) := \mathbf{x}_i^{n+1} = \frac{\int_{\Omega_i(\mathcal{X}^n)} \rho(\mathbf{x}) x d\mathbf{x}}{\int_{\Omega_i(\mathcal{X}^n)} \rho(\mathbf{x}) d\mathbf{x}}. \quad (3.71)$$

Wenn \mathcal{X}^n ein CV-Diagramm erzeugt, muss zwingend

$$\mathcal{X}^n = T(\mathcal{X}^n) \text{ oder } \mathcal{X}^* = T(\mathcal{X}^*) \quad (3.72)$$

gelten. Durch die Gleichung (3.72) ist es ersichtlich, dass der Lloyd-Algorithmus eine Fixpunktiteration darstellt, wobei der Fixpunkt genau dann vorliegt, wenn die Generatoren der Voronoi-Zellen mit ihren Schwerpunkten übereinstimmen. Der Banachsche Fixpunktsatz findet hierbei bedauerlicherweise keine Anwendung, da der Lloyd-Algorithmus schon für zweidimensionale Gitter nicht kontraktiv ist.

Definition 3.4 (Kontraktion). *Sei (M, d) ein metrischer Raum. Eine Abbildung $\phi : M \mapsto M$ heißt eine Kontraktion, wenn es eine Zahl $\lambda \in [0, 1)$ gibt, mit der für alle $\mathbf{x}, \mathbf{y} \in M$ gilt:*

$$d(\phi(\mathbf{x}), \phi(\mathbf{y})) \leq \lambda d(\mathbf{x}, \mathbf{y}). \quad (3.73)$$

Die Größe λ heißt die Kontraktionszahl. (Quelle [84].)

Die Ungleichung in (3.73) gilt für den Lloyd-Algorithmus nicht, weil der Fixpunkt vom Startwert abhängig ist. Es stellt sich hierbei die Frage, welche Konvergenzordnung der Algorithmus 3.1 besitzt.

Definition 3.5 (Konvergenzordnung). *Unter der Konvergenzordnung versteht man ein $p \in \mathbb{N}$, so dass für eine konvergente Folge \mathbf{x}^n ein $c > 0$ existiert und es gilt:*

$$\|\mathbf{x}^{n+1} - \mathbf{x}^*\| \leq c \|\mathbf{x}^n - \mathbf{x}^*\|^p \text{ für } n = 0, 1, \dots \quad (3.74)$$

wobei \mathbf{x}^* der Grenzwert der Folge (\mathbf{x}^n) ist. Für weitere Details siehe [84].

Im Unterabschnitt 3.6.1 wird gezeigt, dass der Lloyd-Algorithmus nicht höher als die Konvergenz der ersten Ordnung besitzt.

Definition 3.6 (Konvergenzrate). *Sei \mathbf{x}^* der Fixpunkt einer Fixpunktiteration. Die Konvergenzrate ist in dieser Arbeit wie folgt zu verstehen:*

$$\sigma^n = -\ln \left(\frac{\|\mathbf{x}^n - \mathbf{x}^*\|}{\|\mathbf{x}^{n-1} - \mathbf{x}^*\|} \right). \quad (3.75)$$

Je größer σ^n , desto weniger Iterationen werden benötigt, um den Fixpunkt zu erreichen.

Gemäß der Aussagen in [61, 62] sinkt die Konvergenzrate des Lloyd-Algorithmus mit steigender Anzahl der Generatoren. Dies wird weitgehend im Unterabschnitt 3.6.1 untersucht.

3.4 Alternativer Lloyd-Newton-Algorithmus

In diesem Abschnitt stellen wir einen Algorithmus zur Berechnung der CVT vor, der eine deutliche Verbesserung gegenüber dem Lloyd-Algorithmus im Bezug auf die Konvergenzrate besitzt. In [61] wurde eine Lloyd-Newton-Methode angeboten, die von dem sogenannten Newton-Verfahren abgeleitet wurde. Das Newton-Verfahren wird dafür verwendet, um die Nullstellen einer im Allgemeinen nichtlinearen Funktion zu bestimmen. In diesem Abschnitt wird eine alternative Abwandlung vom Newton-Verfahren vorgestellt, die wesentlich effizienter als die in [61] angebotene Methode ist. Zuerst betrachten wir das reine Newton-Verfahren. Dazu führen wir eine Funktion ein, deren Nullstelle zu bestimmen ist.

$$\mathcal{G}(\mathcal{X}) := \frac{1}{2} \nabla \mathcal{F}(\mathcal{X}), \quad (3.76)$$

wobei $\mathcal{F}(\mathcal{X})$ wie in (3.4) definiert ist. Das heißt, es muss

$$\mathcal{G}_i(\mathcal{X}) = \int_{\Omega_i} \rho(\mathbf{x})(\mathbf{x}_i - \mathbf{x}) d\mathbf{x} \quad \text{für } i = 1, 2, \dots, N \quad (3.77)$$

gelten. Wenn die Funktion in (3.76) bzw. (3.77) gleich Null an der Stelle \mathcal{X}^* ist, erzeugt \mathcal{X}^* ein Centroidal-Voronoi-Diagramm. Der Algorithmus 3.2 stellt die Anwendung des Newton-Verfahrens auf die Gleichung in (3.77) dar. Der Satz 3.5 zeigt, welche Gestalt die einzelnen Elemente der Jacobi-Matrix $\frac{\partial}{\partial \mathcal{X}} \mathcal{G}(\mathcal{X})$ haben.

Algorithmus 3.2 Newton Verfahren zur Berechnung der CVT

Seien $\Omega \subset \mathbb{R}^3$ abgeschlossen und $\mathcal{M} := \{(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)^T \mid \mathbf{x}_i \in \Omega, \text{ für } i = 1, 2, \dots, N\}$. Sei weiterhin $\rho(\mathbf{x}) : \mathcal{M} \mapsto \mathbb{R}$ die Zelldichte. Zu bestimmen sei eine Nullstelle der Funktion $\mathcal{G}(\mathcal{X})$, wie diese in (3.77) definiert ist. Wähle einen Startvektor $\mathcal{X}^0 \in \mathcal{M}$ und setze $n = 0$.

do

1. Konstruiere das Voronoi-Diagramm von Ω zum gegebenen \mathcal{X}^n .
2. Werte die Jacobimatrix $\frac{\partial \mathcal{G}}{\partial \mathcal{X}}(\mathcal{X}^n)$ und die Funktion $\mathcal{G}(\mathcal{X}^n)$ aus.
3. Löse das Gleichungssystem $\frac{\partial \mathcal{G}}{\partial \mathcal{X}}(\mathcal{X}^n) \Delta \mathcal{X}^n = -\mathcal{G}(\mathcal{X}^n)$.
4. Bestimme $\mathcal{X}^{n+1} = \mathcal{X}^n + \Delta \mathcal{X}^n$.
5. $n = n + 1$.

while $\|\Delta \mathcal{X}^n\| > tol$

Satz 3.5. Sei $\mathcal{G}(\mathcal{X})$ wie in (3.76)-(3.77) definiert. Die einzelnen Blockmatrizen der Jacobi-Matrix $\frac{\partial}{\partial \mathcal{X}} \mathcal{G}(\mathcal{X})$ haben folgende Form:

$$\frac{\partial}{\partial \mathbf{x}_i} \mathcal{G}_i(\mathcal{X}) = - \sum_{j \in \mathcal{N}_i} \int_{\partial \Omega_{ij}} \rho(\mathbf{x}) \frac{(\mathbf{x} - \mathbf{x}_i)(\mathbf{x} - \mathbf{x}_i)^T}{\|\mathbf{x}_j - \mathbf{x}_i\|} d\mathbf{x} + \mathbf{1} \int_{\Omega_i} \rho(\mathbf{x}) d\mathbf{x} \quad (3.78)$$

$$\frac{\partial}{\partial \mathbf{x}_j} \mathcal{G}_i(\mathcal{X}) = \int_{\partial \Omega_{ij}} \rho(\mathbf{x}) \frac{(\mathbf{x}_i - \mathbf{x})(\mathbf{x}_j - \mathbf{x})^T}{\|\mathbf{x}_j - \mathbf{x}_i\|} d\mathbf{x} = \left(\frac{\partial}{\partial \mathbf{x}_i} \mathcal{G}_j(\mathcal{X}) \right)^T, \quad (3.79)$$

wobei $\mathbf{1} \in \mathbb{R}^{3 \times 3}$ die Einheitsmatrix ist. $\partial \Omega_{ij}$ bezeichnet die gemeinsame Fläche zwischen den i -ten und j -ten Volumenelementen. Für ein Volumenelement am Gebietsrand gilt

$$\partial \Omega_i = \{\cup_{j \in \mathcal{N}_i} \partial \Omega_{ij}\} \bigcup \{\cup_{l \in \mathcal{B}_i} \partial \Omega_{il}\}, \quad (3.80)$$

wobei $\partial\Omega_{il} \subset \partial\Omega$ und \mathcal{B}_i die Indexmenge der Randflächen des t -ten Elements bezeichnet. Die Gleichungen (3.78)-(3.79) stimmen auch für alle Randelemente, da die Flächenintegrale über $\partial\Omega_{il}$ verschwinden. Weiterhin ist $\frac{\partial}{\partial \mathbf{x}} \mathcal{G}(\mathbf{x})$ symmetrisch.

Beweis. Gelten die Voraussetzungen wie in Satz 3.5. Mit Anwendung des Satzes 3.1 erhalten wir aus (3.77):

$$\frac{\partial}{\partial x_i^l} \mathcal{G}_i^k(\mathbf{x}) = \frac{\partial}{\partial x_i^l} \int_{\Omega_i} \rho(\mathbf{x})(x_i^k - x^k) d\mathbf{x} = \int_{\Omega_i} \frac{\partial}{\partial x_i^l} (\rho(\mathbf{x})(x_i^k - x^k)) d\mathbf{x} \quad (3.81)$$

$$= \int_{\Omega_i} \left[\underbrace{\frac{\partial (\rho(\mathbf{x})(x_i^k - x^k))}{\partial \mathbf{x}}}_{(a)} \frac{\partial \mathbf{x}}{\partial x_i^l} + \underbrace{\frac{\partial (\rho(\mathbf{x})(x_i^k - x^k))}{\partial \mathbf{x}_i^k}}_{(b)} \frac{\partial \mathbf{x}_i^k}{\partial x_i^l} \right] d\mathbf{x} \quad (3.82)$$

Mit dem Lemma 3.1 und dem Gaußschen Satz erhalten wir für (a):

$$\int_{\Omega_i} (a) d\mathbf{x} = \int_{\Omega_i} \nabla \cdot \left(\rho(\mathbf{x})(x_i^k - x^k) \frac{\partial \mathbf{x}}{\partial x_i^l} \right) d\mathbf{x} = \int_{\partial\Omega_i} \rho(\mathbf{x})(x_i^k - x^k) \frac{\partial \mathbf{x}}{\partial x_i^l} \cdot \mathbf{n} d\mathbf{x} \quad (3.83)$$

Durch Aufteilen des Oberflächenintegrals $\int_{\partial\Omega_i} d\mathbf{x}$ in Integrale über einzelne Flächenelemente $\partial\Omega_{ij}$ erhält man aus (3.83)

$$\int_{\Omega_i} (a) d\mathbf{x} = \sum_{j \in \mathcal{N}_i \setminus \partial\Omega_{ij}} \int_{\partial\Omega_{ij}} \rho(\mathbf{x})(x_i^k - x^k) \frac{\partial \mathbf{x}}{\partial x_i^l} \cdot \mathbf{n} d\mathbf{x}, \quad (3.84)$$

wobei \mathcal{N}_i die Indexmenge der Nachbarn des i -ten Volumenelements ist. Lemma 3.2 liefert für (3.84)

$$\int_{\Omega_i} (a) d\mathbf{x} = \sum_{j \in \mathcal{N}_i \setminus \partial\Omega_{ij}} \int_{\partial\Omega_{ij}} \rho(\mathbf{x})(x_i^k - x^k) \frac{x^l - x_i^l}{\|\mathbf{x}_j - \mathbf{x}_i\|} d\mathbf{x}. \quad (3.85)$$

Aus (3.82) erhalten wir für (b):

$$(b) = \rho(\mathbf{x}) \frac{\partial \mathbf{x}_i^k}{\partial x_i^l} = \rho(\mathbf{x}) e^k \cdot e^l, \quad (3.86)$$

Einsetzen von (3.85) und (3.86) in (3.82) liefert

$$\frac{\partial}{\partial x_i^l} \mathcal{G}_i^k(\mathbf{x}) = \sum_{j \in \mathcal{N}_i \setminus \partial\Omega_{ij}} \int_{\partial\Omega_{ij}} \rho(\mathbf{x})(x_i^k - x^k) \frac{x^l - x_i^l}{\|\mathbf{x}_j - \mathbf{x}_i\|} d\mathbf{x} + e^k \cdot e^l \int_{\Omega_i} \rho(\mathbf{x}) d\mathbf{x}. \quad (3.87)$$

Die Matrix-Vektor-Form der Gleichung (3.87) ist

$$\frac{\partial}{\partial \mathbf{x}_i} \mathcal{G}_i(\mathbf{x}) = - \sum_{j \in \mathcal{N}_i \setminus \partial\Omega_{ij}} \int_{\partial\Omega_{ij}} \rho(\mathbf{x}) \frac{(\mathbf{x} - \mathbf{x}_i)(\mathbf{x} - \mathbf{x}_i)^T}{\|\mathbf{x}_j - \mathbf{x}_i\|} d\mathbf{x} + \mathbb{1} \int_{\Omega_i} \rho(\mathbf{x}) d\mathbf{x}, \quad (3.88)$$

wobei $\mathbb{1} \in \mathbb{R}^{3 \times 3}$ die Einheitsmatrix ist. Mit der Tatsache, dass $\frac{\partial x_i^k}{\partial x_j^l} = 0$ und $\frac{\partial \mathbf{x}}{\partial x_j^l} \cdot \mathbf{n}_u = 0$ für $\mathbf{x} \in \partial\Omega_{iu}$ mit $u \in \mathcal{N}_i \wedge u \neq j$, erhält man analog

$$\frac{\partial}{\partial x_j^l} \mathcal{G}_i^k(\mathbf{x}) = \int_{\partial\Omega_{ij}} \rho(\mathbf{x})(x_i^k - x^k) \frac{x_j^l - x_i^l}{\|\mathbf{x}_j - \mathbf{x}_i\|} d\mathbf{x}. \quad (3.89)$$

Es ist offensichtlich, dass die Blockmatrix $\frac{\partial}{\partial \mathbf{x}_i} \mathcal{G}_i(\mathcal{X})$ aus (3.88) symmetrisch ist. Aus Gleichung (3.89) folgt außerdem, dass

$$\frac{\partial}{\partial x_j^l} \mathcal{G}_i^k(\mathcal{X}) = \int_{\Omega_{ij}} \rho(\mathbf{x})(x_j^l - x_i^l) \frac{x_i^k - x^k}{\|\mathbf{x}_j - \mathbf{x}_i\|} d\mathbf{x} = \frac{\partial}{\partial x_i^k} \mathcal{G}_j^l(\mathcal{X}). \quad (3.90)$$

Das heißt, es gilt $\frac{\partial \mathcal{G}}{\partial \mathcal{X}} = (\frac{\partial \mathcal{G}}{\partial \mathcal{X}})^T$. □

Die Matrix $\frac{\partial \mathcal{G}}{\partial \mathcal{X}}$ muss laut Definition symmetrisch sein, da diese wiederum die Hesse-Matrix der Funktion $\mathcal{F}(\mathcal{X})$ in (3.4) geteilt durch zwei ist. Damit muss die Matrix $\frac{\partial \mathcal{G}}{\partial \mathcal{X}}$ in der lokalen Umgebung des Minimums von $\mathcal{F}(\mathcal{X})$ positiv definit sein (siehe [7, 84]). In der Praxis hat sich diese Aussage bestätigt. Wenn die Matrix $\frac{\partial \mathcal{G}}{\partial \mathcal{X}}$ positiv definit ist, kann so ein effizientes Verfahren wie die Methode der Konjugierten-Gradienten zur Lösung des Gleichungssystems in Algorithmus 3.2 verwendet werden.

In der Praxis hat es sich herausgestellt, dass der Algorithmus 3.2 in seiner reinen Form nicht durchführbar ist. Dies hat die folgende Ursache. Da die Funktion $\mathcal{G}(\mathcal{X})$ im Allgemeinen nicht linear ist, besitzt diese mehrere Nullstellen. Dies kann man leicht an der Tatsache erkennen, dass eine Vertauschung von zwei Generatoren einer CVT wiederum eine CVT liefert. Außerdem ist beim Algorithmus 3.2 nicht ausgeschlossen, dass ein Generator außerhalb des Definitionsgebietes Ω geraten kann. Die in [61] beschriebene Lloyd-Newton-Methode löst dieses Problem durch Reduktion der Verschiebung $\Delta \mathcal{X}^n$. Für jeden Generator wird überprüft, ob dieser im Definitionsgebiet liegt. Die Verschiebung $\Delta \mathcal{X}^n$ wird soweit verkleinert, bis dieser Generator im Definitionsgebiet landet. Außerdem wurde in [61] das Newton-Verfahren auf die Gleichung

$$\int_{\Omega_i} \rho(\mathbf{x}) \mathbf{x} d\mathbf{x} \left/ \int_{\Omega_i} \rho(\mathbf{x}) d\mathbf{x} - \mathbf{x}_i \right. = 0 \quad (3.91)$$

angewandt. In diesem Fall ist die jeweilige Jacobi-Matrix der Funktion in (3.91) weder symmetrisch noch positiv definit.

In diesem Abschnitt wird eine alternative Strategie beschrieben, um die Generatoren bei jedem Schritt in Algorithmus 3.2 auf das Definitionsgebiet Ω zu beschränken. Dazu betrachten wir die einzelnen Blockmatrizen in (3.78)-(3.79) der Jacobi-Matrix $\frac{\partial \mathcal{G}}{\partial \mathcal{X}}$. Wir führen neue Variable $\mathcal{H} := \frac{\partial \mathcal{G}}{\partial \mathcal{X}}$ ein, so dass

$$\mathcal{H}_{ii}(\mathcal{X}) := \frac{\partial}{\partial \mathbf{x}_i} \mathcal{G}_i(\mathcal{X}) \quad (3.92)$$

$$\mathcal{H}_{ij}(\mathcal{X}) := \frac{\partial}{\partial \mathbf{x}_j} \mathcal{G}_i(\mathcal{X}) \quad (3.93)$$

gilt. Nun erzeugen wir aus \mathcal{H} eine Matrix $\tilde{\mathcal{H}}$, so dass

$$\tilde{\mathcal{H}}_{ii}(\mathcal{X}) := \mathbb{1} \int_{\Omega_i} \rho(\mathbf{x}) d\mathbf{x} \quad (3.94)$$

$$\tilde{\mathcal{H}}_{ij}(\mathcal{X}) = \left(\tilde{\mathcal{H}}_{ji}(\mathcal{X}) \right)^T := 0 \quad (3.95)$$

ist. Damit erhalten wir für $\tilde{\mathcal{H}}$ eine Diagonalmatrix, wobei auf der Diagonale die Massen der einzelnen Volumenelemente stehen. Die $(3i + k)$ -te Zeile des zu lösenden Gleichungssystems

$$\tilde{\mathcal{H}}(\mathcal{X}^n) \Delta \mathcal{X}^n = -\mathcal{G}(\mathcal{X}^n) \quad (3.96)$$

aus dem Algorithmus 3.2 nimmt damit die folgende Gestalt an:

$$\int_{\Omega_i} \rho(\mathbf{x}) d\mathbf{x} \Delta x_i^k = - \int_{\Omega_i} \rho(\mathbf{x})(x_i^k - x^k) d\mathbf{x} = -x_i^k \int_{\Omega_i} \rho(\mathbf{x}) d\mathbf{x} + \int_{\Omega_i} \rho(\mathbf{x}) x^k d\mathbf{x}, \quad (3.97)$$

wobei hier auf den Index der Iteration n verzichtet wurde. Nach einer Umformung von (3.97) erhalten wir

$$\Delta x_i^k + x_i^k = \frac{\int_{\Omega_i} \rho(\mathbf{x}) x^k d\mathbf{x}}{\int_{\Omega_i} \rho(\mathbf{x}) d\mathbf{x}}. \quad (3.98)$$

Wenn wir (3.98) in die Vorschrift des Algorithmus 3.2 einsetzen, erhalten wir

$$(x_i^k)^{n+1} = (x_i^k)^n + (\Delta x_i^k)^n = \frac{\int_{\Omega_i} \rho(\mathbf{x}) x^k d\mathbf{x}}{\int_{\Omega_i} \rho(\mathbf{x}) d\mathbf{x}}. \quad (3.99)$$

Das heißt, der Algorithmus 3.2 mit der modifizierten Jacobi-Matrix $\hat{\mathcal{H}}$ wie in (3.94)-(3.95) verwandelt sich in den Lloyd-Algorithmus. Es stellt sich die Frage, ob es möglich ist, sowohl die Konvergenzrate des Lloyd-Algorithmus zu erhöhen, als auch die Stabilität des Algorithmus 3.2 zu gewährleisten. Möglich wird dies dadurch, dass wir eine Konstante $\alpha \in (0, 1)$ und eine Matrix $\hat{\mathcal{H}}$ einführen, so dass folgendes gilt:

$$\hat{\mathcal{H}}_{ii}(\mathcal{X}) = \alpha \left(- \sum_{j \in \mathcal{N}_i} \int_{\partial\Omega_{ij}} \rho(\mathbf{x}) \frac{(\mathbf{x} - \mathbf{x}_i)(\mathbf{x} - \mathbf{x}_i)^T}{\|\mathbf{x}_j - \mathbf{x}_i\|} d\mathbf{x} \right) + \mathbb{1} \int_{\Omega_i} \rho(\mathbf{x}) d\mathbf{x} \quad (3.100)$$

$$\hat{\mathcal{H}}_{ij}(\mathcal{X}) = \alpha \left(\int_{\partial\Omega_{ij}} \rho(\mathbf{x}) \frac{(\mathbf{x}_i - \mathbf{x})(\mathbf{x}_j - \mathbf{x})^T}{\|\mathbf{x}_j - \mathbf{x}_i\|} d\mathbf{x} \right). \quad (3.101)$$

Durch die Verwendung von $\hat{\mathcal{H}}$ als Jacobi-Matrix entsteht ein quasi Newton-Verfahren, das durch den Algorithmus 3.3 definiert wird. Bei $\alpha = 0$ erhält man somit den Lloyd-Algorithmus, und

Algorithmus 3.3 Lloyd-Newton-Algorithmus zur Berechnung der CVT

Seien $\Omega \subset \mathbb{R}^3$ abgeschlossen und $\mathcal{M} := \{(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)^T \mid \mathbf{x}_i \in \Omega, \text{ für } i = 1, 2, \dots, N\}$. Sei weiterhin $\rho(\mathbf{x}) : \mathcal{M} \mapsto \mathbb{R}$ die Zelldichte. Zu bestimmen sei eine Nullstelle der Funktion $\mathcal{G}(\mathcal{X})$, wie diese in (3.77) definiert ist. Wähle einen Startvektor $\mathcal{X}^0 \in \mathcal{M}$ und setze $n = 0$.

do

1. Konstruiere das Voronoi-Diagramm von Ω zum gegebenen \mathcal{X}^n .
2. Wähle α so, dass die Matrix $\hat{\mathcal{H}}(\mathcal{X}^n)$ positiv definit ist.
3. Werte die Matrix $\hat{\mathcal{H}}(\mathcal{X}^n)$ und die Funktion $\mathcal{G}(\mathcal{X}^n)$ aus.
4. Löse das Gleichungssystem $\hat{\mathcal{H}}(\mathcal{X}^n)\Delta\mathcal{X}^n = -\mathcal{G}(\mathcal{X}^n)$.
5. Bestimme $\mathcal{X}^{n+1} = \mathcal{X}^n + \Delta\mathcal{X}^n$.
6. $n = n + 1$.

while $\|\Delta\mathcal{X}^n\| > tol$

bei $\alpha = 1$ das reine Newton-Verfahren.

Experimentell wurde festgestellt, dass bei einer symmetrisch positiv definiten Jacobi-Matrix $\hat{\mathcal{H}}(\mathcal{X}^n)$ alle Generatoren im Definitionsgebiet bleiben. Als eine plausible Erklärung bietet sich dafür folgendes an: Außerhalb des Definitionsgebietes kann die Funktion $\mathcal{F}(\mathcal{X}) = \sum_i \int_{\Omega_i} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^2 d\mathbf{x}$ kein Minimum sondern nur ihr Maximum annehmen. Im Bereich des Maximums muss die Jacobi-Matrix $\hat{\mathcal{H}}(\mathcal{X}^n)$ negativ definit sein. Dies gilt jedoch nur für eine lokale Umgebung des Maximums (siehe [7, 84]). Diese Begründung kann damit nicht als mathematischer Beweis für den oben beschriebenen Zusammenhang gelten. Aufgrund von sehr komplizierten Gestalt der Jacobi-Matrix $\frac{\partial \mathcal{G}}{\partial \mathcal{X}}(\mathcal{X}^n)$ erweist sich die mathematische Betrachtung der Wendestellen von $\mathcal{F}(\mathcal{X})$ als wenig praktikabel. Denn dafür müsste die dritte Ableitung von $\mathcal{F}(\mathcal{X})$ berechnet werden, die einem Tensor dritter Stufe entspricht.

Der angesprochene Zusammenhang ist für die Praxis auch nicht notwendig, denn man kann beide Bedingungen relativ effizient überprüfen. Bei den Generatoren reicht es zu prüfen, ob diese in ihren alten Voronoi-Regionen liegen. Diese Bedingung ist ohnehin für die Durchführbarkeit des in dieser Arbeit entwickelten Algorithmus zur Erzeugung von Voronoi-Diagrammen notwendig (siehe den Unterabschnitt 2.2.2). Wenn die Matrix $\hat{\mathcal{H}}(\mathcal{X}^n)$ beispielsweise nicht positiv definit ist, wird man bei der *Incomplete-Cholesky* auf das Problem der Wurzel aus einer negativen Zahl stoßen (siehe den nächsten Abschnitt).

In der Praxis hat es sich herausgestellt, dass bei einem klein genug gewählten α die Matrix $\hat{\mathcal{H}}(\mathcal{X}^n)$ nicht nur positiv definit, sondern auch irreduzibel diagonal dominant ist (siehe [84]), was den Einsatz des effizienten Präkonditionierers *Incomplete-Cholesky* ermöglicht. Bei irreduziblen Diagonaldominanz muss die Matrix irreduzibel und an einer einzigen Zeile strikt diagonal dominant sein. Die Diagonaldominanz lässt sich leicht erklären: Die Matrix $\tilde{\mathcal{H}}(\mathcal{X}^n)$ ist diagonal dominant. Wenn α klein genug ist, dann ist der Unterschied zwischen $\hat{\mathcal{H}}(\mathcal{X}^n)$ und $\tilde{\mathcal{H}}(\mathcal{X}^n)$ auch klein, so dass die Diagonaldominanz auch bei $\hat{\mathcal{H}}(\mathcal{X}^n)$ erhalten bleibt. Eine ausführlichere Studie zur Wahl von α erfolgt in dem Unterabschnitt 3.6.2.

Bemerkung 3.4:

Bei einer günstigen Wahl von α im Algorithmus 3.3 wird die Matrix $\hat{\mathcal{H}}(\mathcal{X}^n)$ sowohl positiv definit als auch irreduzibel diagonal dominant.

3.5 Auswertung der mehrfachen Integrale

Um die in den Abschnitten 3.3, 3.4 beschriebenen Algorithmen durchführen zu können, müssen die Volumenintegrale in (3.69) und die Flächenintegrale in (3.100)-(3.101) ausgewertet werden. Da die Zelldichte $\rho(\mathbf{x})$ meistens nicht in mathematisch geschlossener Form vorliegt, müssen diese Volumen- und Flächenintegrale durch geeignete Quadraturformeln approximiert werden.

3.5.1 Volumenintegrale

Hier werden wir uns mit den folgenden Integralen beschäftigen:

$$\int_{\Omega_i} \rho(\mathbf{x}) d\mathbf{x} \tag{3.102}$$

$$\int_{\Omega_i} \rho(\mathbf{x}) \mathbf{x} d\mathbf{x}. \tag{3.103}$$

Die Approximation der Integrale in (3.102)-(3.103) sollte möglichst gut sein. In dem Unterabschnitt 4.6.2 wird gezeigt, dass für eine stetig differenzierbare Funktion

$$\phi : \mathbb{R}^3 \mapsto \mathbb{R}, \quad \mathbf{x} \mapsto \phi(\mathbf{x}) \quad (3.104)$$

folgende Gleichung gilt:

$$\int_{\Omega_i} \phi(\mathbf{x}) d\mathbf{x} = \phi(\mathbf{x}_i^*) \int_{\Omega_i} d\mathbf{x} + \int_{\Omega_i} \mathcal{O}(\|\mathbf{x} - \mathbf{x}_i^*\|^2) d\mathbf{x}, \quad (3.105)$$

wobei \mathbf{x}_i^* der geometrische Schwerpunkt des Volumenelements Ω_i ist. D.h. die Approximation des Integrals $\int_{\Omega_i} \phi(\mathbf{x}) d\mathbf{x}$ durch (3.105) hat den Abbruchfehler zweiter Ordnung. Die Näherungen $\int_{\Omega_i} \rho(\mathbf{x}) d\mathbf{x} \approx |\Omega_i| \rho(\mathbf{x}_i^*)$ und $\int_{\Omega_i} \rho(\mathbf{x}) \mathbf{x} d\mathbf{x} \approx |\Omega_i| \rho(\mathbf{x}_i^*) \mathbf{x}_i^*$ sind jedoch nicht zulässig, weil durch solche Approximation folgendes gelten würde:

$$\frac{\int_{\Omega_i} \rho(\mathbf{x}) \mathbf{x} d\mathbf{x}}{\int_{\Omega_i} \rho(\mathbf{x}) d\mathbf{x}} \approx \frac{|\Omega_i| \rho(\mathbf{x}_i^*) \mathbf{x}_i^*}{|\Omega_i| \rho(\mathbf{x}_i^*)} = \mathbf{x}_i^*. \quad (3.106)$$

Das heißt, man würde für den mit der Zelldichte gewichteten Schwerpunkt den ungewichteten Schwerpunkt erhalten. Man sieht hierdurch die Notwendigkeit einer wesentlich genaueren Approximation der Volumenintegrale in (3.102)-(3.103).

In dieser Arbeit verwenden wir folgende Technik, um die Volumenintegrale in (3.102)-(3.103) zu berechnen. Wir zerlegen jedes Polyeder in eine Menge von Tetraedern und wenden Quadraturformeln für Tetraeder an. Angenommen das Polyeder Ω_i lässt sich in eine Menge von Tetraedern $\{T_{ij}\}_{j=1}^{M_i}$ mit $M_i \in \mathbb{N}$ zerlegen, so dass $\bigcap_{j=1}^{M_i} (T_{ij} \setminus \partial T_{ij}) = \emptyset$ und $\bigcup_{j=1}^{M_i} T_{ij} = \Omega_i$. Laut dem Satz 3.2 gilt für eine stetige Funktion

$$g : \Omega_i \mapsto \mathbb{R}, \quad \mathbf{x} \mapsto g(\mathbf{x}), \quad (3.107)$$

folgende Gleichung:

$$\int_{\Omega_i} g(\mathbf{x}) d\mathbf{x} = \sum_{j=1}^{M_i} \int_{T_{ij}} g(\mathbf{x}) d\mathbf{x}. \quad (3.108)$$

Die Gleichung (3.108) können wir sowohl für $\rho(\mathbf{x})$ als auch für $\rho(\mathbf{x}) \mathbf{x}$ verwenden. Um den Approximationsfehler der Volumenintegrale zu reduzieren, soll jedes Ω_i möglichst viele Tetraeder enthalten. Deshalb verwenden wir für die Zerlegung von Ω_i einen zusätzlichen Stützpunkt. Entscheidend ist bei der Wahl des Stützpunktes, dass dieser immer innerhalb des Polyeders liegt. Im Falle eines konvexen Polyeders Ω_i bietet sich dafür der geometrische Schwerpunkt an.

Angenommen \mathbf{x}_i^* ist der Schwerpunkt von Ω_i , dann können wir Ω_i in eine Menge von Pyramiden zerlegen. Jede Pyramide P_{ij} enthält den Eckpunkt \mathbf{x}_i^* und die Teilfläche $\partial\Omega_{ij}$ des Polyeders Ω_i . Die Abbildung 3.3 zeigt diese Teilfläche.

Anschließend wird jede Pyramide mit Hilfe eines zusätzlichen Punktes x_{pc} in Tetraeder zerlegt. Man erhält für jede Pyramide genau so viele Tetraeder wie viele Knoten die Oberfläche $\partial\Omega_{ij}$ enthält (Siehe die Abbildung 3.3). Obwohl man durch den Verzicht auf den Hilfspunkt x_{pc} eine effizientere Zerlegung der Pyramiden erreichen würde, bei der beispielsweise die Pyramide in der Abbildung 3.3 nur 4 Tetraeder enthalten würde, ist dies nicht empfehlenswert. Denn so eine Zerlegung würde unvermeidlich zu einem höheren Approximationsfehler der Integrale in (3.102)-(3.103) führen. In dieser Arbeit verwenden wir für x_{pc} den geometrischen Schwerpunkt der Oberfläche $\partial\Omega_{ij}$.

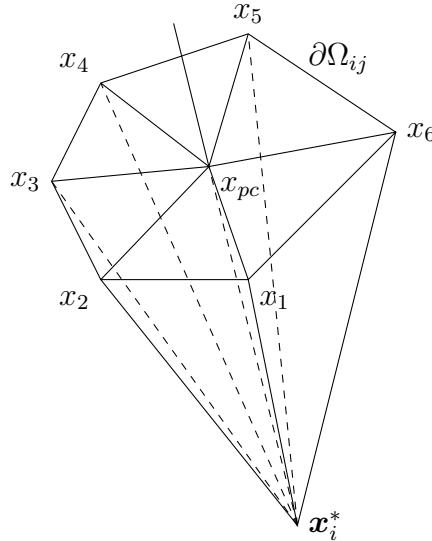


Abbildung 3.3: Zerlegung einer Pyramide in Tetraeder

Die Volumenintegrale für die Tetraeder $\{T_{ij}\}_{j=1}^{M_i}$ werden mit Hilfe der Quadraturformeln berechnet. In der Literatur werden diese für ein Referenztetraeder \tilde{T} mit Eckpunkten

$$\tilde{\mathbf{a}} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \tilde{\mathbf{b}} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \tilde{\mathbf{c}} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \tilde{\mathbf{d}} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

angegeben. Mit Anwendung des Transformationssatzes (Satz 3.4 im Unterabschnitt 3.2.2) können die vorgerechneten Quadraturformeln für ein beliebiges Tetraeder verwendet werden. Die Hauptgleichung des Transformationssatzes lautet:

$$\int_{\Phi(\Omega)} f(\mathbf{x}) d\mathbf{x} = \int_{\Omega} f(\Phi(\mathbf{y})) \left| \det \left(\frac{\partial \Phi(\mathbf{y})}{\partial \mathbf{y}} \right) \right| d\mathbf{y}. \quad (3.109)$$

Für ein Tetraeder T mit Eckpunkten $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$ gilt:

$$\Phi(\mathbf{y}) = \mathbf{A}\mathbf{y} + \mathbf{a}, \text{ mit } \mathbf{A} = (\mathbf{b} - \mathbf{a} \quad \mathbf{c} - \mathbf{a} \quad \mathbf{d} - \mathbf{a}), \quad (3.110)$$

wobei $\mathbf{y} \in \mathbb{R}^3$ ein beliebiger Punkt im Referenztetraeder bezeichnet. Damit gilt:

$$\int_T \rho(\mathbf{x}) d\mathbf{x} = \int_{\tilde{T}} \rho(\Phi(\mathbf{y})) \det(\mathbf{A}) d\mathbf{y} = \det(\mathbf{A}) \int_{\tilde{T}} \rho(\Phi(\mathbf{y})) d\mathbf{y} \quad (3.111)$$

$$\int_T \rho(\mathbf{x}) \mathbf{x} d\mathbf{x} = \int_{\tilde{T}} \rho(\Phi(\mathbf{y})) \Phi(\mathbf{y}) \det(\mathbf{A}) d\mathbf{y} = \det(\mathbf{A}) \int_{\tilde{T}} \rho(\Phi(\mathbf{y})) \Phi(\mathbf{y}) d\mathbf{y}. \quad (3.112)$$

Die Integrale in (3.111)-(3.112) können mit Hilfe von Quadraturformeln mit unterschiedlichem Genauigkeitsgrad approximiert werden. Eine Quadraturformel hat den Genauigkeitsgrad P , wenn sie alle Polynome bis zum Höchstgrad P exakt integriert. Mit Anwendung einer Quadraturfor-

mel mit N_s Stützstellen auf (3.111) und (3.112) erhält man

$$\int_T \rho(\mathbf{x}) d\mathbf{x} \approx \frac{|\det(\mathbf{A})|}{6} \sum_{k=1}^{N_s} \omega_k \rho(\Phi(\mathbf{y}_k)) \quad (3.113)$$

$$\int_T \rho(\mathbf{x}) \mathbf{x} d\mathbf{x} \approx \frac{|\det(\mathbf{A})|}{6} \sum_{k=1}^{N_s} \omega_k \rho(\Phi(\mathbf{y}_k)) \Phi(\mathbf{y}_k). \quad (3.114)$$

Der Faktor $\frac{1}{6}$ ist ein Teil der Gewichte ω_k und ist gleich dem Volumen des Referenztetraeders. Für das Polyeder Ω_i würde man mit (3.108) und (3.113)-(3.114)

$$\int_{\Omega_i} \rho(\mathbf{x}) d\mathbf{x} \approx \sum_{j=1}^{M_i} \frac{|\det(\mathbf{A}_{ij})|}{6} \sum_{k=1}^{N_s} \omega_k \rho(\Phi_{ij}(\mathbf{y}_k)) \quad (3.115)$$

$$\int_{\Omega_i} \rho(\mathbf{x}) \mathbf{x} d\mathbf{x} \approx \sum_{j=1}^{M_i} \frac{|\det(\mathbf{A}_{ij})|}{6} \sum_{k=1}^{N_s} \omega_k \rho(\Phi_{ij}(\mathbf{y}_k)) \Phi_{ij}(\mathbf{y}_k) \quad (3.116)$$

erhalten, wobei \mathbf{A}_{ij} und $\Phi_{ij}(\mathbf{y})$ bei jedem Tetraeder wie in (3.110) zu verstehen sind.

Eine gute Beschreibung vieler verschiedener Klassen von Quadraturformeln wird in [58] angeboten. Eine wichtige Klasse von Quadraturformeln ergibt sich durch den Ansatz, den Integrand durch ein Interpolationspolynom vom Grad P zu approximieren und dieses zu integrieren. Die Zelldichte $\rho(\mathbf{x}) = \left(\frac{1}{\mathcal{L}(\mathbf{x})}\right)^5$ lässt sich aufgrund ihres asymptotischen Teiles $\frac{1}{\mathcal{L}(\mathbf{x})}$ durch kein Polynom exakt zu interpolieren. Daher fällt es schwer den geeigneten Genauigkeitsgrad der Quadraturformel für das Integrand $\rho(\mathbf{x})\mathbf{x}$ zu bestimmen. Mit Sicherheit kann man dennoch sagen, dass dafür zumindest die Quadraturformel des zweiten Grades notwendig ist. Denn schon bei einer linearen Zelldichte $\rho(\mathbf{x})$ ist die Funktion $\rho(\mathbf{x})\mathbf{x}$ quadratisch. D.h. die Quadraturformel des zweiten Genauigkeitsgrades würde erst eine lineare Näherung der Zelldichte $\rho(\mathbf{x})$ ausmachen. Die Stützstellen und die Gewichte der Quadraturformel des zweiten Grades sind in der Tabelle 3.1 zu sehen.

ω_k	y_k^1	y_k^2	y_k^3
$\frac{1}{4}$	$\frac{5-\sqrt{5}}{20}$	$\frac{5-\sqrt{5}}{20}$	$\frac{5-\sqrt{5}}{20}$
$\frac{1}{4}$	$\frac{5+3\sqrt{5}}{20}$	$\frac{5-\sqrt{5}}{20}$	$\frac{5-\sqrt{5}}{20}$
$\frac{1}{4}$	$\frac{5-\sqrt{5}}{20}$	$\frac{5+3\sqrt{5}}{20}$	$\frac{5-\sqrt{5}}{20}$
$\frac{1}{4}$	$\frac{5-\sqrt{5}}{20}$	$\frac{5-\sqrt{5}}{20}$	$\frac{5+3\sqrt{5}}{20}$

Tabelle 3.1: Gaußsche Quadraturformel des zweiten Grades für das Referenztetraeder

Wir haben untersucht, welche Auswirkung der Genauigkeitsgrad der verwendeten Quadraturformel beim Lloyd-Algorithmus auf das CV-Gitter haben kann. Dabei wurde eine C++-Bibliothek benutzt, die John Burkardt auf seiner Internetseite [36] bereitgestellt hat. Diese Bibliothek dient dazu, die Quadraturformeln beliebigen Grades zu berechnen. Wir verwenden hier die sogenannten *Keast-Rules*. Die Abbildung 3.4 zeigt die Berechnungsergebnisse. Die Tatsache,

Kapitel 3. Gitterbewegung mithilfe von CVT

dass alle Gitter sehr unterschiedliche Muster haben, zeigt wie empfindlich die Auswertung der Volumenintegrale von der gewählten Quadraturformel ist. Der Abbildung 3.4 kann man entnehmen, dass die meist regelmäßigen Gitter bei dem fünften Grad und dem sechsten Grad entstehen. Die Quadraturformel des ersten Genauigkeitsgrades bewirkt im Vergleich zu anderen Quadraturformeln ein viel größeres Kantenlängenverhältnis, was auf ihren viel zu höheren Approximationsfehler deutet.

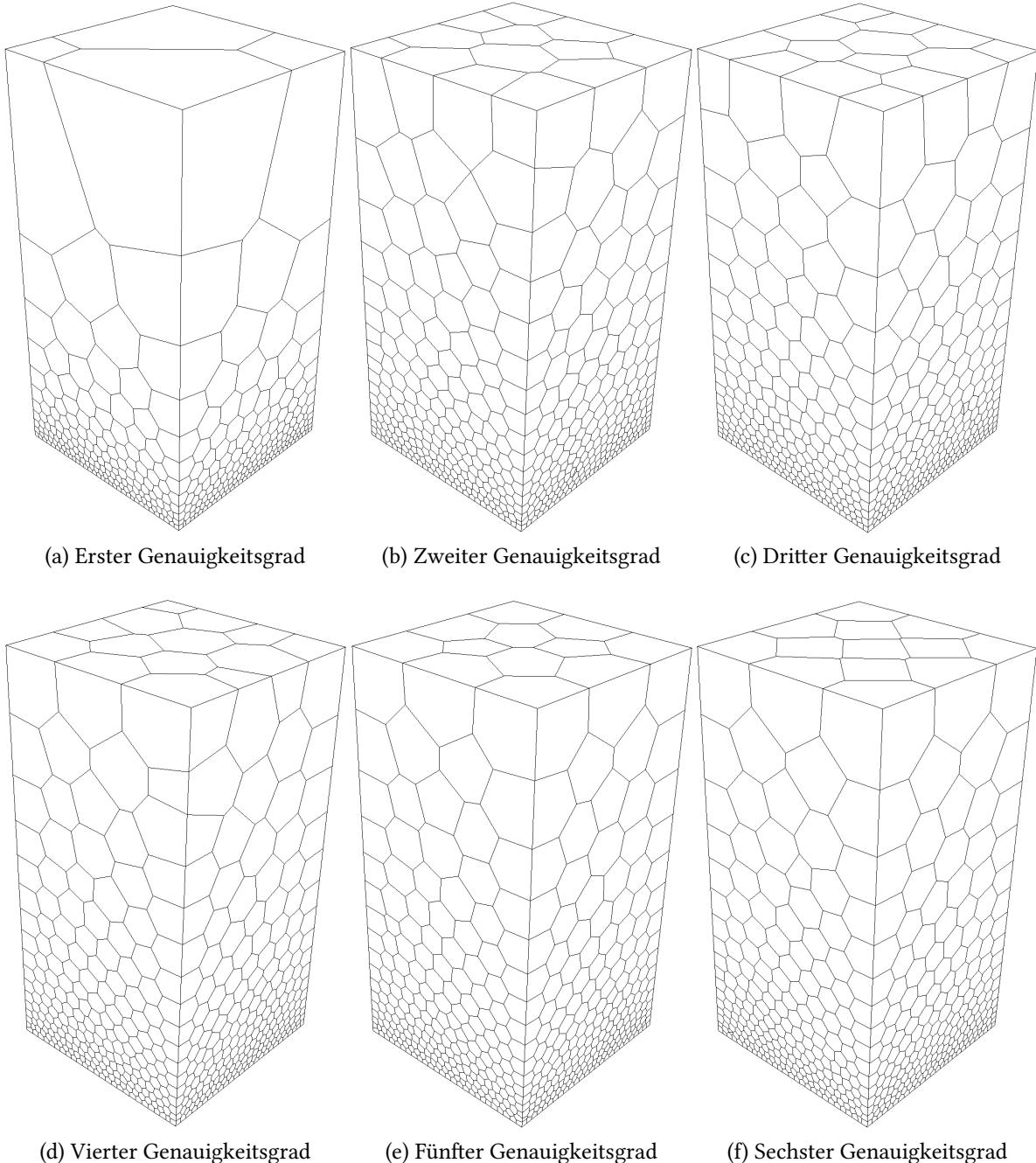


Abbildung 3.4: CVT eines rechteckigen Prismas der Länge 2 mit einem Kantenverhältnis von 0.1 zu 1. D.h. die Kantenlänge ist eine lineare Funktion des Ortes: $\mathcal{L}(\mathbf{x}) = 0.1 \cdot (2 - x^1) + 1 \cdot x^1$. Dabei ist x^1 die Koordinate der Längsrichtung des Prismas. Quadraturformeln des Genauigkeitsgrades von 1 bis 6. Der Lloyd-Algorithmus wurde bis zum absoluten Fixpunkt durchgeführt. Das Modell enthält 3651 Zellen.

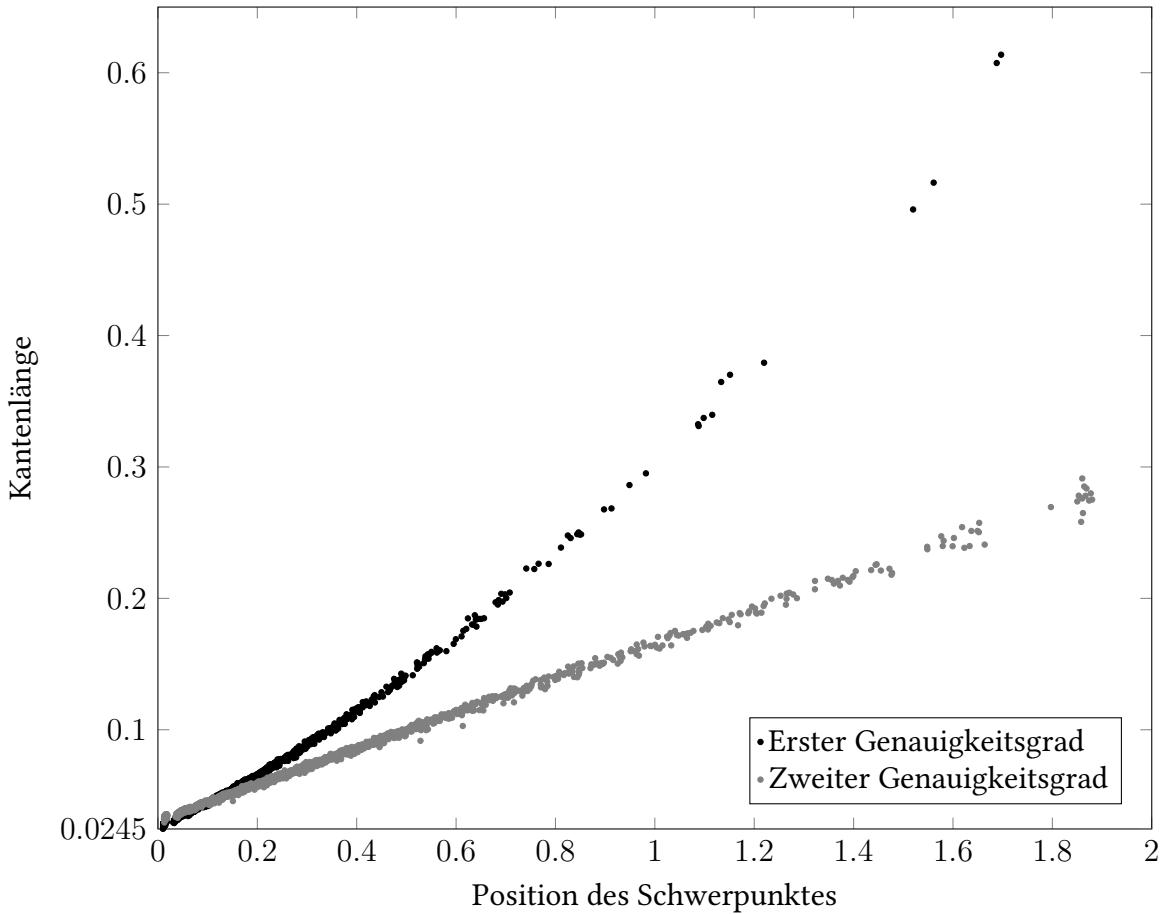


Abbildung 3.5: Die erreichte Kantenlänge gegenüber der Ortskoordinate

Um die verwendeten Quadraturformeln quantitativ zu bewerten, gehen wir wie folgt vor. Das Modell und die Zelldichte sind dabei wie in der Beschriftung der Abbildung 3.4 gegeben. Für jedes Volumenelement Ω_i berechnen wir

$$\mathcal{L}_i = \frac{1}{\sqrt[3]{|V_i|}}, \quad (3.117)$$

was etwa der real erreichten mittleren Kantenlänge dieses Elements entspricht. Der Wert \mathcal{L}_i wird dem auf die x^1 -Achse projizierten geometrischen Schwerpunkt jedes Volumenelements zugeordnet. Die Abbildung 3.5 zeigt diese Wertepaare für Quadraturformeln des ersten und des zweiten Genauigkeitsgrades. Hier ist gut erkennbar, dass die erreichte Kantenlänge beim zweiten Genauigkeitsgrad wie eine Gerade aussieht, was auch der Vorgabe entspricht. Der Verlauf der Kantenlänge beim ersten Genauigkeitsgrad sieht eher wie eine Parabel aus. Dies dient als ein praktischer Beleg, dass die Quadraturformel des ersten Grades für die Approximation der Integrale in (3.102)-(3.103) nicht geeignet ist. Weiterhin haben wir anhand dieser Wertepaare Ausgleichsgeraden bestimmt. Für die Allgemeinform einer Gerade

$$\mathcal{L}(x^1) = ax^1 + b \quad (3.118)$$

erhält man die Koeffizienten durch die folgenden Gleichungen:

$$a = \frac{\sum_{i=1}^n (x_i^1 - \bar{x}^1)(\mathcal{L}_i - \bar{\mathcal{L}})}{\sum_{i=1}^n (x_i^1 - \bar{x}^1)^2} \quad (3.119)$$

$$b = \bar{\mathcal{L}} - ax^1. \quad (3.120)$$

Kapitel 3. Gitterbewegung mithilfe von CVT

Die Gleichungen in (3.119)-(3.120) kommen durch die sogenannte lineare Regression zustande, was ein sehr bekanntes Problem ist, weshalb wir hier auf die Angabe der Literaturquellen verzichten. Da die Länge des Prismas gleich 2 ist, sollte bei einer korrekten Verteilung der Zellen

$$\frac{\mathcal{L}(2)}{\mathcal{L}(0)} = \frac{2a + b}{b} \approx 10 \quad (3.121)$$

gelten. Anhand der Tabelle 3.2 kann man feststellen, dass $\frac{\mathcal{L}(2)}{\mathcal{L}(0)}$ beim fünften Genauigkeitsgrad den höchsten Wert annimmt. Danach sinkt dieser Wert mit dem steigenden Genauigkeitsgrad. Dies zeugt davon, dass es keine Konvergenz der Quadraturformeln für das untersuchte Problem gibt. Es scheint lediglich eine optimale Quadraturformel zu geben. Dies kann damit zusammenhängen, dass mit dem steigenden Genauigkeitsgrad der Quadraturformel die Anzahl der Stützstellen und damit die Anzahl der Rechenoperationen wächst. Dies führt zu einem größeren numerischen Gesamtfehler.

Beim Vergleich der Werte in der Tabelle 3.2 kann man auch feststellen, dass das Ergebnis des zweiten Genauigkeitsgrades sich nur unwesentlich von dem Ergebnis des fünften Genauigkeitsgrades unterscheidet.

Grad der Quadraturformel	Anzahl der Stützstellen	Kantenlängenverhältnis
1	4	24.7415116353758648
2	5	9.40678019307457092
3	10	9.40097773505222811
4	11	9.42872549625578138
5	15	9.47183851875520588
6	24	9.45902400944795474
7	31	9.42490063263038635
8	45	9.34937313069549703

Tabelle 3.2: Das erreichte Kantenlängenverhältnis bei verschiedenen Genauigkeitsgraden der Quadraturformeln. Der Sollwert des Kantenlängenverhältnisses ist 10. Es handelt sich um die *Keast-Rules*.

Bemerkung 3.1:

Für die Berechnung der Volumenintegrale in (3.102)-(3.103) eignen sich die Quadraturformeln des zweiten und höherer Genauigkeitsgrade.

Bemerkung 3.2:

Aufgrund des zu hohen Rechenaufwandes steigt der Approximationsfehler ab der Quadraturformel des siebenten Grades.

3.5.2 Flächenintegrale

In diesem Unterabschnitt beschäftigen wir uns mit der Approximation der folgenden Flächenintegrale:

$$\int_{\partial\Omega_{ij}} \rho(\mathbf{x}) \frac{(\mathbf{x} - \mathbf{x}_i)(\mathbf{x} - \mathbf{x}_i)^T}{\|\mathbf{x}_j - \mathbf{x}_i\|} d\mathbf{x} \quad (3.122)$$

$$\int_{\partial\Omega_{ij}} \rho(\mathbf{x}) \frac{(\mathbf{x}_i - \mathbf{x})(\mathbf{x}_j - \mathbf{x})^T}{\|\mathbf{x}_j - \mathbf{x}_i\|} d\mathbf{x}. \quad (3.123)$$

Die Integrale in (3.122)-(3.123) müssen beim Lloyd-Newton-Algorithmus im Abschnitt 3.4 ausgewertet werden. Ähnlich wie im Unterabschnitt 3.5.1 werden die Vielecke in Dreiecke zerlegt, deren Integrale mithilfe von Quadraturformeln berechnet werden. Die Vorschrift der Zerlegung ist in der Abbildung 3.6 zu sehen.

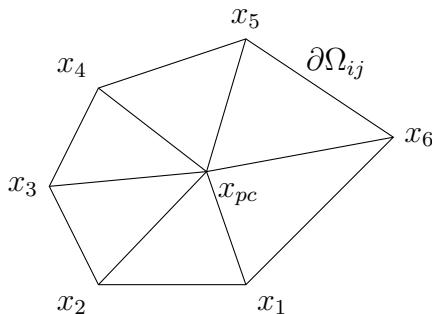


Abbildung 3.6: Zerlegung der Flächenelemente in Dreiecke.

Mit einfachen Umformungen erhalten wir aus (3.122):

$$\int_{\partial\Omega_{ij}} \rho(\mathbf{x}) \frac{(\mathbf{x} - \mathbf{x}_i)(\mathbf{x} - \mathbf{x}_i)^T}{\|\mathbf{x}_j - \mathbf{x}_i\|} d\mathbf{x} = \frac{1}{\|\mathbf{x}_j - \mathbf{x}_i\|} \int_{\partial\Omega_{ij}} \rho(\mathbf{x})(\mathbf{x} - \mathbf{x}_i)(\mathbf{x} - \mathbf{x}_i)^T d\mathbf{x} \quad (3.124)$$

$$= \frac{\int_{\partial\Omega_{ij}} \rho(\mathbf{x}) \mathbf{x} \mathbf{x}^T d\mathbf{x} - \int_{\partial\Omega_{ij}} \rho(\mathbf{x}) \mathbf{x} d\mathbf{x} \mathbf{x}_i^T - \mathbf{x}_i \int_{\partial\Omega_{ij}} \rho(\mathbf{x}) \mathbf{x}^T d\mathbf{x} + \mathbf{x}_i \mathbf{x}_i^T \int_{\partial\Omega_{ij}} \rho(\mathbf{x}) d\mathbf{x}}{\|\mathbf{x}_j - \mathbf{x}_i\|}. \quad (3.125)$$

Wir zeigen, wie das Flächenintegral $\int_{\partial\Omega_{ij}} \rho(\mathbf{x}) \mathbf{x} \mathbf{x}^T d\mathbf{x}$ durch die Quadraturformel mit N_s Stützstellen angenähert werden kann. Mit Anwendung des Transformationssatzes erhält man

$$\int_{\partial\Omega_{ij}} \rho(\mathbf{x}) \mathbf{x} \mathbf{x}^T d\mathbf{x} \approx \sum_{j=1}^{M_{ij}} \frac{\beta_{ij}}{2} \sum_{k=1}^{N_s} \omega_k \rho(\Phi_{ij}(\mathbf{y}_k)) \Phi_{ij}(\mathbf{y}_k) (\Phi_{ij}(\mathbf{y}_k))^T, \quad (3.126)$$

wobei

$$\Phi_{ij}(\mathbf{y}) := \mathbf{A}_{ij}\mathbf{y} + \mathbf{a}_{ij}, \text{ mit } \mathbf{A}_{ij} := (\mathbf{b}_{ij} - \mathbf{a}_{ij} \quad \mathbf{c}_{ij} - \mathbf{a}_{ij}), \quad (3.127)$$

$$\beta_{ij} := \|(\mathbf{b}_{ij} - \mathbf{a}_{ij}) \times (\mathbf{c}_{ij} - \mathbf{a}_{ij})\| \quad (3.128)$$

und $\sum_{j=1}^{M_{ij}}$ die Summe über alle Dreiecke mit Eckpunkten \mathbf{a}_{ij} , \mathbf{b}_{ij} , \mathbf{c}_{ij} des Flächenelements $\partial\Omega_{ij}$ sind.

Eine Berücksichtigung der Ergebnisse aus dem Unterabschnitt 3.5.1 führt uns zu folgendem Rückschluss. Mit einer linearen Näherung der Zelldichte $\rho(\mathbf{x})$ ist mindestens die Quadraturformel des dritten Genauigkeitsgrades notwendig, um den Approximationsfehler bei dem Integral $\int_{\partial\Omega_{ij}} \rho(\mathbf{x}) \mathbf{x} \mathbf{x}^T d\mathbf{x}$ in Grenzen zu halten. Dabei ist zu beachten, dass der Lloyd-Newton-Algorithmus gegenüber den Fehlern in der Jacobi-Matrix $\hat{\mathcal{H}}(\mathcal{X})$ in (3.100)-(3.101) wesentlich empfindlicher als der Lloyd-Algorithmus gegenüber den Fehlern in der Berechnung der gewichteten Schwerpunkte ist. Die Quadraturformel des dritten Genauigkeitsgrades für das Referenzdreieck ist in der Tabelle 3.3 angegeben.

ω_k	y_k^1	y_k^2
$\frac{3}{60}$	0	0
$\frac{3}{60}$	1	0
$\frac{3}{60}$	0	1
$\frac{8}{60}$	$\frac{1}{2}$	0
$\frac{8}{60}$	$\frac{1}{2}$	$\frac{1}{2}$
$\frac{8}{60}$	0	$\frac{1}{2}$
$\frac{27}{60}$	$\frac{1}{3}$	$\frac{1}{3}$

Tabelle 3.3: Quadraturformel des dritten Genauigkeitsgrades für das Referenzdreieck

3.6 Konvergenz

Um die Konvergenz der in diesem Kapitel beschriebenen Algorithmen zu analysieren, erzeugen wir mehrere Gitter mit unterschiedlicher Anzahl von Elementen. Die Fehlerreduktion untersuchen wir anhand von

$$\delta^n := \frac{\|\mathcal{X}^n - \mathcal{X}^*\|}{\|\mathcal{X}^{n-1} - \mathcal{X}^*\|}, \quad (3.129)$$

wobei \mathcal{X}^n bzw. \mathcal{X}^{n-1} der Vektor mit Generatoren im n -ten bzw. $(n-1)$ -ten Schritt und \mathcal{X}^* der Fixpunkt (Lösung) sind. Bei einer konvergenten Fixpunktiteration muss

$$0 < \delta^n < 1, \quad \forall n \quad (3.130)$$

gelten. Je kleiner δ^n desto schneller reduziert sich der Fehler. Im Falle einer linearen Fixpunktiteration ist der Term δ^n konstant. Die Konvergenzrate wird hier durch

$$\sigma^n := -\ln(\delta^n) = -\ln \left(\frac{\|\mathcal{X}^n - \mathcal{X}^*\|}{\|\mathcal{X}^{n-1} - \mathcal{X}^*\|} \right) \quad (3.131)$$

definiert. Im Gegensatz zur Fehlerreduktion gilt hierbei: Je weniger Iterationen bis zum Erreichen des Fixpunktes benötigt werden, desto höher ist die Konvergenzrate.

Bei der Untersuchung der Konvergenz wird das in der Abbildung 3.7 dargestellte Modell verwendet. Es ist sehr aufschlussreich den relativen Fehler im n -ten Schritt

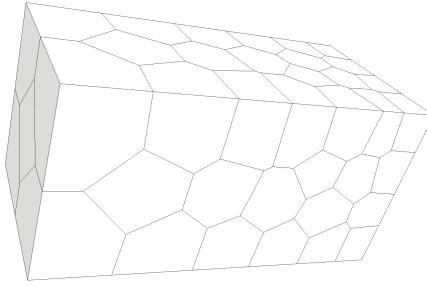


Abbildung 3.7: Orthogonales Prisma. Abmessungen: H/B/L: 1m/1m/2m

$$e^n := \frac{\|\mathcal{X}^n - \mathcal{X}^*\|}{\|\mathcal{X}^*\|} \quad (3.132)$$

zu betrachten. Beim Erreichen eines Fixpunktes sollte e^n unabhängig von dem untersuchten Gitter im Bereich der Maschinentoleranz liegen. Für die Auswertung von δ^n und e^n muss zuerst der Fixpunkt \mathcal{X}^* berechnet werden. Es ist zu beachten, dass verschiedene Algorithmen zur Konstruktion von CVT auch zu verschiedenen Fixpunkten führen können. In dieser Arbeit hat es sich herausgestellt, dass der Lloyd-Algorithmus und der Lloyd-Newton-Algorithmus bei gleichen Startwerten meistens verschiedene Grenzwerte haben. D.h. der Fixpunkt muss extra für jeden der verwendeten Algorithmen berechnet werden, um δ^n und e^n korrekt auszuwerten. Das Erreichen eines Fixpunktes kann man anhand von

$$\|\Delta\mathcal{X}^n\| := \|\mathcal{X}^n - \mathcal{X}^{n-1}\| \quad (3.133)$$

feststellen. In der Nähe eines Fixpunktes konvergiert $\|\Delta\mathcal{X}^n\|$ gegen einen kleinen Wert, der sich nicht mehr ändert. Es ist jedoch zu beachten, dass $\|\Delta\mathcal{X}^n\|$ am Fixpunkt bei höherer Anzahl der Generatoren auch größere Werte annimmt.

3.6.1 Konvergenzrate beim Lloyd-Algorithmus

Für die Approximation der Volumenintegrale wird hier die Quadraturformel des sechsten Genauigkeitsgrades verwendet. Zuerst wollen wir uns den Term e^n bei homogener bzw. inhomogener Zelldichte anschauen. Anhand der Abbildungen 3.8a bzw. 3.8b kann man feststellen, dass der Lloyd-Algorithmus zum Erreichen des „maschinellen“ Fixpunktes relativ viele Iterationen benötigt. Im Falle der homogenen Zelldichte sind bei dem feinsten Gitter mit 3651 Zellen sogar 10 Tausend Iterationen nicht ausreichend. Bei der inhomogenen Zelldichte sind davon mehrere Gitter mit 637, 1519 und 3651 Zellen betroffen. Das heißt, wir können die Aussage in [61, 62] über eine sinkende Konvergenz des Lloyd-Algorithmus mit steigender Anzahl der Elemente bestätigen. Der Zusammenhang der Fehlerreduktion mit der Zellenanzahl ist beinahe linear. Das heißt für ein Gitter mit einer Million Zellen bräuchte man etwa

$$\frac{10^4 \cdot 10^6}{1519} \approx 6 \cdot 10^6 \quad (3.134)$$

Iterationen, um den Fixpunkt zu erreichen. Es gibt zu bedenken, dass die praxisrelevanten Modelle bis zu mehrere Millionen Elemente beinhalten können.

Weiterhin ist folgende Aussage von Interesse. Es ist offensichtlich, dass der Lloyd-Algorithmus in keiner Weise eine lineare Fixpunktiteration ist. Erst nach einer gewissen Anzahl von Iterationen zeigt sich ein lineares Konvergenzverhalten, das in der Abbildung 3.8 einem in logarithmischer Skalierung linearen Abfall von e^n entspricht. In der Abbildung 3.9 entspricht diesem Verlauf ein beinahe konstanter Wert von δ^n . Interessant ist dabei, dass in den ersten Schritten δ^n

wesentlich kleiner ist. Dies kann als positiv angesehen werden, wenn nur wenige Schritte des Lloyd-Algorithmus als eine Art Korrektur durchgeführt werden müssen.

Anhand der Abbildung 3.8 lässt sich auch feststellen, dass der Lloyd-Algorithmus bei der homogenen Zelldichte schneller den Fixpunkt als bei der inhomogenen Zelldichte erreicht. Dabei kommen in der Praxis die Gitter mit der homogenen Zelldichte sehr selten vor.

Zum Schluss beschäftigen wir uns noch mit der Frage, ob es eine Abhängigkeit zwischen der Konvergenz des Lloyd-Algorithmus und dem Gradienten der Zelldichte gibt. D.h. wie stark unterscheidet sich die Zelldichte von einem Rand zum anderen. Der Abbildung 3.10 kann man entnehmen, dass der Lloyd-Algorithmus bei höheren Gradienten der Zelldichte auch höhere Konvergenzrate besitzt. Dies ist besonders in den ersten 1000 Iterationen zu beobachten (Siehe Abbildung 3.10a). Dieses Verhalten war nicht unmittelbar zu erwarten, da die homogene Zelldichte bei allen untersuchten Gittern zu höheren Konvergenzraten führte.

Anbetacht der erwähnten Tatsachen kann man es sich schwer vorstellen, dass der Lloyd-Algorithmus für die Gitterbewegung geeignet sein könnte. Im Abschnitt 3.7 wird es jedoch gezeigt, dass dieser Algorithmus für diesen Zweck verwendet werden kann.

Bemerkung 3.1 (Zusammenhang der Konvergenz mit der Anzahl der Zellen):

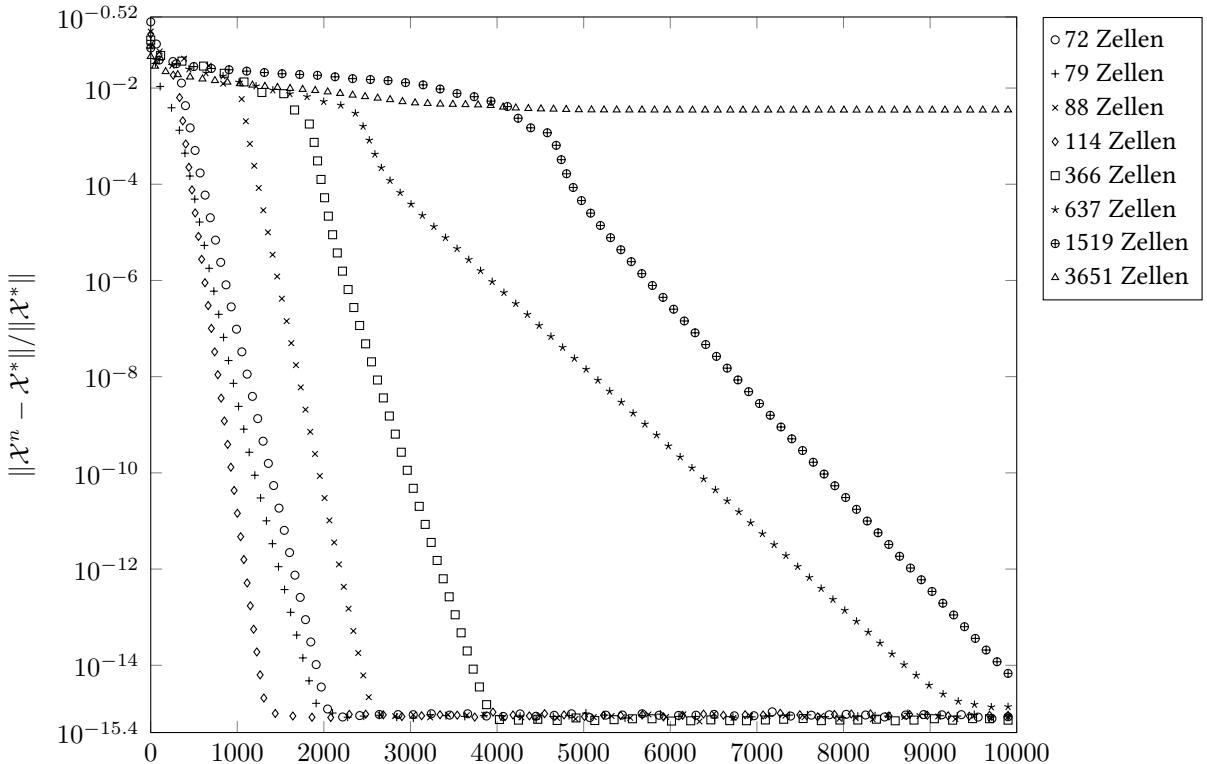
Die Konvergenzrate des Lloyd-Algorithmus sinkt mit wachsender Anzahl der Generatoren, was auch die Aussage in [61, 62] bestätigt. Die Anzahl der Iterationen bis zum Grenzwert lässt sich durch etwa die Anzahl der Generatoren multipliziert mit 6 bestimmen.

Bemerkung 3.2:

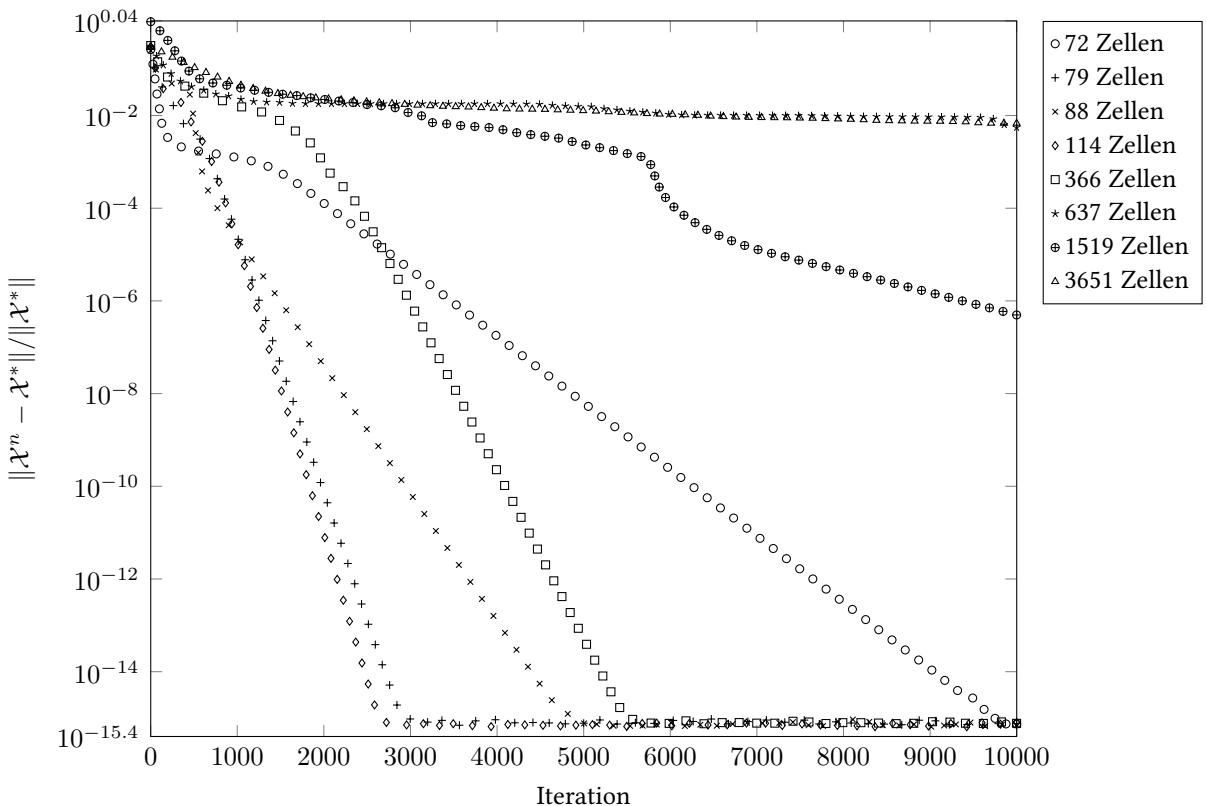
Zu beachten ist dabei, dass der Lloyd-Algorithmus in seinen ersten Schritten, oder auch bei größten Abweichungen von dem Fixpunkt, höhere Konvergenzraten hat.

Bemerkung 3.3:

Für höhere Gradienten der Zelldichte sind höhere Konvergenzraten des Lloyd-Algorithmus zu erwarten.

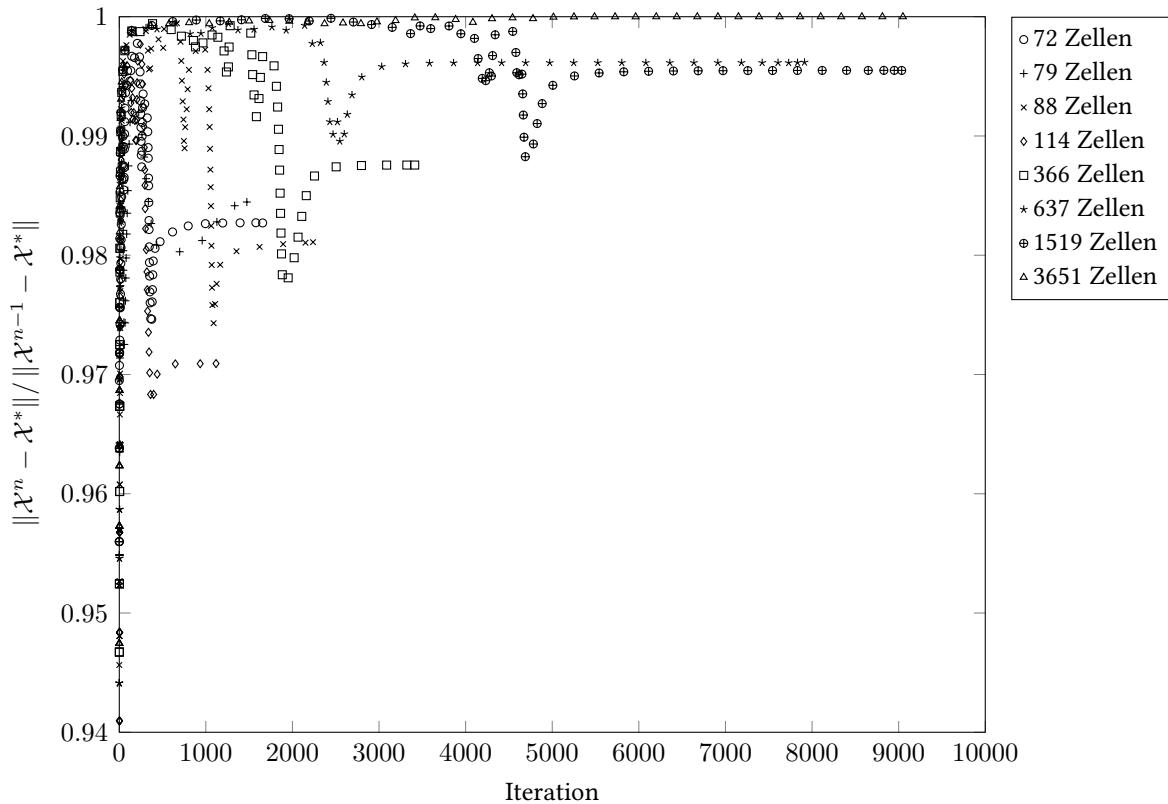


(a) Homogene Zelldichte.

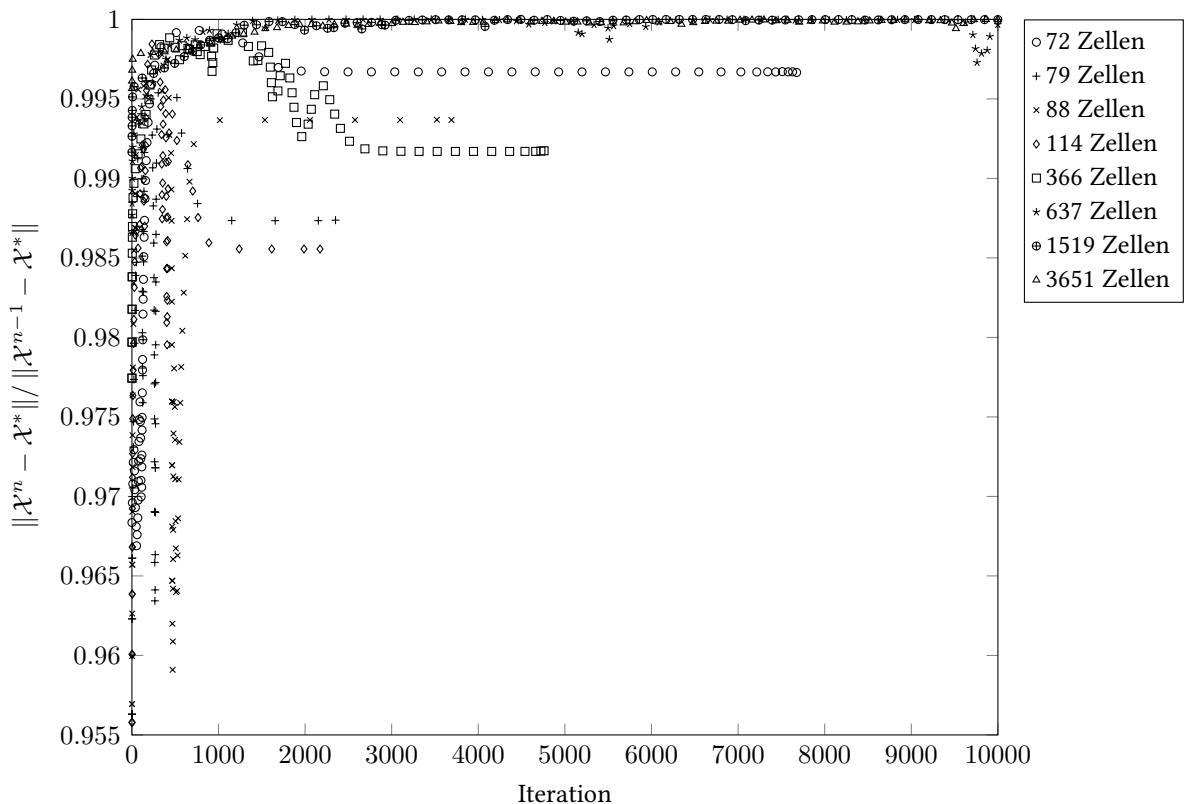


(b) Inhomogene Zelldichte. Die Kantenlänge beträgt bezüglich der Abb. 3.7 links 1 und rechts 0.5.

Abbildung 3.8: Konvergenz von Lloyd-Algorithmus. Acht verschiedene Gitter: von 72 bis 3651 Zellen. Modell ist in der Abb. 3.7 dargestellt.

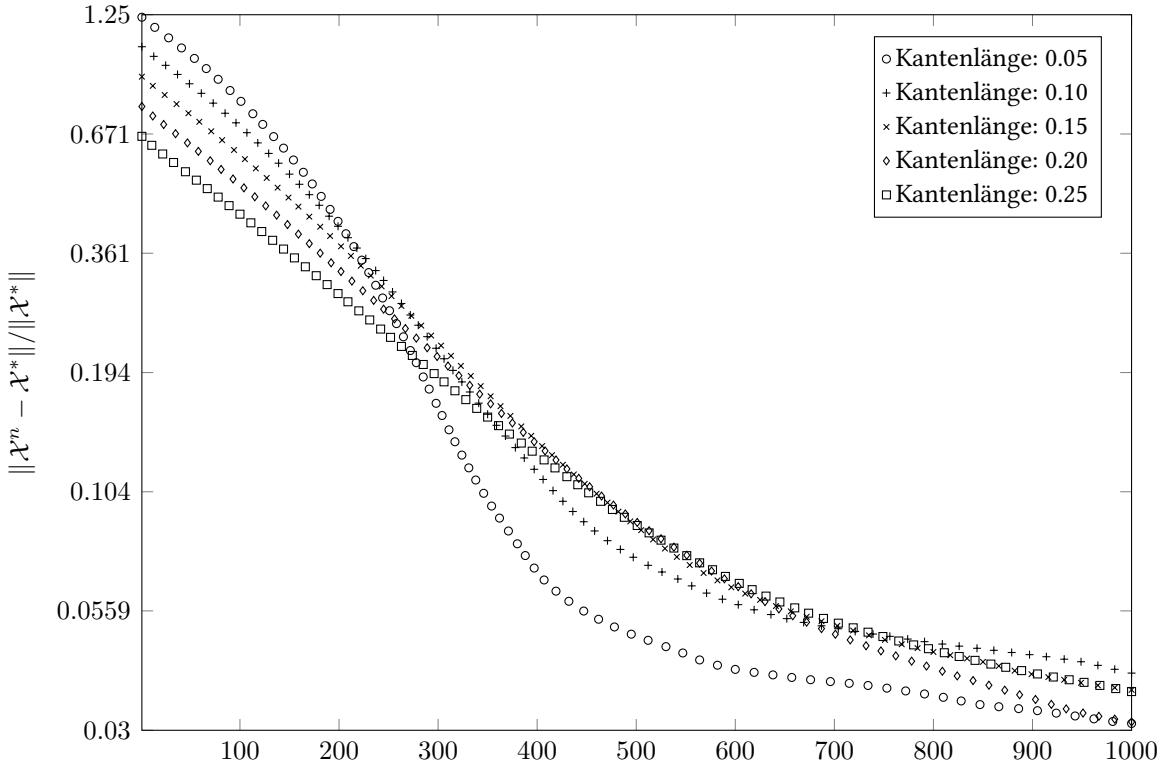


(a) Homogene Zelldichte.

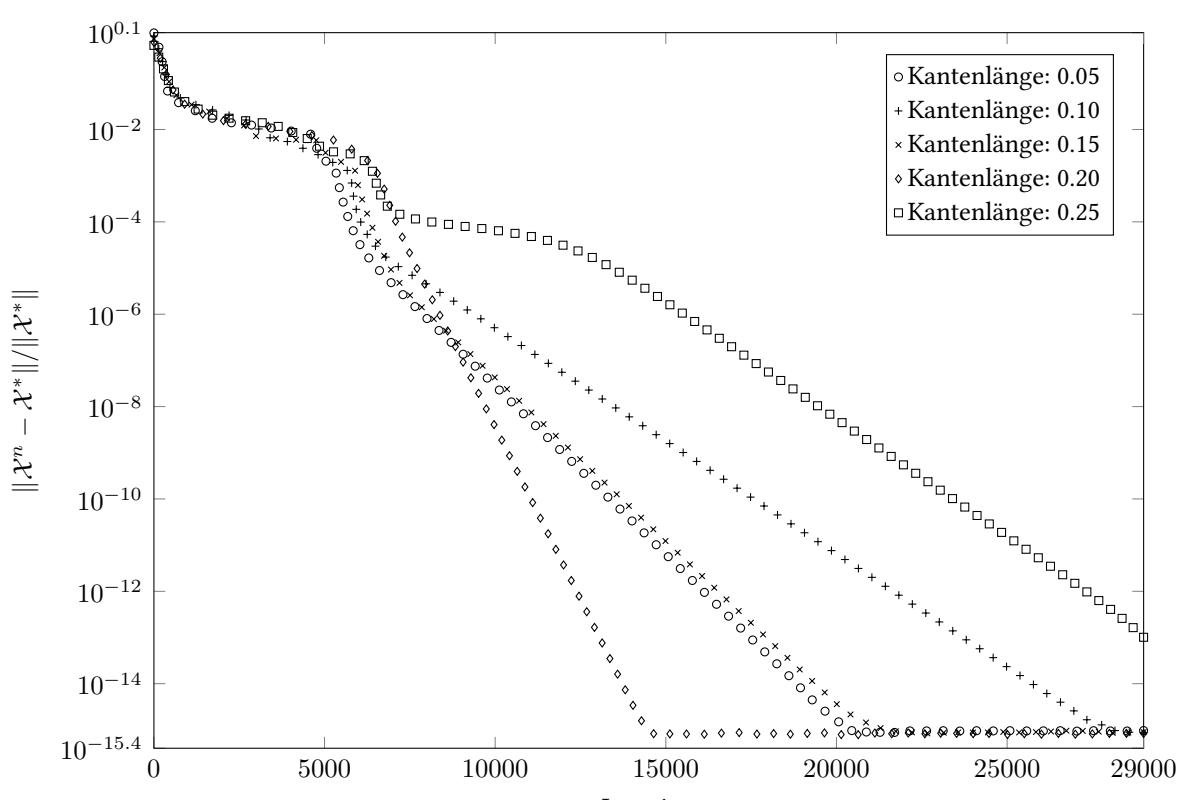


(b) Inhomogene Zelldichte. Die Kantenlänge beträgt bezüglich der Abb. 3.7 links 1 und rechts 0.5.

Abbildung 3.9: Fehlerreduktion beim Lloyd-Algorithmus. Acht verschiedene Gitter: von 72 bis 3651 Zellen. Modell ist in Abb. 3.7 dargestellt.



(a) Erste 1000 Iterationen.



(b) Bis 29000 Iterationen.

Abbildung 3.10: Abhangigkeit der Konvergenz vom Gradienten der Zelldichte beim Lloyd-Algorithmus. Im Bezug auf die Abbildung 3.7 betragt die Kantenlange rechts 1 und links wie in der Legende angegeben.

3.6.2 Konvergenzrate beim alternativen Lloyd-Newton-Algorithmus

In diesem Unterabschnitt werden keine homogenen Gitter untersucht, da diese bei praxisrelevanten Problemstellungen keine Verwendung finden. Für die Berechnung der Volumenintegrale bzw. der Flächenintegrale wurden hier die Quadraturformeln des sechsten bzw. des siebenten Genauigkeitsgrades verwendet.

Die Abbildung 3.11 zeigt e^n für den Lloyd-Newton-Algorithmus bei verschiedenen α . Höhere Werte als 0.3 konnten für α nicht erreicht werden, da die in dieser Arbeit entwickelte Voronoi-Zerlegung bei großen Verschiebungen der Generatoren nicht durchführbar ist.

Beim Betrachten der Abbildung 3.11a fällt es auf den ersten Blick auf, dass die Konvergenzrate beim Lloyd-Newton-Algorithmus in den ersten 400 Schritten so gut wie linear abhängig vom Parameter α ist. In den weiteren Schritten ist diese Abhängigkeit nicht mehr gegeben (Vergleiche auch die Abbildung 3.11b). Anhand der Abbildung 3.11b kann man jedoch folgende Feststellung machen: Zum Erreichen eines Fixpunktes benötigt der Lloyd-Newton-Algorithmus für alle untersuchten α weniger Iterationen als der Lloyd-Algorithmus. Mit $\alpha = 0.15$ stellt sich schon nach 16000 Iterationen der gleiche Zustand wie beim Lloyd-Algorithmus erst nach 27000 Schritten ein.

Die Abbildung 3.11a lässt vermuten, dass der Lloyd-Newton-Algorithmus in den ersten Iterationen für größere α auch höhere Konvergenzraten besitzt. D.h. bei einem an der Eins grenzenden α könnte eine beinahe quadratische Konvergenz des klassischen Newton-Verfahrens erreicht werden.

Die Abbildung 3.12 veranschaulicht die Verbesserung der Fehlerreduktion des Lloyd-Newton-Algorithmus gegenüber dem Lloyd-Algorithmus. Auch beim Lloyd-Newton-Algorithmus erkennt man eine strenge Abhängigkeit der Konvergenzrate von der Anzahl der Generatoren, die darauf beruht, dass dieser Algorithmus einen Mischterm des Lloyd-Algorithmus enthält.

Bemerkung 3.4:

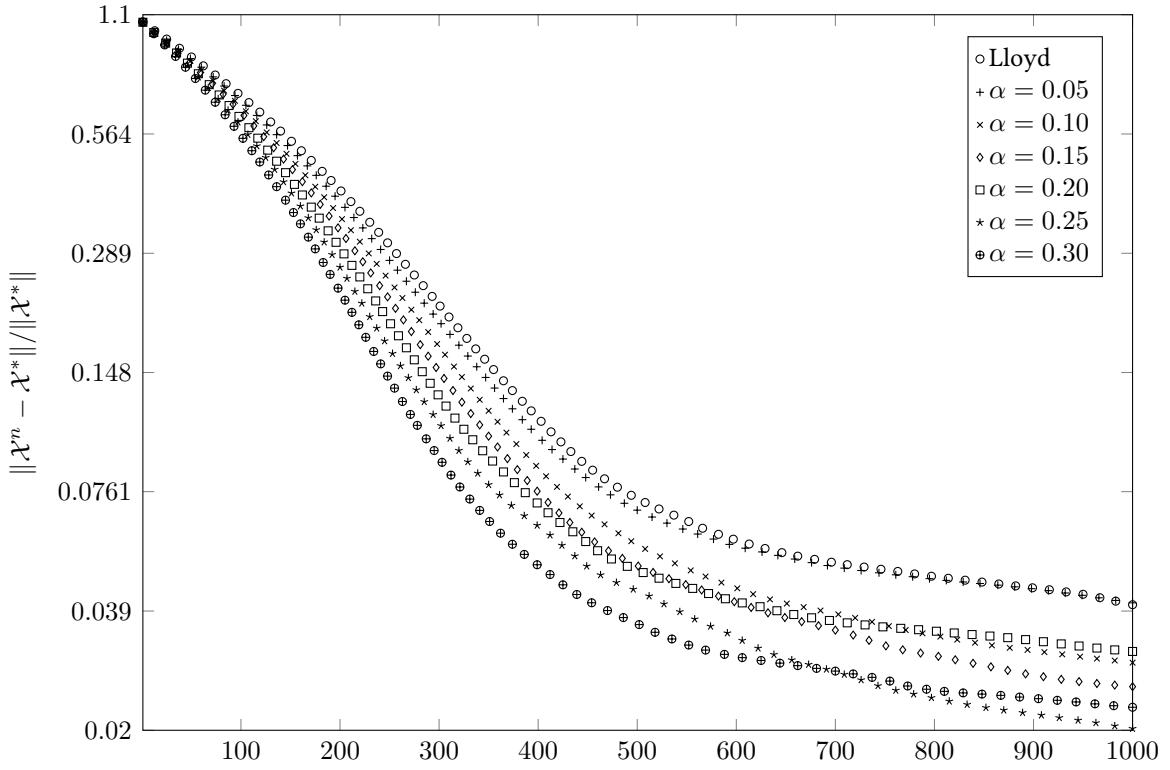
Nach einer gewissen Anzahl von Iterationen des Lloyd-Newton-Algorithmus sind nur sehr geringe Verschiebungen von Generatoren zu beobachten, weshalb danach ein α wesentlich höher als 0.3 erreicht werden könnte. Jedoch erweist sich eine algorithmische Bestimmung eines optimalen α pro Iteration als sehr rechenaufwendig, da für jede Überprüfung von α ein neues Voronoi-Diagramm erzeugt werden muss.

Bemerkung 3.5:

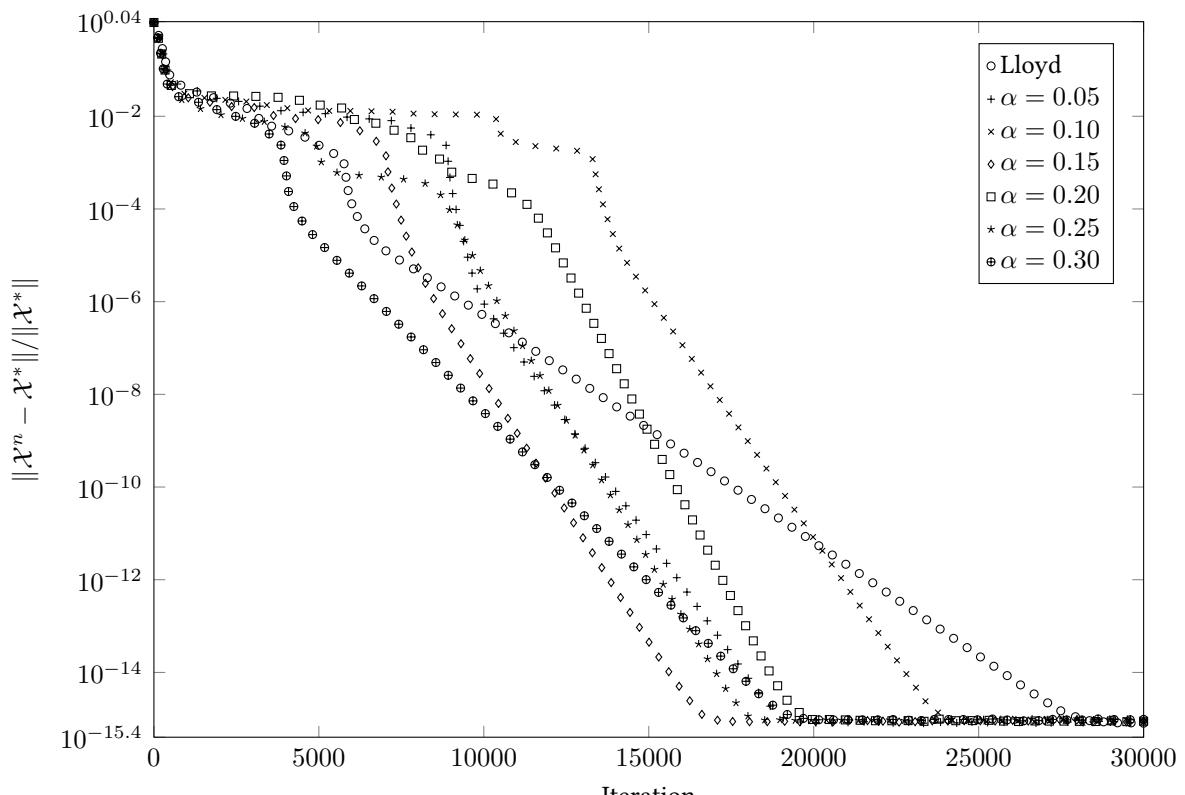
Um Robustheit des in dieser Arbeit entwickelten Lloyd-Newton-Algorithmus zu gewährleisten, empfiehlt es sich für α die Werte aus dem Intervall $(0, 0.25]$ zu wählen.

Bemerkung 3.6:

Bei einer günstigen Wahl von α kann der Lloyd-Newton-Algorithmus die doppelte Konvergenzrate gegenüber dem Lloyd-Algorithmus erreichen, d.h. für ein Gitter mit N Elementen benötigt man ca. $3N$ Iterationen des Lloyd-Newton-Algorithmus, um ein Centroidal-Voronoi-Diagramm des definierten Gebietes zu erzeugen.



(a) Erste 1000 Iterationen.



(b) Bis 30000 Iterationen.

Abbildung 3.11: Abhangigkeit der Konvergenz vom Parameter α beim Lloyd-Newton-Algorithmus. Inhomogene Zeldichte: Kantenlangenverhaltnis von 0.1 zu 1. Das Gitter mit 1519 Elementen.

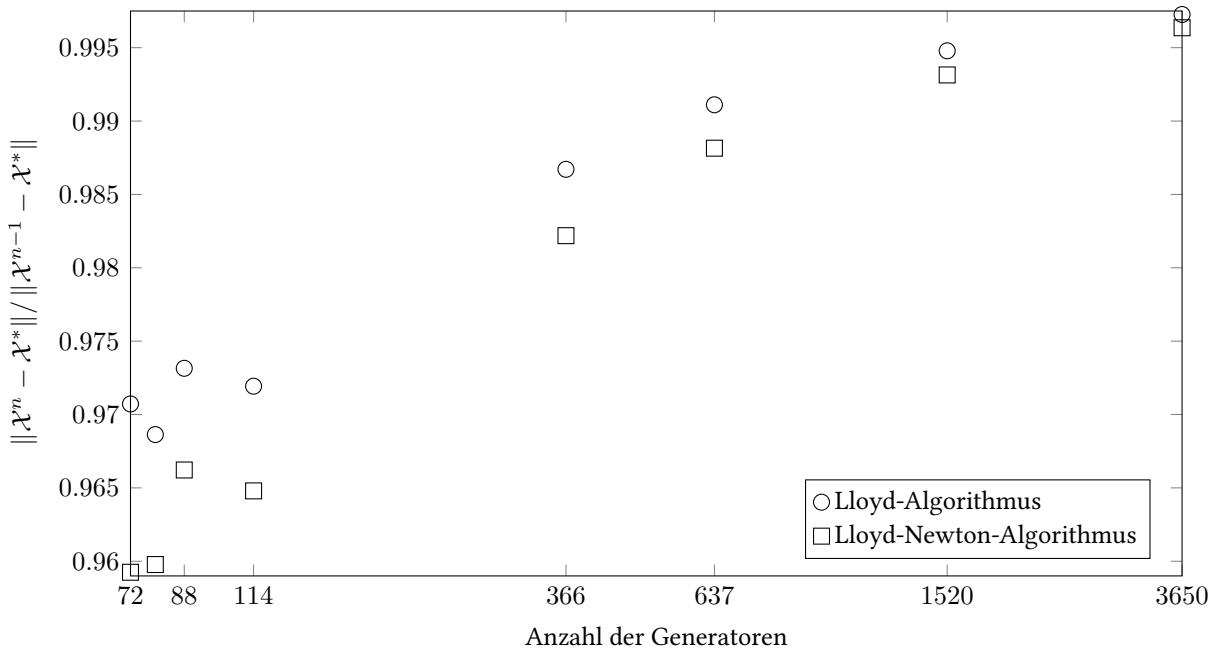


Abbildung 3.12: Durchschnittliche Konvergenzraten aus den ersten 20 Iterationen des Lloyd-Algorithmus und des Lloyd-Newton-Algorithmus bei $\alpha = 0.3$. Inhomogene Zelldichte mit Kantenlängenverhältnis von 0.5 zu 1.

3.7 Wodurch entsteht Gitterbewegung?

Wie bereits im Abschnitt 3.2 beschrieben wurde, besitzen die CV-Gitter die fundamentale Eigenschaft im globalen Sinne optimal zu sein. Denn diese entstehen durch Minimierung einer Fehlerfunktion. Wir nutzen diese globale Optimalität von CVT, um auf Verschiebungen des Gebietsrandes zu antworten. Die Gitterbewegung erhält man als Resultat der Wiederherstellung der Centroidal-Voronoi-Eigenschaft des darunterliegenden Gitters.

Im Kapitel 2 wurde angegeben, dass der Gebietsrand durch eine zusätzliche triangulierte Fläche repräsentiert wird. Wir stellen uns folgende Situation vor: Es gibt ein CV-Gitter mit N Generatoren, das durch ein Randgitter begrenzt ist. Wir verschieben dieses Randgitter und lassen die Generatoren unberührt. Der Schnitt des durch diese Generatoren erzeugten Voronoi-Diagramms mit dem verschobenen Randgitter ist keine CVT mehr, da die Schwerpunkte der Randzellen sich dadurch verschieben, die Generatoren bleiben jedoch unberührt. Mithilfe des Lloyd-Algorithmus korrigieren wird dieses gestörte Gitter und erhalten ein CV-Diagramm. Diese Korrektur führt zu Bewegung des ganzen Gitters. Da jeder Schritt des Lloyd-Algorithmus eine Voronoi-Zerlegung des zeitlich veränderlichen Gebietes zur Folge hat, ist es beinahe unmöglich mehrere Schritte des Lloyd-Algorithmus pro Simulationsschritt zu machen. In der Praxis hat es sich herausgestellt, dass auch ein einziger Schritt des Lloyd-Algorithmus die Qualität des darunterliegenden Gitters bei Randverschiebungen aufrecht erhalten kann. Eine Zusammenfassung der beschriebener Schritte liefert der Algorithmus 3.4.

Algorithmus 3.4 Ein Algorithmus zur Netzbewegung mithilfe von CVT

Sei ein CV-Gitter gegeben. Für jeden Zeitschritt einer zeitlich aufgelösten Simulation führe folgende Anweisungen aus:

while Solange der Endpunkt des Zeitintervalls nicht erreicht ist. **do**

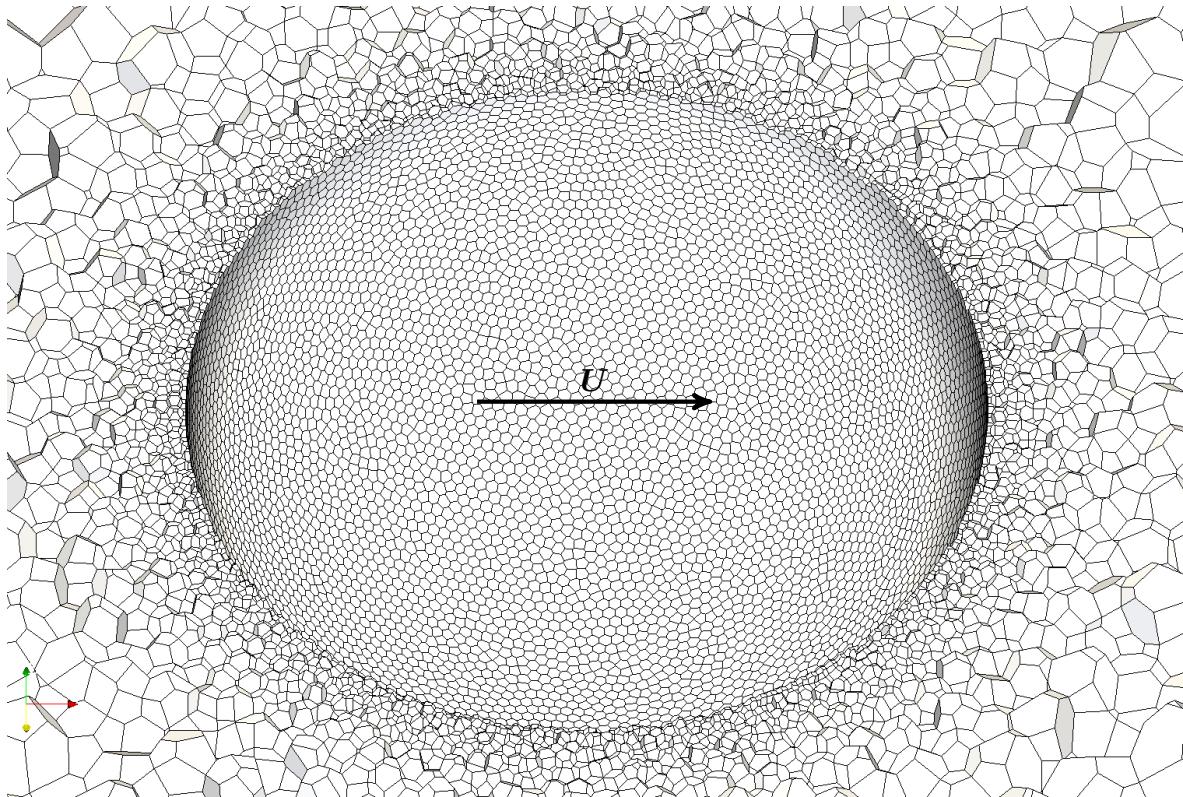
1. Berechne die Verteilung der Zelldichte.
2. Berechne neue Schwerpunkte und aktualisiere die Generatoren.
3. Berechne das Voronoi-Diagramm, das durch neue Generatoren definiert ist.
4. Verschiebe das Randgitter.
5. Verschneide das Voronoi-Diagramm mit dem Randgitter.

end while

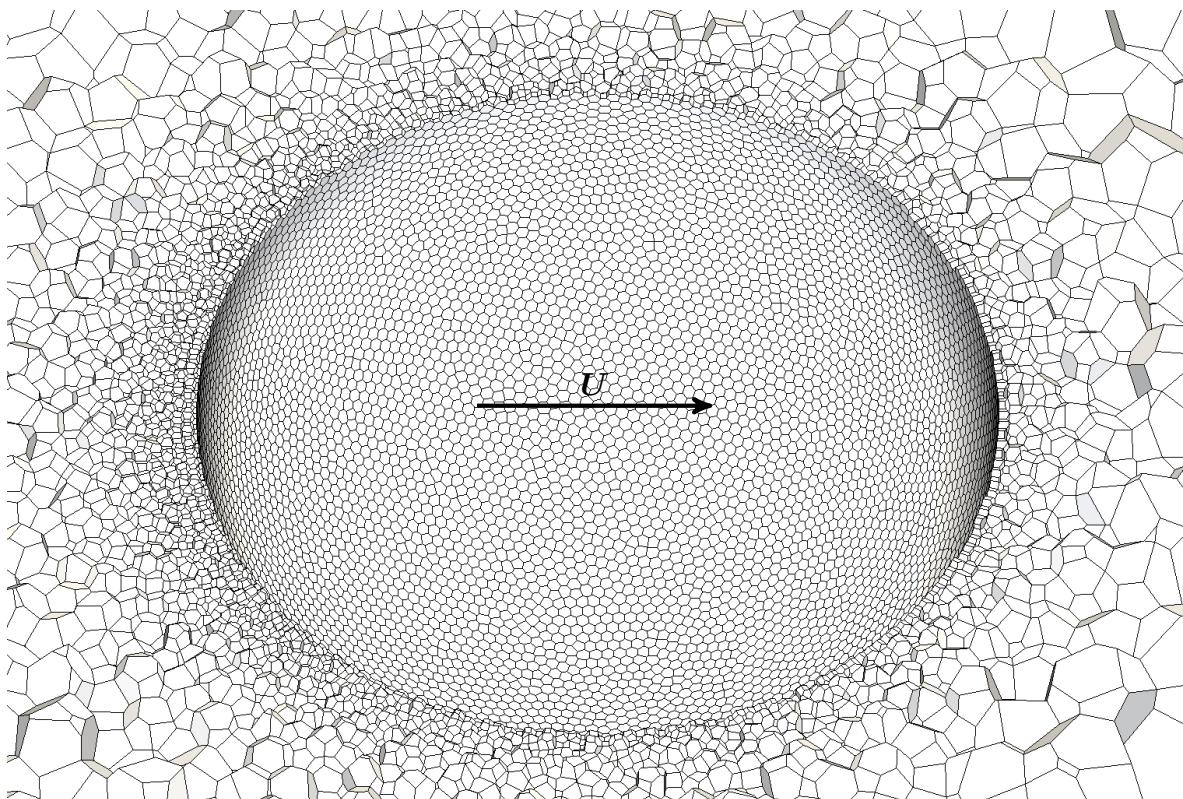
In einer unserer früheren Arbeiten [83] haben wir folgendes Phänomen beobachtet: Obwohl der Algorithmus 3.4 die Qualität des darunterliegenden Gitters aufrecht erhalten kann, reicht dieser meistens nicht aus, um die vorgegebene Verteilung der Zelldichte zu erfüllen. Bei der Simulation einer frei fallenden Stahlkugel in einem viskosen Fluid hat es sich herausgestellt, dass die Zelldichte im vorderen Bereich der Kugel im Laufe der zeitlichen Simulation abfällt. Siehe die Abbildung 3.13. Die Ursache für dieses Problem hat folgende Herkunft: Wie wir bereits im Unterabschnitt 3.6.1 festgestellt haben, benötigt man

$$\frac{10^4 \cdot 775380}{1500} \approx 5 \cdot 10^6 \quad (\text{Vergleiche die Gleichung in (3.134)}) \quad (3.135)$$

Schritte des Lloyd-Algorithmus, um ein bis auf Rundungsfehler exaktes Centroidal-Voronoi-Gitter zu erreichen. Dies bedeutet, nach jeder Verschiebung des Randgitters müssten wir $5 \cdot 10^6$ Schritte des Lloyd-Algorithmus durchführen, um eine vorgegebene Verteilung der Zelldichte zu erfüllen. Da dieses Vorgehen sehr ineffizient ist, lösen wir das beschriebene Problem auf eine alternative Weise. Wir modifizieren den Algorithmus 3.4 dadurch, dass wir zu den neu berechneten Generatoren eine zusätzliche Verschiebung addieren, die als Laplace-Glättung der am Gebietsrand vordefinierter Verschiebung bestimmt wird. In den meisten Fällen kann die Randbedingung



(a) Das Netz bei dem Zeitpunkt $t = 0$



(b) Das Netz bei dem Zeitpunkt $t = 0.3585$. Die Kameraposition wurde mit der Kugel verschoben.

Abbildung 3.13: Simulation einer frei fallenden Kugel in einem viskosen Fluid. Die Bewegungsrichtung zeigt nach rechts. Das Gesamtgitter enthält 775380 Voronoi-Zellen. Nach einer gewissen Zeit ist die Netzfeinheit am linken Rand der Kugel deutlich höher als am rechten Rand (Siehe die Abb. 3.13b).

zur Verschiebung der Generatoren gleich der Verschiebung des Randgitters gesetzt werden. Die beschriebene Verbesserung ist im Algorithmus 3.5 enthalten.

Algorithmus 3.5 Ein verbesserter Algorithmus zur Netzbewegung auf Basis von CVT

Sei ein CV-Gitter gegeben. Für jeden Zeitschritt einer zeitlich aufgelösten Simulation führe folgende Anweisungen aus:

while Solange der Endpunkt des Zeitintervalls nicht erreicht ist. **do**

1. Berechne die Verteilung der Zelldichte.
2. Berechne neue Schwerpunkte und aktualisiere die Generatoren.
3. Löse die Laplace-Gleichung $\Delta u(x) = 0$ für die Verschiebung u , wobei die Randbedingung für u durch den Anwender vorgegeben wird.
4. Für alle inneren Generatoren: Addiere die zusätzliche Verschiebung aus dem Punkt 3 zu den berechneten Generatoren aus dem Punkt 2.
5. Berechne das Voronoi-Diagramm, das durch neue Generatoren definiert ist.
6. Verschiebe das Randgitter.
7. Verschneide das Voronoi-Diagramm mit dem verschobenen Randgitter.

end while

Es muss hier noch folgendes erwähnt werden. Der Algorithmus 3.5 ist nur dann durchführbar, wenn beim Schnitt mit dem verschobenen Randgitter alle ursprünglichen Volumenelemente im neuen Gebiet bleiben. Sonst müssen die Zellen entfernt werden, was nicht das Ziel der ursprünglichen Idee zur Netzbewegung ist. Das heißt, es muss eine Art Courant-Zahl für die Geschwindigkeit der Netzränder geben. Die Courant-Zahl wird im 1-dimensionalen Fall durch

$$C_i = \frac{v_i \Delta t}{\Delta x_i} < 1, \text{ für } i = 1, 2, \dots, N \quad (3.136)$$

definiert. Dadurch ergeben sich gewisse Schranken für die Zeitschrittweite des verwendeten Diskretisierungsverfahrens. Die Bedingung in (3.136) sollte sowohl für die Fluid-Geschwindigkeit als auch für die Netz-Geschwindigkeit erfüllt sein.

Da die Ungleichung in (3.136) für jedes Element erfüllt werden muss, bedeutet dies, dass die Zeitschrittweite Δt durch die kleinste Zelle mit der größten Geschwindigkeit begrenzt ist. Meistens befindet sich die kleinste Zelle des Gitters auf dem Rand des Gebiets. Das bedeutet: Wenn man die Courant-Bedingung in (3.136) für die Fluid-Geschwindigkeit nicht verletzt, wird die Bedingung in (3.136) auch für die Netz-Geschwindigkeit erfüllt. Zu beachten ist dabei, dass bei der Simulation von Strömungen in zeitabhängigen Gebieten die Netz-Geschwindigkeit des Randes so gewählt wird, dass diese kleiner oder gleich der Fluid-Geschwindigkeit auf dem Rand ist.

Von großer Bedeutung ist noch die Tatsache, dass der Lloyd-Algorithmus bei größeren Abweichungen von dem Fixpunkt wesentlich höhere Konvergenzraten hat. Vergleiche die Bemerkung 3.2. Dies führt dazu, dass bei einer relativ großen Verschiebung des Randes ein einziger Schritt des Lloyd-Algorithmus ausreicht, um größte Störungen im CV-Gitter "wegzuglätt". Offensichtlich sind die größten Störungen durch die Randzellen gegeben. Somit dient der Lloyd-Algorithmus wie ein Glättungsfilter, der die Randverschiebungen ins Innere fortsetzt.

3.7.1 Vorbereiten des initialen Netzes zur Strömungssimulation

Für die Vorbereitung des initialen Gitters wird ein externer Vernetzer benötigt. Bei der in dieser Arbeit entwickelten Methode werden sowohl tetraedrische als auch polyedrische Gitter unterstützt. Bei transienten Strömungssimulationen ist es jedoch empfehlenswert den Algorithmus

Kapitel 3. Gitterbewegung mithilfe von CVT

3.5 mit einem CV-Netz zu starten. Denn sonst erhält man viel zu große Verschiebungen des darunterliegenden Gitters. Da der Lloyd-Algorithmus sehr niedrige Konvergenzrate besitzt (Vergleiche Bemerkung 3.1), benutzen wir für die Erzeugung von CV-Gittern den im Abschnitt 3.4 beschriebenen alternativen Lloyd-Newton-Algorithmus. Angenommen, es liegt ein Gitter mit einer Million Volumenelemente vor, bei dem die gewünschte ZelldichteVerteilung bereits erfüllt ist. Dann reichen von 100 bis 200 Schritte des Lloyd-Newton-Algorithmus aus, um den relativen Fehler $\|\mathcal{X}^n - \mathcal{X}^*\|/\|\mathcal{X}^*\|$ unter den Wert 0.05 zu bringen. In den weiteren Iterationen kann man visuell nur geringfügige Verschiebungen der Netznoten feststellen. Auch die in dem Unterabschnitt 4.6.4 definierten Konvergenzkriterien erfahren nach ca. 200 Iterationen dieses Algorithmus keine großen Änderungen.

Falls die gewünschte ZelldichteVerteilung nicht erfüllt ist, können für ein Netz mit einer Million Elemente laut Bemerkung 3.6 im ungünstigsten Fall bis zu 3 Millionen Iterationen des Lloyd-Newton-Algorithmus benötigt werden, um diese ZelldichteVerteilung zu erreichen. Hierdurch erkennt man, dass der entwickelte Lloyd-Newton-Algorithmus nicht die quadratische Konvergenzordnung des klassischen Newton-Verfahrens besitzt. Die Ursache liegt dabei in der Einschränkung, dass der neue Generator innerhalb seiner alten Voronoi-Region liegen soll.

Im Hinblick auf die beschriebene Problematik hat sich im Rahmen dieser Arbeit ein Ansatz herausgebildet, bei dem das initiale Gitter durch drei Stufen erzeugt wird. Die in diesen Stufen generierte Gitter werden durch G_1 , G_2 und G_3 bezeichnet. G_3 entspricht dem initialen Gitter für die transiente Strömungssimulation. Der entwickelte Ansatz ist im Algorithmus 3.6 zusammengefasst.

Algorithmus 3.6 Vorbereiten des initialen Netzes zur Strömungssimulation

1. Zur Erzeugung des initialen Gitters wird eine Vernetzung-Software benötigt. Im Rahmen dieser Arbeit wurde zu diesem Zweck der kommerzielle Präprozessor ANSA verwendet.
2. Beim Vernetzen des zu simulierten Modells sollte der Anwender die Elementgröße des Flächenmodells so wählen, damit diese der gewünschten ZelldichteVerteilung am Gebietsrand entspricht.
3. Die Anzahl der Volumenelemente im initialen Gitter soll so sein, dass die gewünschte ZelldichteVerteilung realisiert werden kann. Zuerst sollte ein vorläufiges Gitter G_1 mit \tilde{N} Elementen erzeugt werden. Damit kann die optimale Anzahl der Volumenelemente im initialen Netz durch

$$N = \int_{\Omega} \left(\frac{\xi}{\mathcal{L}(\mathbf{x})} \right)^3 d\mathbf{x} \approx \sum_{i=1}^{\tilde{N}} \left(\frac{\xi}{\mathcal{L}_i} \right)^3 |\Omega_i| = \xi^3 \sum_{i=1}^{\tilde{N}} \frac{|\Omega_i|}{(\mathcal{L}_i)^3} \quad (3.137)$$

bestimmt werden, wobei

- Ω - das Rechengebiet,
- $\mathcal{L}(\mathbf{x})$ - die vorgegebene Kantenlänge,
- $\xi = \frac{113}{250}$ - das experimentell bestimmte Verhältnis der mittleren Kantenlänge des durchschnittlichen Polyeders zu dem dritten Wurzel aus dem Volumen dieses Polyeders,
- \mathcal{L}_i - die Kantenlänge am i -ten Element,
- $|\Omega_i|$ - der Volumeninhalt des i -ten Elements

sind. Durch die Kenntnis von N wird G_2 generiert.

4. Aus G_2 wird durch ca. 200 Schritte Lloyd-Newton-Algorithmus G_3 erzeugt.
-

3.7.2 Optimierung der Randzellen mithilfe von CCVT

In diesem Unterabschnitt wird beschrieben, wie die Flächenelemente auf dem Rand optimiert werden können. In [59, 63] wurde die sogenannte CCVT (Engl. Constrained Centroidal Voronoi Tessellation) beschrieben, bei der die Generatoren der Randelemente zwangsläufig auf dem Randgitter liegen. Im Sinne einer Optimalitätsbedingung werden dabei die gewichteten Schwerpunkte dieser Elemente auf das Randgitter projiziert. Im Allgemeinen ist die Projektion eines Punktes auf eine nicht ebene parametrische Fläche im \mathbb{R}^3 nicht eindeutig. So ist beispielsweise die Projektion des Zentrums einer Kugel auf deren Oberfläche die Menge der Punkte dieser Oberfläche. In dieser Arbeit wurde eine Art diskrete „Projektion“ entwickelt, die immer durchführbar ist. Es handelt sich dabei um keine wirkliche Projektion, sondern um eine Prozedur bei der jedoch die Generatoren der Randzellen auf dem Randgitter landen.

Die im weiteren diskutierten Randflächen, Randkanten oder gemeinsame Eckpunkte beziehen sich auf das alte Gitter. Die Verschiebung des Randgitters kann dabei berücksichtigt werden, indem man diese auf die alten Randknoten des Volumengitters überträgt. Abhängig davon wie viele Randflächen eine Randzelle besitzt, ist einer der nachfolgenden Schritte anzuwenden:

- Genau eine Randfläche: Wir setzen den Generator auf den gewichteten Schwerpunkt der entsprechenden Randfläche. Die Krümmung der Randfläche wird dabei vernachlässigt. Der gewichtete Schwerpunkt für die Randfläche \mathcal{S}_i berechnet sich durch

$$\frac{\int_{\mathcal{S}_i} \tilde{\rho}(\mathbf{x}) \mathbf{x} d\mathbf{x}}{\int_{\mathcal{S}_i} \tilde{\rho}(\mathbf{x}) d\mathbf{x}}, \quad (3.138)$$

wobei $\tilde{\rho}(\mathbf{x})$ die flächenbezogene Zelldichte ist. Analog wie im Unterabschnitt 3.2.3 kann man zeigen, dass

$$\tilde{\rho}(\mathbf{x}) = \frac{1}{(\mathcal{L}(\mathbf{x}))^4}. \quad (3.139)$$

$\mathcal{L}(\mathbf{x})$ bezeichnet hierbei die Kantenlänge. Es muss wiederum folgende Näherung gelten:

$$\int_{\mathcal{S}_i} \tilde{\rho}(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^2 d\mathbf{x} \approx \beta. \quad (3.140)$$

Näherungsweise gilt

$$\int_{\mathcal{S}_i} \tilde{\rho}(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^2 d\mathbf{x} \approx \tilde{\rho}_i r_i^4 \approx \beta \Rightarrow \tilde{\rho}_i \approx \frac{1}{r_i^4}, \quad (3.141)$$

wobei $\tilde{\rho}_i$ die zu dem Generator \mathbf{x}_i zugehörige Zelldichte und r_i der größte Radius des Flächenelements \mathcal{S}_i sind. Die Integrale in (3.138) müssen, wie in dem Unterabschnitt 3.5.2 beschrieben, durch mindestens die Quadraturformel des zweiten Genauigkeitsgrades approximiert werden.

- Genau zwei Randflächen: Wir suchen nach einer gemeinsamen Kante in den beiden Randflächen. Wenn die Suche erfolgreich ist, setzen wir den Generator auf den gewichteten Schwerpunkt dieser Kante:

$$\frac{\int_{\mathcal{E}_i} \hat{\rho}(\mathbf{x}) \mathbf{x} d\mathbf{x}}{\int_{\mathcal{E}_i} \hat{\rho}(\mathbf{x}) d\mathbf{x}}, \quad (3.142)$$

wobei \mathcal{E}_i die Punktmenge der gemeinsamen Kante bezeichnet. Analog zum Punkt 1. wird hier die Zelldichte $\hat{\rho}(\mathbf{x})$ durch

$$\hat{\rho}(\mathbf{x}) = \frac{1}{(\mathcal{L}(\mathbf{x}))^3} \quad (3.143)$$

definiert. Die Integrale in (3.142) können mit Hilfe der Simpsonschen Formel approximiert werden. Für eine stetige Funktion $f : \mathbb{R} \mapsto \mathbb{R}$, $y \mapsto f(y)$ hat diese Formel die Gestalt

$$\int_a^b f(y) dy \approx \frac{b-a}{6} \left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right). \quad (3.144)$$

Die Quadraturformel in (3.144) hat den Genauigkeitsgrad 3. Ein großer Vorteil dieser Formel besteht darin, dass diese den Anfangs- und Endpunkt enthält, was die Anzahl der Auswertungen des Typs $\frac{1}{(\mathcal{L}(\mathbf{x}))^3}$ reduziert lässt. Die Trapezregel hat lediglich den Genauigkeitsgrad 1 und ist deshalb für die Approximation der Integrale in (3.142) nicht geeignet.

Wir zeigen, wie das Integral im Nenner in (3.142) mit Hilfe der Simpsonschen Formel ausgewertet wird:

$$\int_{\mathcal{E}_i} \frac{1}{(\mathcal{L}(\mathbf{x}))^3} d\mathbf{x} \approx \frac{\|\mathcal{E}_i^b - \mathcal{E}_i^a\|}{6} \left((\mathcal{L}(a))^{-3} + 4 \left(\mathcal{L}\left(\frac{a+b}{2}\right) \right)^{-3} + (\mathcal{L}(b))^{-3} \right), \quad (3.145)$$

wobei $\mathcal{E}_i^a, \mathcal{E}_i^b$ der Anfangs- und der Endpunkt von \mathcal{E}_i sind.

Im Falle, wenn beide Randflächen keine gemeinsame Kante haben, führen wir den Schritt 1 für die Randfläche, deren gewichteter Schwerpunkt näher zu dem gewichteten Schwerpunkt des darunterliegenden Volumenelements liegt.

3. Drei Randflächen oder mehr:

- Suche zuerst nach drei Randflächen, die einen gemeinsamen Eckpunkt haben. Setze den Generator auf diesen Eckpunkt. Wenn mehrere gemeinsame Punkte gefunden werden, wähle den Eckpunkt, der näher zum gewichteten Schwerpunkt des Volumenelements liegt.
- Wenn keine drei Randflächen mit einem gemeinsamen Eckpunkt existieren, suche nach zwei Randflächen, die eine gemeinsame Kante enthalten. Bestimme wie im Schritt 2 den gewichteten Schwerpunkt und erkläre diesen zum Generator. Wenn mehrere gemeinsame Kanten existieren, wähle die Kante, deren gewichtete Schwerpunkt näher zum gewichteten Schwerpunkt des Volumenelements liegt.
- Wenn es keine zwei Randflächen gibt, die eine gemeinsame Kante enthalten, fahre mit Schritt 1 fort. Wähle dabei die Randfläche, deren gewichtete Schwerpunkt näher zum gewichteten Schwerpunkt des Volumenelements liegt.

3.7.3 Randbedingungen zur Verschiebung der Generatoren

Der entwickelte Algorithmus zur Gitterbewegung bietet drei Randbedingungen zur Verschiebung der Randelemente an:

- **freeSurface**: Dabei werden alle Generatoren durch den Lloyd-Algorithmus bestimmt, d.h. diese werden auf die Stellen der gewichteten Schwerpunkte platziert. Das dadurch erzeugte Voronoi-Diagramm wird danach mit dem Randgitter verschnitten. Dabei können sich die Randelemente von dem Randgitter loslösen, und sich ins Innere fortbewegen. Ein großer Vorteil dieser Randbedingung besteht darin, dass eine vorgegebene Verteilung der Zeldichte bestmöglich erfüllt wird. Während des Loslösens von dem Randgitter können jedoch sehr kleine Randflächenelemente entstehen, wodurch sich die Qualität des darunterliegenden Gitters beträchtlich verschlechtern kann. Außerdem können dabei während der Simulation im Bereich konkaver Ränder die im Abschnitt 2.5 beschriebenen Probleme auftreten, was zum Abbruch der Simulation führen kann. Es empfiehlt sich daher diese Randbedingung nur für konvexe Randteile zu definieren.
- **projectOnToSurface**: Projektion auf das Randgitter mithilfe der im Unterabschnitt 3.7.2 beschriebenen Methode. Dabei werden die Datenstrukturen des Gitters vom alten Zeitschritt verwendet und die Verschiebung des Randgitters nicht berücksichtigt. Damit kann sich die Position der Generatoren im Bezug auf das Randgitter verändern. Hierdurch werden die Randflächen der Volumenelemente während einer transienten Simulation optimiert.

- `moveWithSurface`: In einer vorbereitenden Phase werden die Generatoren der Randzellen auf das Randgitter projiziert, wodurch sowohl der Index des entsprechenden Dreiecks als auch die relativen Koordinaten bezüglich der dieses Dreieck aufspannenden Vektoren gewonnen werden. Nach der Verschiebung des Randgitters werden die neuen Positionen der Generatoren der Randelemente mithilfe dieser gespeicherten Daten bestimmt. Damit werden die Randknoten des Voronoi-Gitters an die bestimmten relativen Koordinaten des Randgitters gebunden.

Diese Randbedingung hat gewisse Vorteile bei solchen Bewegungsarten wie Verschiebung oder Rotation, dessen Oberbegriff Verrückung heißt. Bei der Verrückung des Randgitters werden die Randknoten genauso transformiert als ob diese Transformation direkt auf diese Knoten angewandt würde. Dadurch wird das Volumen eines Rechengebietes bei der Verrückung eines in diesem Rechengebiet eingeschlossenen Körpers erhalten. Dies wird anhand des oben beschriebenen Beispiels „Frei fallende Stahlkugel in einem viskosen Fluid“ deutlich. Wir bezeichnen das Volumen des zur Stahlkugel gehörenden Gitters mit V_0 , das Volumen des Zylinders mit V_1 und das Volumen des Rechengebietes mit $V_2 = V_1 - V_0$. Bei Verrückung der zur der Stahlkugel gehörenden Netzknoten bleibt sowohl V_0 als auch V_1 konstant, damit bleibt auch $V_2 = V_1 - V_0$ konstant.

Dieser Umstand hat eine große Bedeutung bei der Simulationen von Strömungen inkompressibler Fluide, da diese Simulationen im numerischen Sinne empfindlich gegenüber Störungen in der Volumenerhaltung sind. Bei kompressiblen Fluiden können zwar solche Störungen auch zu Verfälschung der physikalischen Phänomene führen, die numerische Stabilität bleibt davon meist unberührt. Denn die Verletzung der Volumenerhaltung führt in diesem Fall lediglich zur Änderung der Dichte.

3.8 Wahl der Quadraturformeln zur Auswertung der Volumen- bzw. der Flächenintegrale

In dem Abschnitt 3.5 haben wir folgende Feststellung gemacht: Die Quadraturformel des fünften Genauigkeitsgrades liefert den kleinsten Fehler bei der Approximation des Volumenintegrals $\int_{\Omega_i} \rho(\mathbf{x}) \mathbf{x} d\mathbf{x}$ (vergleiche die Tabelle 3.2 und die Abbildung 3.4). Diese Feststellung kann im Allgemeinen nicht auf alle Geometrien bzw. Zelldichten übertragen werden.

In dieser Arbeit verwenden wir für die Approximation der Volumen- bzw. der Flächenintegrale die Quadraturformeln des zweiten bzw. des dritten Genauigkeitsgrades. Diese sind in den Tabellen 3.1 und 3.3 dargestellt. Das sind die notwendigen Genauigkeitsgrade, um die Funktionen $\rho(\mathbf{x}) \mathbf{x}$ bzw. $\rho(\mathbf{x}) \mathbf{x} \mathbf{x}^T$ durch ein Polynom des zweiten bzw. des dritten Grades anzunähern (vergleiche die Bemerkung 3.1). Für diese Entscheidung gibt es folgende Gründe:

- In der Abbildung 3.4 ist es ersichtlich, dass die Endzustände der Gitter, die den Quadraturformeln des zweiten und höherer Genauigkeitsgrade entsprechen, sich zwar beträchtlich unterscheiden, ihre Qualität weist jedoch keine großen Unterschiede auf.
- Bei der Gitterbewegung hat eine mögliche Reduktion des Rechenaufwands eine höhere Priorität als eine exakte Auswertung der Volumenintegrale. Diese Auswertung nimmt ohnehin einen beträchtlichen Anteil der Rechenzeit auf. Bei einem Gitter mit 200 Tausend Elementen beträgt das Verhältnis der Rechenzeit für die Auswertung der Volumenintegrale zu der Rechenzeit für die Konstruktion des entsprechenden Voronoi-Diagramms 1/4.

3.8. Wahl der Quadraturformeln zur Auswertung der Volumen- bzw. der Flächenintegrale

- Eine hohe Genauigkeit der Approximation der Volumenintegrale ist auch nicht notwendig, da aufgrund der langsamen Konvergenz des Lloyd-Algorithmus nur Annäherungen an die CV-Gitter erreicht werden können.

4 Nutzung des Verfahrens mit der FVM

Die Finite-Volumen-Methode findet ihren Einsatz bei Diskretisierung der sogenannten Grundgleichungen der Strömungsmechanik. Die Strömungsmechanik beschreibt das Verhalten der Fluide unter Einwirkung von Kräften. Bei komplizierten Geometrien ist man meistens nicht in der Lage, die Prozesse auf molekularer Ebene zu beschreiben. Anstatt dessen werden diese Prozesse auf makroskopischen Skalen modelliert, wobei zwischen den Teilchen nicht mehr unterschieden wird. Dies ermöglicht die Beobachtung eines infinitesimalen (*unendlich kleinen*) Kontrollvolumens. Bezüglich dieses Kontrollvolumens werden die Erhaltungssätze der Strömungsmechanik formuliert. Siehe [37, 50].

In diesem Kapitel werden die Grundgleichungen der Strömungsmechanik für bewegliche Gitter formuliert. Danach wird eine Methode vorgestellt, bei der die Netzflüsse an den Flächenelementen so rekonstruiert werden, dass der sogenannte geometrische Erhaltungssatz erfüllt ist. Die Abschnitte 4.1, 4.2 und 4.4 wurden mithilfe der Literaturquelle [50] verfasst.

4.1 Lagrangesche und Eulersche Darstellung

Wir betrachten ein Gebiet $\Omega_0 \subset \mathbb{R}^3$, welches der Ausgangskonfiguration (zum Zeitpunkt $t = 0$) des Kontinuums entspricht. $\mathbf{X} \in \Omega_0$ steht für die einzelnen Teilchen des Kontinuums. \mathbf{X} werden Lagrange-Koordinaten oder auch materielle Koordinaten genannt. Angenommen, die Bewegung eines Teilchens im Raum lässt sich durch die \mathcal{C}^1 invertierbare Abbildung

$$\Phi : \Omega_0 \times T \mapsto \Omega_t \subset \mathbb{R}^3 \quad (4.1)$$

beschreiben. \mathcal{C}^1 invertierbar bedeutet hierbei, dass Φ und die Umkehrabbildung Φ^{-1} einmal stetig differenzierbar sind und die Jacobi-Matrix

$$D\Phi = \frac{\partial \Phi}{\partial \mathbf{X}} \quad (4.2)$$

ist für jedes $\mathbf{X} \in \Omega_0$ und $t \geq 0$ invertierbar. Ω_t bezeichnet das Gebiet, welches der Momentankonfiguration (zum Zeitpunkt t) entspricht. $T = [t_0, t_{end}] \subset \mathbb{R}$ ist das betrachtete Zeitintervall. Sei

$$\mathbf{x} = \Phi(\mathbf{X}, t) \quad (4.3)$$

die Position des Teilchens mit Koordinaten \mathbf{X} zum Zeitpunkt t . Die Größe \mathbf{x} wird als eulersche bzw. räumliche Koordinate bezeichnet. Abhängig von den verwendeten Koordinaten gibt es zwei grundlegende Beschreibungen einer physikalischen Größe und ihres zeitlichen Verhaltens:

- Die Lagrangesche Darstellung, bei der die unabhängigen Variablen die Koordinaten $\mathbf{X} \in \Omega_0$ und die Zeit t sind. Hier werden die Teilchen in ihrem zeitlichen Verlauf verfolgt;
- Die Eulersche Darstellung, die als unabhängige Variablen die Koordinaten $\mathbf{x} \in \Omega_t$ und die Zeit t benutzt. Hier wird ein fester Punkt im Raum beobachtet, an dem sich zu verschiedenen Zeiten auch verschiedene Teilchen des Kontinuums befinden.

Angenommen f^L ist die Lagrange-Darstellung einer physikalischen Größe und f^E ihre Euler-Darstellung, dann ist der Übergang von einer Darstellung zur anderen durch

$$f^L(\mathbf{X}, t) = f^E(\Phi(\mathbf{X}, t), t) = f^E \circ \Phi \quad (4.4)$$

gegeben. Die Geschwindigkeit eines Teilchens mit Koordinaten $\mathbf{X} \in \Omega_0$ lässt sich als die Ableitung der Verschiebung nach der Zeit bei fixierten \mathbf{X} -Koordinaten errechnen. Man erhält damit

$$\mathbf{U}(\mathbf{X}, t) = \frac{\partial(\Phi(\mathbf{X}, t) - \Phi(\mathbf{X}, 0))}{\partial t} = \frac{\partial \Phi(\mathbf{X}, t)}{\partial t}. \quad (4.5)$$

Die Ableitung nach der Zeit mit fixierten Lagrange-Koordinaten heißt materielle oder substantielle Ableitung und wird mit dem Symbol $\frac{D}{Dt}$ bezeichnet. Die materielle Ableitung einer stetig differenzierbaren Funktion $f(\mathbf{x}, t)$ in Eulerscher Darstellung kann mithilfe der Kettenregel ermittelt werden:

$$\frac{Df(\Phi(\mathbf{X}, t), t)}{Dt} = \left(\frac{\partial f}{\partial \mathbf{x}} \quad \frac{\partial f}{\partial t} \right) \begin{pmatrix} \frac{\partial \Phi(\mathbf{X}, t)}{\partial \mathbf{x}} \\ \frac{\partial \Phi(\mathbf{X}, t)}{\partial t} \end{pmatrix} = \nabla f \cdot \mathbf{U} + \frac{\partial f}{\partial t}. \quad (4.6)$$

In der Gleichung (4.6) ist es ersichtlich, dass die materielle Ableitung einer Funktion sich aus ihrer zeitlichen Änderung an einem festen Ort und aus ihrem konvektiven Anteil zusammensetzt. Die Konvektion wird durch die Bewegung des Kontinuums hervorgerufen.

Eine weitere für die Herleitung der Erhaltungssätze bedeutende Gleichung ist der Reynolds'sche Transportsatz

$$\frac{D}{Dt} \int_{V_t} f d\mathbf{x} = \int_{V_t} \left[\frac{\partial f}{\partial t} + \nabla \cdot (f \mathbf{U}) \right] d\mathbf{x}. \quad (4.7)$$

Wie in dem Skript [81] gezeigt ist, resultiert der Reynolds'sche Transportsatz aus (4.6) und dem Transformationssatz.

4.2 Erhaltungssätze der Strömungsmechanik

4.2.1 Massenerhaltung

Seien $\rho(\mathbf{x}, t)$ die Massendichte eines Kontinuums, dann ist deren Masse durch ein Integral von $\rho(\mathbf{x}, t)$ über sein Volumen gegeben. Für das Teilvolumen V_t erhält man die Masse durch

$$m(V_t, t) = \int_{V_t} \rho(\mathbf{x}, t) d\mathbf{x}. \quad (4.8)$$

Da V_t ein zeitlich veränderliches Gebiet ist, hat die Erhaltung der Masse in diesem Kontext folgende Bedeutung. Ein Teilchen kann sich zu verschiedenen Zeitpunkten an abweichenden Orten aufhalten, seine Masse bleibt jedoch konstant. D.h. die materielle Ableitung der Masse muss verschwinden, und damit gilt

$$\frac{D}{Dt} m(V_t, t) = \frac{D}{Dt} \int_{V_t} \rho(\mathbf{x}, t) d\mathbf{x} = 0. \quad (4.9)$$

Mit Hilfe des Reynolds'schen Transportsatzes (4.7) erhalten wir aus (4.9)

$$\int_{V_t} \left[\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{U}) \right] d\mathbf{x} = 0. \quad (4.10)$$

4.2.2 Impulserhaltung

Der Satz der Impulserhaltung basiert auf dem zweiten Newtonschen Gesetz, das einen Zusammenhang zwischen den auf einen Körper wirkenden Kräften und seiner Beschleunigung herstellt. Der Impuls eines mit der Zeit deformierten Gebietes V_t errechnet sich durch

$$\mathbf{I}(t) = \int_{V_t} \rho(\mathbf{x}, t) \mathbf{U}(\mathbf{x}, t) d\mathbf{x}. \quad (4.11)$$

Das zweite Newtonsche Gesetz sagt aus, dass die zeitliche Änderung des Impulses gleich der Summe der wirkenden Kräfte ist. Die zeitliche Änderung entspricht der materiellen Ableitung. Sei $\mathbf{F}(t)$ die Summe aller angreifenden Kräfte, dann gilt

$$\frac{D}{Dt} \int_{V_t} \rho(\mathbf{x}, t) \mathbf{U}(\mathbf{x}, t) d\mathbf{x} = \mathbf{F}(t). \quad (4.12)$$

Durch zeilenweises Anwenden des Reynolds'schen Transportsatzes auf die linke Seite von (4.12) erhält man

$$\int_{V_t} \left[\frac{\partial}{\partial t} (\rho U^i) + \nabla \cdot (\rho U^i \mathbf{U}) \right] d\mathbf{x} = F^i(\mathbf{x}, t), \quad (4.13)$$

wobei der Index rechts oben der entsprechenden Koordinatenrichtung entspricht. Mit einfachen Umformungen erhält man in vektorieller Darstellung

$$\sum_{i=1}^3 \nabla \cdot (\rho U^i \mathbf{U}) e^i = (\rho \mathbf{U} \mathbf{U}^T) \cdot \nabla. \quad (4.14)$$

Obwohl die Schreibweise in (4.14) in der Literatur selten vorkommt, ist diese im algebraischen Sinne korrekt. Die übliche Schreibweise $\nabla \cdot (\rho \mathbf{U} \mathbf{U})$ ist hingegen nicht eindeutig, da hierbei nicht deutlich ist, ob ($\nabla \cdot$) zeilenweise oder spaltenweise durchgeführt wird. Das Produkt $\mathbf{U} \mathbf{U}$ ist auch etwas irreführend. Aus diesen Gründen benutzen wir im weiteren die Schreibweise $(\rho \mathbf{U} \mathbf{U}^T) \cdot \nabla$. Damit können wir (4.13) durch

$$\int_{V_t} \left[\frac{\partial}{\partial t} (\rho \mathbf{U}) + (\rho \mathbf{U} \mathbf{U}^T) \cdot \nabla \right] d\mathbf{x} = \mathbf{F}(t) \quad (4.15)$$

schreiben.

Die angreifenden Kräfte können in zwei Gruppen eingeteilt werden: Die Flächenkräfte $\boldsymbol{\sigma} \mathbf{n}$ und die Volumenkräfte $\rho \Lambda$, wobei $\boldsymbol{\sigma} \mathbf{n}$ ein Maß für Kraft pro Flächeneinheit und Λ für Kraft pro Volumeneinheit sind. \mathbf{n} bezeichnet die Flächennormale. Die Gesamtkraft $\mathbf{F}(t)$ ist durch

$$\mathbf{F}(t) = \int_{V_t} \rho(\mathbf{x}, t) \Lambda(\mathbf{x}, t) d\mathbf{x} + \int_{\partial V_t} \boldsymbol{\sigma}(\mathbf{x}, t) \mathbf{n} d\mathbf{x} \quad (4.16)$$

gegeben. Dabei ist ∂V_t der Rand des Gebietes V_t . Ein typisches Beispiel für Λ ist die Gravitation. Der Tensor $\boldsymbol{\sigma}$ ist der Cauchyscher Spannungstensor. In dem ersten Ansatz wird $\boldsymbol{\sigma}$ durch

$$\boldsymbol{\sigma} = -P \mathbb{1} + \boldsymbol{\tau} \quad (4.17)$$

modelliert, wobei P der Druck und $\boldsymbol{\tau}$ der sogenannte Zähigkeitstensor sind. Für newtonsche Fluide gilt

$$\boldsymbol{\tau} = \lambda (\nabla \cdot \mathbf{U}) \mathbb{1} + \mu (\nabla \mathbf{U} + (\nabla \mathbf{U})^T), \quad (4.18)$$

wobei μ die dynamische Viskosität und λ die erste Lamé-Konstante sind. Der Gradient-Operator ∇ eines Vektors \mathbf{v} ist als Jacobi-Matrix von \mathbf{v} zu verstehen. D.h. es gilt $\nabla \mathbf{v} = \frac{\partial \mathbf{v}(\mathbf{x})}{\partial \mathbf{x}}$. In Anlehnung an [24] kann λ über die experimentell gut validierte Stokes-Relation $\lambda = -\frac{2}{3}\mu$ bestimmt werden. Mit dem Gausschen Satz erhalten wir aus (4.16)

$$\mathbf{F}(t) = \int_{V_t} [\rho(\mathbf{x}, t)\Lambda(\mathbf{x}, t) + \boldsymbol{\sigma}(\mathbf{x}, t) \cdot \nabla] d\mathbf{x}, \quad (4.19)$$

wobei $\boldsymbol{\sigma}(\mathbf{x}, t) \cdot \nabla$ für die Divergenz von $\boldsymbol{\sigma}(\mathbf{x}, t)$ steht. Ein Einsetzen von (4.18) in (4.19) liefert

$$\mathbf{F}(t) = \int_{V_t} \left[\rho(\mathbf{x}, t)\Lambda(\mathbf{x}, t) - \nabla(P + \frac{2}{3}\mu\nabla \cdot \mathbf{U}) + (\mu(\nabla \mathbf{U} + (\nabla \mathbf{U})^T)) \cdot \nabla \right] d\mathbf{x}. \quad (4.20)$$

In dieser Arbeit verzichten wir auf die Beschreibung der turbulenten Phänomene und verweisen auf die Literatur [23]. Nach einer algebraischen Umformung nimmt (4.20) damit die Form

$$\mathbf{F}(t) = \int_{V_t} \left[\rho(\mathbf{x}, t)\Lambda(\mathbf{x}, t) - \nabla P + \frac{1}{3}\mu\nabla \nabla \cdot \mathbf{U} + \mu\Delta\mathbf{U} \right] d\mathbf{x} \quad (4.21)$$

an, wobei Δ der zeilenweise Laplace-Operator ist.

4.3 Navier-Stokes-Gleichungen

Die durch die Massenerhaltung und die Impulserhaltung entstehende Gleichungen fassen wir zusammen. Da V_t ein beliebig gewähltes Gebiet ist, können wir auf den Integrand verzichten. Die Kontinuitätsgleichung und die Impulsgleichung stellen ein nicht geschlossenes Gleichungssystem dar, weil es nur 4 Gleichungen für 5 Unbekannte (U_x, U_y, U_z, P, ρ) gibt. Die thermische Energiegleichung und die für das betrachtete Fluid geltende thermodynamische Zustandsgleichung vervollständigen dieses System, wodurch dieses Lösungen für Strömungsprobleme besitzt, wenn Anfangs- und Randbedingungen vorliegen. In [37] ist eine Energiegleichung in kompakter Form angegeben. Mit dieser Gleichung erhalten wir

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{U}) = 0 \quad (4.22)$$

$$\frac{\partial}{\partial t}(\rho \mathbf{U}) + (\rho \mathbf{U} \mathbf{U}^T) \cdot \nabla - \rho \Lambda - \boldsymbol{\sigma} \cdot \nabla = 0 \quad (4.23)$$

$$\frac{\partial}{\partial t}(\rho E) + \nabla \cdot (\rho E \mathbf{U}) - \rho \mathbf{g} \cdot \mathbf{U} - \nabla \cdot (\boldsymbol{\sigma} \cdot \mathbf{U}) + \nabla \cdot \mathbf{q} - \rho Q = 0, \quad (4.24)$$

wobei $\boldsymbol{\sigma}$ der Spannungstensor, E die thermische Energie, \mathbf{q} der Wärmefluss über ∂V und Q die Wärmequelle sind. Für die Herleitung der Energiegleichung siehe [24]. Für die Newtonischen Fluide gilt

$$\boldsymbol{\sigma} = - \left(P + \frac{2}{3}\mu(\nabla \cdot \mathbf{U}) \right) \mathbb{1} + \mu (\nabla \mathbf{U} + (\nabla \mathbf{U})^T). \quad (4.25)$$

Das Fouriersche Gesetz für die Wärmeleitung lautet

$$\mathbf{q} = -\xi \nabla T, \quad (4.26)$$

wobei ξ der Wärmeleitungskoeffizient ist. Im Falle einer laminaren Strömung eines Newtonischen und kompressiblen Fluids nimmt die Impulsgleichung die Form

$$\frac{\partial}{\partial t}(\rho \mathbf{U}) + (\rho \mathbf{U} \mathbf{U}^T) \cdot \nabla - \rho \boldsymbol{\Lambda} + \nabla P - \frac{1}{3}\mu \nabla \nabla \cdot \mathbf{U} - \mu \Delta \mathbf{U} = 0 \quad (4.27)$$

an. Bei inkompressiblen Fluiden, d.h. Fluiden mit konstanter Dichte, gilt

$$\frac{\partial \rho}{\partial t} = 0 \Rightarrow \nabla \cdot \mathbf{U} = 0. \quad (4.28)$$

Damit vereinfacht sich (4.22)-(4.24) zu

$$\frac{\partial \mathbf{U}}{\partial t} + (\mathbf{U} \mathbf{U}^T) \cdot \nabla - \boldsymbol{\Lambda} + \frac{\nabla P}{\rho} - \nu \Delta \mathbf{U} = 0 \quad (4.29)$$

$$\nabla \cdot \mathbf{U} = 0, \quad (4.30)$$

wobei $\nu = \frac{\mu}{\rho}$ die kinematische Viskosität ist.

4.4 Arbitrary-Lagrange-Euler-Formulierung

Bei der Arbitrary-Lagrange-Euler-Methode wird ein zusätzliches Referenzgebiet Ω_Y mit einer C^1 -stetigen Abbildung Φ eingeführt, so dass sich folgende Situation ergibt:

$$\Phi : (\Omega_X, T) \mapsto \Omega_t, \quad (\mathbf{X}, t) \mapsto \mathbf{x}(t) = \Phi(\mathbf{X}, t) \quad (4.31)$$

$$\Psi : (\Omega_Y, T) \mapsto \Omega_t, \quad (\mathbf{Y}, t) \mapsto \mathbf{y}(t) = \Psi(\mathbf{Y}, t), \quad (4.32)$$

wobei $\Omega_X \subset \mathbb{R}^3$, $\Omega_Y \subset \mathbb{R}^3$ und $\Omega_t \subset \mathbb{R}^3$ sind. Ω_Y ist das Gebiet, in dem die Gitterknoten enthalten sind. Die Abbildung 4.1 illustriert den Zusammenhang zwischen Ω_X , Ω_Y und Ω_t . Zu beachten ist, dass Ω_t beide Bildmengen durch Φ und durch Ψ enthält. D.h. es gibt einen Punkt

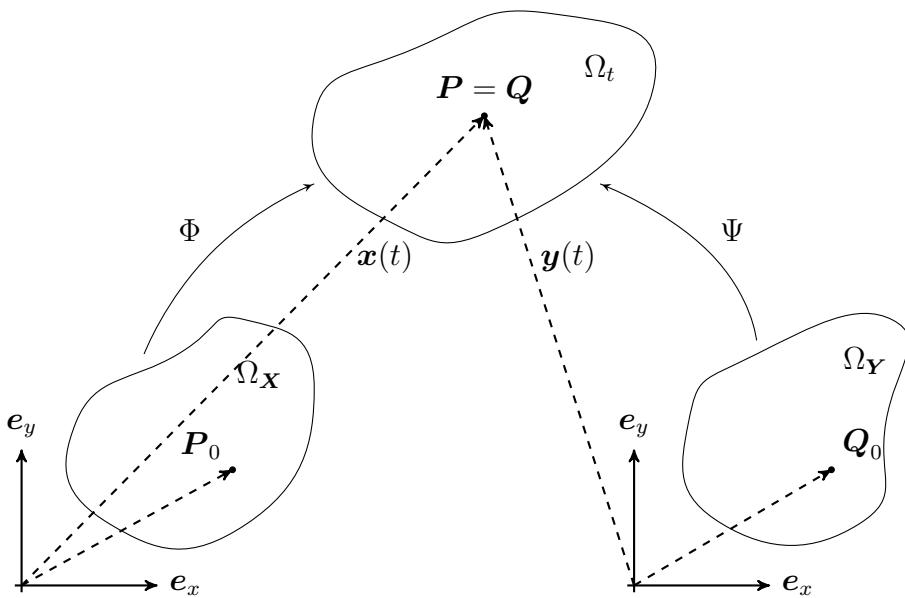


Abbildung 4.1: Veranschaulichung der ALE-Formulierung

\mathbf{P} , der zum Zeitpunkt t das Bild der Punkte \mathbf{P}_0 und \mathbf{Q}_0 ist. Analog zu der Definition in dem Unterabschnitt 4.1 erhält man die Geschwindigkeit des Gitters durch

$$\mathbf{W}(\mathbf{Y}, t) = \frac{\partial}{\partial t} (\Psi(\mathbf{Y}, t) - \Psi(\mathbf{Y}, 0)) = \frac{\partial \Psi(\mathbf{Y}, t)}{\partial t}. \quad (4.33)$$

Für eine beliebige skalare Funktion

$$\psi : \Omega_t \times T \mapsto \mathbb{R} \quad (4.34)$$

können wir eine spezielle Form des Reynolds'schen Transportsatzes formulieren

$$\frac{d}{dt} \int_{V_y} \psi d\mathbf{x} = \int_{V_y} \frac{\partial}{\partial t} \psi + \nabla \cdot (\psi \mathbf{W}) d\mathbf{x}, \quad (4.35)$$

wobei $V_y \subset \Omega_t$ das Bild des Teilvervolumens $V_Y \subset \Omega_Y$ ist. $\frac{d}{dt}$ ist die Zeitableitung mit der fixierten \mathbf{Y} Koordinate, was eine Art materielle Ableitung für das Gitter ist. Wir formulieren (4.35) um, und erhalten

$$\int_{V_y} \frac{\partial}{\partial t} \psi d\mathbf{x} = \frac{d}{dt} \int_{V_y} \psi d\mathbf{x} - \int_{V_y} \nabla \cdot (\psi \mathbf{W}) d\mathbf{x}. \quad (4.36)$$

Wir integrieren die Gleichung in (4.22) über V_y und erhalten

$$\int_{V_y} \frac{\partial \rho}{\partial t} d\mathbf{x} = - \int_{V_y} \nabla \cdot (\rho \mathbf{U}) d\mathbf{x}. \quad (4.37)$$

Ein Einsetzen von (4.37) in (4.36) liefert für $\psi = \rho$

$$\frac{d}{dt} \int_{V_y} \rho d\mathbf{x} + \int_{V_y} \nabla \cdot (\rho(\mathbf{U} - \mathbf{W})) d\mathbf{x} = 0. \quad (4.38)$$

Eine Integration von (4.23) über V_y liefert

$$\int_{V_y} \frac{\partial}{\partial t} (\rho \mathbf{U}) d\mathbf{x} + \int_{V_y} [(\rho \mathbf{U} \mathbf{U}^T) \cdot \nabla - \rho \boldsymbol{\Lambda} - \boldsymbol{\sigma} \cdot \nabla] d\mathbf{x} = 0. \quad (4.39)$$

Nach dem Einsetzen von (4.39) in (4.36) erhält man für $\psi = \rho \mathbf{U}$

$$\frac{d}{dt} \int_{V_y} \rho \mathbf{U} d\mathbf{x} + \int_{V_y} [(\rho \mathbf{U} (\mathbf{U}^T - \mathbf{W}^T)) \cdot \nabla - \rho \boldsymbol{\Lambda} - \boldsymbol{\sigma} \cdot \nabla] d\mathbf{x} = 0. \quad (4.40)$$

Die grundlegende Veränderung in den Gleichungen (4.38), (4.40) besteht in der Zeitableitung, die bezüglich der fixierten \mathbf{Y} Koordinate definiert ist. D.h. die zeitliche Änderung einer Funktion bezieht sich auf einen bestimmten Knoten im Gitter, der sich mit der Zeit bewegt. V_y ist dabei auch ein zeitlich veränderliches Integrationsgebiet. Damit entspricht V_y einem Volumenelement des darunterliegenden Gitters.

4.5 Geometrische Erhaltungssätze

In den meisten Problemstellungen ist die Geschwindigkeit des Gitters \mathbf{W} nicht als eine Funktion gegeben. Bei der Verformung eines Gitters ist \mathbf{W} durch die Geschwindigkeit der Knoten definiert. Die in dieser Arbeit verwendete Methode zur Gitterbewegung lässt nicht zu, die Gitterknoten zu verfolgen, da die Topologie der Flächenelemente sich mit der Zeit ändert. Es hat

sich herausgestellt, dass obwohl \mathbf{W} lediglich als Referenzgeschwindigkeit gilt, kann diese jedoch nicht jeden willkürlichen Wert annehmen. In [27] wurden für \mathbf{W} zwei Erhaltungssätze formuliert. Der erste lautet

$$\frac{d}{dt} \int_{V_y} d\mathbf{x} - \int_{V_y} \nabla \cdot \mathbf{W} d\mathbf{x} = 0. \quad (4.41)$$

Die Gleichung in (4.41) wird in der Literatur als *Volume Conservation Law* (VCL) oder auch als *Geometric Conservation Law* (GCL) bezeichnet. Eine andere Form von VCL ist

$$\frac{d}{dt} \int_{V_y} d\mathbf{x} = \int_{\partial V_y} \mathbf{W} \cdot \mathbf{n} d\mathbf{x}. \quad (4.42)$$

Durch (4.42) kann man die Bedeutung von VCL wie folgt interpretieren. Die Summe der orthogonalen Komponenten der Geschwindigkeiten der Oberfläche von V_y ist gleich der zeitlichen Änderung seines Volumens. Der Grundgedanke von VCL ist, dass ein bewegtes Gitter die Massenbilanz des dadurch diskretisierten Fluids nicht beeinflussen darf. VCL kann aus dem Reynolds'schen Transportsatz in (4.35) hergeleitet werden, indem man für $\psi = c \in \mathbb{R}$ setzt. Dieser Satz muss trivialerweise auch für konstante Funktionen (insbesondere für $\rho = c \in \mathbb{R}$) erfüllt sein.

Das zweite geometrische Erhaltungsgesetz ist das *Surface Conservation Law* (SCL)

$$\int_{\partial V_y} \mathbf{c} \cdot \mathbf{n} d\mathbf{x} = 0, \text{ mit } \mathbf{c} \in \mathbb{R}^3. \quad (4.43)$$

SCL ergibt sich aus VCL durch die Wahl einer konstanten Gittergeschwindigkeit $\mathbf{W} = \mathbf{c} \in \mathbb{R}^3$ für das ganze Rechengebiet und damit auch für V_y . Denn dabei muss zwingend

$$\frac{d}{dt} \int_{V_y} d\mathbf{x} = 0 \quad (4.44)$$

gelten. Auf den ersten Blick scheint SCL trivial zu sein, denn bei einem Volumenelement mit ebenen Flächen ist SCL immer erfüllt. Die wichtigste Aussage von SCL ist, dass ein Volumenelement durch seine Begrenzungsflächen abgeschlossen ist. Daher trägt SCL in manchen Quellen die Bezeichnung *Geometric Closeness*. SCL spielt eine große Rolle bei der räumlichen Diskretisierung und der Berechnung von Flächennormalen \mathbf{n} . Insbesondere ist eine große Vorsicht bei nicht ebenen Flächenelementen geboten. Die Nichteinhaltung von SCL führt zur Verletzung von VCL und damit auch zur Störung der Massenbilanz.

4.6 Diskretisierung von Differentialoperatoren

Die in diesem Abschnitt verwendeten Diskretisierungsschemata wurden der Literaturquelle [37] entnommen. Wir betrachten hier ein Newtonisches und inkompressibles Fluid. Da die zugehörige PDGL später mittels der Finite-Volumen-Methode diskretisiert wird, ist für den weiteren

Verlauf die integrale Form dieser PDGL relevant. Mit (4.38), (4.40) und (4.41) erhält man

$$\frac{d}{dt} \int_V d\mathbf{x} + \int_V \nabla \cdot \mathbf{U} d\mathbf{x} - \int_V \nabla \cdot \mathbf{W} d\mathbf{x} = 0 \quad (4.45)$$

$$\frac{d}{dt} \int_V \mathbf{U} d\mathbf{x} + \int_V [(\mathbf{U}(\mathbf{U}^T - \mathbf{W}^T)) \cdot \nabla - \boldsymbol{\Lambda} + \nabla p - \nu \Delta \mathbf{U}] d\mathbf{x} = 0 \quad (4.46)$$

$$\frac{d}{dt} \int_V d\mathbf{x} - \int_V \nabla \cdot \mathbf{W} d\mathbf{x} = 0, \quad (4.47)$$

wobei $V(t)$ ein mit der Zeit veränderliches Teilvolumen ist. Bei den Volumenintegralen $\int_V d\mathbf{x}$ haben wir dabei auf die Angabe des Parameters t verzichtet. Die Größen $\nu = \frac{\mu}{\rho}$ und $p = \frac{P}{\rho}$ sind die kinematische Viskosität und der kinematische Druck. Es ist gut erkennbar, dass bei der Erfüllung von (4.47) die Gleichung in (4.45) sich zu

$$\int_V \nabla \cdot \mathbf{U} d\mathbf{x} = 0 \quad (4.48)$$

vereinfacht. Dies gilt offensichtlich nur für inkompressible Fluide.

4.6.1 Zeitdiskretisierung

Zur Lösung des Systems von differentiellen Gleichungen in (4.45)-(4.47) wurden in den letzten Jahren sehr viele Ansätze entwickelt (siehe [17]). Die meisten davon kann man in zwei Klassen einteilen. Die erste basiert auf der DAE-Form von Navier-Stokes-Gleichungen (siehe [17, 21, 29, 42, 65, 68]) und die zweite gehört zu den sogenannten entkoppelten Lösungsmethoden (siehe [5, 6, 9, 25, 44, 78]).

In diesem Unterabschnitt werden wir uns mit Prämissen auseinandersetzen, die bei der Wahl des Verfahrens zur zeitlichen Diskretisierung von (4.45)-(4.47) entscheidend sind. Dafür abstrahieren wir uns von den bisherigen Ergebnissen und betrachten die Navier-Stokes-Gleichungen in Eulerscher Darstellung aus (4.29)-(4.30), die für inkompressive Fluide gelten. Diese haben die Form

$$\frac{\partial \mathbf{U}}{\partial t} + (\mathbf{U} \mathbf{U}^T) \cdot \nabla - \boldsymbol{\Lambda} + \frac{\nabla P}{\rho} - \nu \Delta \mathbf{U} = 0 \quad (4.49)$$

$$\nabla \cdot \mathbf{U} = 0, \quad (4.50)$$

wobei $\nu = \frac{\mu}{\rho}$ die kinematische Viskosität ist. Angelehnt an [29, 42] kann dieses Gleichungssystem durch die Diskretisierung von räumlichen Differentialoperatoren auf die Form

$$\dot{\mathbf{u}}(t) = \mathbf{K}(\mathbf{u}(t))\mathbf{u}(t) - \mathbf{B}\mathbf{p}(t) + \mathbf{f}(t) \quad (4.51)$$

$$\mathbf{B}^T \mathbf{u}(t) = 0 \quad (4.52)$$

gebracht werden. Dabei sind \mathbf{u} und \mathbf{p} die Vektoren mit Geschwindigkeiten und Drücken an diskreten Punkten im Ort. Die Matrix $\mathbf{K}(\mathbf{u}(t))$ beinhaltet alle linearen und nichtlinearen Operatoren, die auf die Geschwindigkeit angewandt werden. Die Matrix \mathbf{B} bzw. \mathbf{B}^T ist der diskrete Gradient bzw. der diskrete Divergenzoperator. Da in (4.51)-(4.52) für den Druck keine Zeitableitung vorkommt, handelt es sich hierbei um eine Algebraische-Differentialgleichung (Engl. DAE).

Bei der Wahl eines Integrationsverfahrens für eine DAE ist der sogenannte Differentiationsindex entscheidend (siehe [33, 38]). Dieser entspricht der Anzahl der Differentiationen, die zur Umwandlung einer DAE in eine gewöhnliche Differentialgleichung (Engl. ODE) benötigt werden. Als nächstes wollen wir den Differentiationsindex von (4.51)-(4.52) bestimmen. Wir lehnen uns dabei an [29]. Eine Differentiation von (4.52) führt zu

$$\mathbf{B}^T \dot{\mathbf{u}}(t) = 0 \quad (4.53)$$

Durch ein Einsetzen von (4.51) in (4.53) erhält man

$$\mathbf{B}^T (\mathbf{K}(\mathbf{u}(t))\mathbf{u}(t) - \mathbf{B}\mathbf{p}(t) + \mathbf{f}(t)) = 0 \quad (4.54)$$

Eine weitere Differentiation von (4.54) liefert

$$\mathbf{B}^T \left(\frac{\partial \mathbf{K}(\mathbf{u})}{\partial \mathbf{u}} \dot{\mathbf{u}}\mathbf{u} + \mathbf{K}(\mathbf{u})\dot{\mathbf{u}} - \mathbf{B}\dot{\mathbf{p}} + \dot{\mathbf{f}} \right) = 0 \quad (4.55)$$

Eine Umstellung von (4.55) nach $\dot{\mathbf{p}}$ führt zu

$$\dot{\mathbf{p}} = (\mathbf{B}^T \mathbf{B})^{-1} \mathbf{B}^T \left(\frac{\partial \mathbf{K}(\mathbf{u})}{\partial \mathbf{u}} \dot{\mathbf{u}}\mathbf{u} + \mathbf{K}(\mathbf{u})\dot{\mathbf{u}} + \dot{\mathbf{f}} \right) \quad (4.56)$$

Durch die Einführung der Funktion $\mathbf{h}(t, \mathbf{u}, \mathbf{p}) := \mathbf{K}(\mathbf{u})\mathbf{u} - \mathbf{B}\mathbf{p} + \mathbf{f}$ erhalten wir mit (4.51) und (4.56)

$$\dot{\mathbf{u}} = \mathbf{K}(\mathbf{u})\mathbf{u} - \mathbf{B}\mathbf{p} + \mathbf{f} = \mathbf{h}(t, \mathbf{u}, \mathbf{p}) \quad (4.57)$$

$$\dot{\mathbf{p}} = (\mathbf{B}^T \mathbf{B})^{-1} \mathbf{B}^T \left(\frac{\partial \mathbf{K}(\mathbf{u})}{\partial \mathbf{u}} \mathbf{h}\mathbf{u} + \mathbf{K}(\mathbf{u})\mathbf{h} + \dot{\mathbf{f}} \right). \quad (4.58)$$

Um die ODE (4.57)-(4.58) zu erhalten, wurden genau zwei Differentiationen benötigt, was auf den Differentiationsindex 2 deutet. Die Lösung von (4.57)-(4.58) ist nicht praktikabel, denn dafür hätte der Tensor dritter Stufe $\frac{\partial \mathbf{K}(\mathbf{u})}{\partial \mathbf{u}}$ aufgestellt werden müssen, was bei praxisrelevanten Problemstellungen einen immensen Rechenaufwand verursachen würde. Angelehnt an [29] kann für die DAE in (4.51)-(4.52) erst dann der Differentiationsindex bestimmt werden, wenn die Matrix \mathbf{B} vollen Spaltenrang hat. Andernfalls existiert die ODE in (4.58) nicht, da $\mathbf{B}^T \mathbf{B}$ nicht invertierbar ist. In der Literatur wird daher ein allgemeineres Indexkonzept verwendet, bei dem der sogenannte Strangeness-Index eingeführt wird. Angelehnt an [29, 42] hat die DAE (4.51)-(4.52) den Strangeness-Index 1.

Im weiteren beschäftigen wir uns mit dem Fall, bei dem \mathbf{B} den vollen Spaltenrang hat, so dass die DAE in (4.51)-(4.52) den Differentiationsindex 2 besitzt. Wir betrachten eine DAE

$$y = f(y, z) \quad (4.59)$$

$$0 = g(y), \quad (4.60)$$

die den Differentiationsindex 2 besitzt. Angelehnt an [33] sind die expliziten Verfahren für DAE mit dem Index 2 ungeeignet. Es kommt dabei zu dem sogenannten *Drift-Off-Effekt*, bei dem die numerische Lösung schon nach einem Integrationsschritt von der “echten” Lösung stark abweicht (siehe [38]). Reduktion der Schrittweite führt hierbei zu keiner Verbesserung. Laut [33] sind für DAE mit dem Index 2 die BDF-Verfahren (Engl. Backward Differentiation Formulas), zu denen auch das implizite Euler-Verfahren gehört, und die impliziten Runge-Kutta-Verfahren gut geeignet. In [51] wird gezeigt, dass das implizite Euler-Verfahren angewandt auf eine DAE mit

dem Index 2 seine Konsistenzordnung 1 nicht mehr behält. Der lokale Fehler für die Funktion z in (4.59) beträgt dabei $\mathcal{O}(h)$ für die Zeitschrittweite h , was der Konsistenzordnung 0 entspricht. D.h. das implizite Euler-Verfahren kann nicht zur Lösung von (4.51)-(4.52) verwendet werden.

Laut [33] behält ein k -Schritt BDF-Verfahren angewandt auf (4.59)-(4.60) seine Konsistenzordnung $p = k$ für $2 \leq k \leq 6$, benötigt jedoch dabei $k - 1$ zusätzliche Anfangswerte. Die Anfangswerte dürfen für $k = 2$ bzw. $3 \leq k \leq 6$ den Fehler $\mathcal{O}(h^{k+1})$ bzw. $\mathcal{O}(h^k)$ nicht überschreiten. Das bedeutet, um BDF-2 für (4.59)-(4.60) zu nutzen, brauchen wir ein zusätzliches Integrationsverfahren, das die Konsistenzordnung 2 für (4.59)-(4.60) besitzt. Dies könnte ein implizites Runge-Kutta-Verfahren sein. Die Implementierung eines impliziten Runge-Kutta-Verfahrens lediglich zur Berechnung der Anfangswerte für BDF-Verfahren erweist sich als nicht sinnvoll, da dabei vielzählige heutzutage existierende Randbedingungen in der Blockstruktur der Matrix des zu lösenden Gleichungssystems abgebildet werden müssen.

Die direkte Verwendung von Runge-Kutta-Verfahren zur Lösung von räumlich diskretisierten Navier-Stokes-Gleichungen in (4.51)-(4.52) stellt auch keine gute Alternative dar, denn dabei vergrößern sich die Matrizen der zu lösenden linearen Gleichungssysteme, wobei \mathbf{K} und \mathbf{B} bei relevanten Problemstellungen ohnehin die Dimensionen in der Größenordnung von zehn Millionen haben. Zusammenfassend und angelehnt an [29, 42] kann man zurückschließen, dass zur numerisch stabilen Lösung der DAE (4.51)-(4.52) eine Indexreduktion notwendig ist. Daher findet in den meisten heutzutage verwendeten Ansätzen zur zeitlichen Diskretisierung von (4.51)-(4.52) eine Indexreduktion statt, obwohl dies oft nicht explizit erwähnt wird.

In [21] wurde eine Projektionsmethode zur Lösung der Navier-Stokes-Gleichungen beschrieben, die ebenfalls eine Indexreduktion herbeiführt. In [65, 68] wird eine Methode beschrieben, bei der die DAE in (4.51)-(4.52) um eine spezielle Gleichung erweitert wird, so dass man eine DAE mit dem Differentiationsindex 1 erhält. Diese spezielle Gleichung wird aus der abgeleiteten Nebenbedingung, d.h. aus dem Massenerhaltungssatz, gewonnen.

Die meisten kommerziellen Programme sowie frei verfügbare Pakete verwenden den sogenannten SIMPLE-Algorithmus (Engl. Semi-implicit Method for Pressure Linked Equations), der zu der Klasse der entkoppelten Lösungsmethoden gehört (siehe [10, 78]). Da der im Rahmen dieser Arbeit entwickelte Code eine Erweiterung des frei verfügbaren Pakets OpenFOAM (Engl. Field Operation And Manipulation) ist (siehe [55]), haben wir uns für den in diesem Paket inhärenten SIMPLE-Algorithmus entschieden. Es ist zu berücksichtigen, dass bei der Verwendung eines alternativen Algorithmus vielzählige in OpenFOAM vorhandene Randbedingungen sowie die iterativen Löser für lineare Gleichungssysteme hätten reimplementiert werden müssen. Das Hauptaugenmerk dieser Arbeit richtet sich jedoch auf die Entwicklung eines Netzbewegungslösers für große Verschiebungen.

Die Idee des SIMPLE-Algorithmus besteht in einer Art Trennung der Geschwindigkeit vom Druck. Wir zeigen, wie dieser Algorithmus in Kombination mit dem impliziten Euler-Verfahren verwendet werden kann. Eine gewöhnliche Differentialgleichung der Form

$$\frac{d}{dt} \phi(t) = F(t, \phi(t)) \quad (4.61)$$

mit einer Funktion $\phi : \mathbb{R} \mapsto \mathbb{R}$, $t \mapsto \phi(t)$ wird durch die Anwendung des impliziten Euler-Verfahrens auf folgende Vorschrift zurückgeführt:

$$\frac{\phi(t^{k+1}) - \phi(t^k)}{\Delta t} = F(t^{k+1}, \phi(t^{k+1})). \quad (4.62)$$

Eine Anwendung des impliziten Euler-Verfahrens auf (4.51)-(4.52) führt zu

$$\frac{\mathbf{u}^{k+1} - \mathbf{u}^k}{\Delta t} = \mathbf{K}\mathbf{u}^{k+1} - \mathbf{B}\mathbf{p}^{k+1} + \mathbf{f}^{k+1} \quad (4.63)$$

$$0 = \mathbf{B}^T \mathbf{u}^{k+1}, \quad (4.64)$$

wobei Δt die Zeitschrittweite und k der Zeitindex sind. Wir führen eine Variable $\tilde{\mathbf{p}}$, so dass

$$\mathbf{p}^{k+1} = \mathbf{p}^k + \tilde{\mathbf{p}} \quad (4.65)$$

gilt. Mit einer zusätzlichen Variablen $\tilde{\mathbf{u}}$ und (4.65) erhält man aus (4.63)

$$\frac{\mathbf{u}^{k+1} - \tilde{\mathbf{u}} + \tilde{\mathbf{u}} - \mathbf{u}^k}{\Delta t} = \mathbf{K}\mathbf{u}^{k+1} - \mathbf{B}(\mathbf{p}^k + \tilde{\mathbf{p}}) + \mathbf{f}^{k+1}. \quad (4.66)$$

Wir spalten (4.66) in zwei Gleichungen

$$\frac{\tilde{\mathbf{u}} - \mathbf{u}^k}{\Delta t} = \mathbf{K}\mathbf{u}^{k+1} - \mathbf{B}\mathbf{p}^k + \mathbf{f}^{k+1} \quad (4.67)$$

$$\frac{\mathbf{u}^{k+1} - \tilde{\mathbf{u}}}{\Delta t} = -\mathbf{B}\tilde{\mathbf{p}} \quad (4.68)$$

auf. Denn die Summe von (4.67) und (4.68) liefert (4.66). Wir setzen (4.68) in (4.64) ein, und erhalten

$$\Delta t \mathbf{B}^T \mathbf{B} \tilde{\mathbf{p}} = \mathbf{B}^T \tilde{\mathbf{u}}. \quad (4.69)$$

Zusammenfassend erhält man mit (4.63), (4.65), (4.67), (4.68) und (4.69)

$$\frac{\mathbf{u}^{k+1} - \mathbf{u}^k}{\Delta t} = \mathbf{K}\mathbf{u}^{k+1} - \mathbf{B}\mathbf{p}^{k+1} + \mathbf{f}^{k+1} \quad (4.70)$$

$$\tilde{\mathbf{u}} = \mathbf{u}^k + \Delta t (\mathbf{K}\mathbf{u}^{k+1} - \mathbf{B}\mathbf{p}^k + \mathbf{f}^{k+1}) \quad (4.71)$$

$$\Delta t \mathbf{B}^T \mathbf{B} \tilde{\mathbf{p}} = \mathbf{B}^T \tilde{\mathbf{u}} \quad (4.72)$$

$$\mathbf{p}^{k+1} = \mathbf{p}^k + \tilde{\mathbf{p}} \quad (4.73)$$

$$\mathbf{u}^{k+1} = \tilde{\mathbf{u}} - \Delta t \mathbf{B} \tilde{\mathbf{p}}, \quad (4.74)$$

Ein großer Vorteil des Gleichungssystems in (4.70)-(4.74) ist, dass dieses der diskreten Form von einer DAE mit dem Differentiationsindex 1 entspricht. Denn es gibt eine spezielle Gleichung für den Druck (4.72). Eine Zeitableitung in der „kontinuierlichen“ Form würde zu einer ODE führen. Ein weiterer Vorteil von (4.70)-(4.74) ist, dass anstelle eines großen linearen Gleichungssystems nur zwei kleinere Gleichungssysteme (4.70) und (4.72) gelöst werden müssen, was zu einer Reduktion des benötigten Speicherplatzes führt. Zu beachten ist, dass die Schritte (4.70)-(4.74) nur eine einzige Iteration des SIMPLE-Algorithmus darstellen. In der ersten Iteration wird in (4.70) $\mathbf{p}^{k+1} = \mathbf{p}^k$ gesetzt.

In dieser Arbeit verwenden wir den SIMPLE-Algorithmus in Kombination mit dem impliziten Euler-Verfahren zur zeitlichen Diskretisierung des Systems in (4.45)-(4.47).

4.6.2 Ortsdiskretisierung mittels der Finite-Volumen-Methode

Die Finite-Volumen-Methode hat ihre Bezeichnung dadurch erhalten, dass diese bei der Diskretisierung der räumlichen Differentialoperatoren wie ∇ , Δ oder $(\nabla \cdot)$ als Bezugselemente die

endlichen Teilvolumina V_i verwendet. Das betrachtete Berechnungsgebiet Ω wird in N disjunkte und konvexe Teilvolumina V_i zerlegt. Es wird gefordert, dass die darunterliegende PDGL in integraler Form für jedes Teilvolumen erfüllt ist. Meistens werden die gesuchten Funktionen an den geometrischen Schwerpunkten von V_i erfasst. Dies hat folgende Motivation. Für eine mehrmals stetig differenzierbare Funktion

$$\phi : \mathbb{R}^3 \mapsto \mathbb{R}, \quad \mathbf{x} \mapsto \phi(\mathbf{x}), \quad (4.75)$$

die an einem Punkt $\mathbf{x}_c \in \mathbb{R}^3$ bekannt ist, lautet die Taylor-Reihenentwicklung

$$\phi(\mathbf{x}) = \phi(\mathbf{x}_c) + (\mathbf{x} - \mathbf{x}_c) \cdot \nabla \phi(\mathbf{x}_c) + \mathcal{O}(\|\mathbf{x} - \mathbf{x}_c\|^2). \quad (4.76)$$

Eine Integration von (4.76) über ein Volumen $V \subset \mathbb{R}^3$ liefert

$$\int_V \phi(\mathbf{x}) d\mathbf{x} = \int_V \phi(\mathbf{x}_c) d\mathbf{x} + \int_V (\mathbf{x} - \mathbf{x}_c) \cdot \nabla \phi(\mathbf{x}_c) d\mathbf{x} + \int_V \mathcal{O}(\|\mathbf{x} - \mathbf{x}_c\|^2) d\mathbf{x} \quad (4.77)$$

Wenn \mathbf{x}_c der geometrische Schwerpunkt des Volumenelements V ist, gilt

$$\mathbf{x}_c = \frac{\int_V \mathbf{x} d\mathbf{x}}{\int_V d\mathbf{x}} \Rightarrow \int_V (\mathbf{x} - \mathbf{x}_c) d\mathbf{x} = 0. \quad (4.78)$$

Mit (4.78) erhält man aus (4.77)

$$\int_V \phi(\mathbf{x}) d\mathbf{x} = \phi(\mathbf{x}_c) \int_V d\mathbf{x} + \int_V \mathcal{O}(\|\mathbf{x} - \mathbf{x}_c\|^2) d\mathbf{x}. \quad (4.79)$$

Die Gleichung in (4.79) bedeutet, dass ein Volumenintegral von ϕ über V durch den Wert $\phi(\mathbf{x}_c)$ multipliziert mit dem Volumeninhalt V bis auf den Abbruchsfehler zweiter Ordnung angenähert werden kann, was eine sehr effiziente numerische Rechenmethode ist.

Die Diskretisierung des Divergenzoperators bzw. des Gradienten erfolgt durch die Umwandlung des Volumenintegrals in das Flächenintegral mithilfe des Gausschen-Satzes. Die zu lösende PDGL (4.45)-(4.47) nimmt damit die Form

$$\int_{\partial V} \mathbf{U} \cdot \mathbf{n} d\mathbf{x} = 0 \quad (4.80)$$

$$\frac{d}{dt} \int_V \mathbf{U} d\mathbf{x} + \int_{\partial V} [(\mathbf{U}(\mathbf{U}^T - \mathbf{W}^T) - \nu \nabla \mathbf{U} + p \mathbf{1}) \cdot \mathbf{n}] d\mathbf{x} - \int_V \Lambda d\mathbf{x} = 0 \quad (4.81)$$

$$\frac{d}{dt} \int_V d\mathbf{x} - \int_{\partial V} \mathbf{W} \cdot \mathbf{n} d\mathbf{x} = 0 \quad (4.82)$$

an. Dabei sind \mathbf{n} die Flächennormale und ∂V der Rand von V . Analog zu (4.79) erfolgt die Approximation der Flächenintegrale mittels

$$\int_{\partial V} \phi(\mathbf{x}) d\mathbf{x} = \phi(\mathbf{x}_f) \int_{\partial V} d\mathbf{x} + \int_{\partial V} \mathcal{O}(\|\mathbf{x} - \mathbf{x}_f\|^2) d\mathbf{x}, \quad (4.83)$$

wobei \mathbf{x}_f der geometrische Schwerpunkt des entsprechenden Flächenelements ist. Wir zeigen hier eine mögliche Variante der räumlichen Diskretisierung von (4.80)-(4.82). In speziellen Fällen können auch genauere Diskretisierungsschemata verwendet werden. Außerdem werden wir uns nur mit inneren Volumenelementen beschäftigen. Eine ausführliche Betrachtung der Randbedingungen sowie weiterer Diskretisierungsschemata werden in [37] angeboten. Im weiteren Verlauf verwenden wir folgende Bezeichnungen:

- N ist die Anzahl der Volumenelemente
- \mathbf{x}_i ist der geometrische Schwerpunkt des i -ten Volumenelements V_i
- \mathcal{N}_i ist die Menge der Nachbarindizes von V_i
- \mathbf{x}_{ij} ist der geometrische Schwerpunkt des Flächenelements zwischen den Volumenelementen V_i und V_j
- Die Hervorhebung $(\cdot)_{ij}$ soll auch zum Ausdruck bringen, dass die gekennzeichnete Größe auf dem Flächenelement zwischen V_i und V_j definiert ist. Meistens auf dem Flächenschwerpunkt \mathbf{x}_{ij} .

Für eine Funktion $\phi(\mathbf{x})$ ist $\nabla\phi \cdot \mathbf{n}$ die Ableitung in Richtung \mathbf{n} . Die Richtungsableitung $\nabla\phi \cdot \mathbf{n}$ auf dem Flächenschwerpunkt \mathbf{x}_{ij} kann durch

$$\nabla\phi(\mathbf{x}_{ij}) \cdot \mathbf{n}_{ij} \approx \frac{\phi(\mathbf{x}_j) - \phi(\mathbf{x}_i)}{\|\mathbf{x}_j - \mathbf{x}_i\|} \quad (4.84)$$

angenähert werden. Für ein allgemeines Gitter mit konvexen Elementen ist diese Näherung bis auf den Abbruchsfehler erster Ordnung genau (siehe [37]). Die Interpolation einer Funktion $\phi(\mathbf{x})$ auf das Flächenelement \mathbf{x}_{ij} erfolgt durch

$$\phi(\mathbf{x}_{ij}) \approx (1 - \gamma_{ij})\phi(\mathbf{x}_i) + \gamma_{ij}\phi(\mathbf{x}_j), \text{ mit } \gamma_{ij} = \gamma(\mathbf{x}_{ij}) = \frac{\|\mathbf{x}_{ij} - \mathbf{x}_i\|}{\|\mathbf{x}_j - \mathbf{x}_i\|}. \quad (4.85)$$

Die Approximation der Zeitableitung mittels des impliziten Euler-Verfahrens sowie die Approximation der Volumen- bzw. der Flächenintegrale führt zu

$$\sum_{j \in \mathcal{N}_i} \mathbf{U}_{ij}^k \cdot \mathbf{n}_{ij}^k A_{ij}^k = 0 \quad (4.86)$$

$$\frac{\mathbf{U}_i^k V_i^k - \mathbf{U}_i^{k-1} V_i^{k-1}}{\Delta t} \quad (4.87)$$

$$+ \sum_{j \in \mathcal{N}_i} \left(\mathbf{U}_{ij}^k (\mathbf{U}_{ij}^k \cdot \mathbf{n}_{ij}^k A_{ij}^k - \mathbf{W}_{ij}^k \cdot \mathbf{n}_{ij}^k A_{ij}^k) - \nu A_{ij}^k \frac{\mathbf{U}_j^k - \mathbf{U}_i^k}{\|\mathbf{x}_j^k - \mathbf{x}_i^k\|} + p_{ij}^k \mathbf{n}_{ij}^k A_{ij}^k \right) = V_i^k \Lambda$$

$$\frac{V_i^k - V_i^{k-1}}{\Delta t} = \sum_{j \in \mathcal{N}_i} \mathbf{W}_{ij}^k \cdot \mathbf{n}_{ij}^k A_{ij}^k, \quad (4.88)$$

wobei der Index rechts oben k bzw. rechts unten i sich auf die Zeit t^k bzw. auf den Zellmittelpunkt \mathbf{x}_i bezieht. A_{ij}^k ist der Inhalt der Fläche, die zu dem Flächenmittelpunkt \mathbf{x}_{ij} zugeordnet ist. \mathbf{n}_{ij}^k ist ihre zugehörige Flächennormale. Die skalare Größe

$$M_{ij}^k := \mathbf{W}_{ij}^k \cdot \mathbf{n}_{ij}^k A_{ij}^k \quad (4.89)$$

taucht in englischer Literatur meistens unter dem Namen *mesh flux* auf. Wir bezeichnen diese Größe als Netzfluss. Die Gleichung in (4.88) stellt eine Nebenbedingung für die Netzflüsse dar. Im Abschnitt 4.7 wird beschrieben, wie in dieser Arbeit die das VCL erfüllende Netzflüsse bestimmt werden. Diese Netzflüsse werden in jedem Zeitschritt berechnet und in (4.87) eingesetzt.

4.6.3 SIMPLE-Algorithmus für die FVM

Es wurde bereits erwähnt, dass der im Rahmen dieser Arbeit entwickelte Code eine Erweiterung des frei verfügbaren Pakets *OpenFOAM* (siehe Kapitel 5) darstellt. Die in OpenFOAM eingesetzte Variante des SIMPLE-Algorithmus ist bedauerlicherweise nicht dokumentiert, weshalb wir in diesem Abschnitt eine ausführliche Herleitung dieser Variante anbieten.

Die Idee des SIMPLE-Algorithmus besteht in einer Aufspaltung der diskreten Impulsgleichung in zwei Gleichungen, wobei eine der Gleichungen in die Kontinuitätsgleichung eingesetzt wird, wodurch eine spezielle Gleichung für den Druck erzeugt wird (siehe [5, 6, 9, 25, 44, 78]). Da das Gleichungssystem entkoppelt wird, müssen einige Iterationen durchgeführt werden, bis sowohl die Impulsgleichung als auch die Kontinuitätsgleichung erfüllt ist. Durch diesen iterativen Algorithmus wird gleichzeitig der nichtlineare Konvektionsterm $\mathbf{U}_{ij}^k (\mathbf{U}_{ij}^k \cdot \mathbf{n}_{ij}^k)$ "linearisiert". Die Größe

$$\Phi_{ij}^k := \mathbf{U}_{ij}^k \cdot \mathbf{n}_{ij}^k A_{ij}^k \quad (4.90)$$

bezeichnet man als Massenfluss (Engl. *mesh flux*). Mit (4.85), (4.89) und (4.90) nimmt (4.87) die Form

$$\begin{aligned} & \frac{\mathbf{U}_i^k V_i^k - \mathbf{U}_i^{k-1} V_i^{k-1}}{\Delta t} \\ & + \left(\sum_{j \in \mathcal{N}_i} ((1 - \gamma_{ij}^k) \mathbf{U}_i^k + \gamma_{ij}^k \mathbf{U}_j^k) (\Phi_{ij}^k - M_{ij}^k) - \nu A_{ij}^k \frac{\mathbf{U}_j^k - \mathbf{U}_i^k}{\|\mathbf{x}_j^k - \mathbf{x}_i^k\|} + p_{ij}^k \mathbf{n}_{ij}^k A_{ij}^k \right) = V_i^k \Lambda \end{aligned} \quad (4.91)$$

an. Wir setzen $\omega_{ij}^k := \|\mathbf{x}_j^k - \mathbf{x}_i^k\|$ und stellen (4.91) nach \mathbf{U}_i^k um

$$\begin{aligned} & \left(\frac{V_i^k}{\Delta t} + \sum_{j \in \mathcal{N}_i} \left((1 - \gamma_{ij}^k) (\Phi_{ij}^k - M_{ij}^k) + \frac{\nu A_{ij}^k}{\omega_{ij}^k} \right) \right) \mathbf{U}_i^k \\ & = - \sum_{j \in \mathcal{N}_i} \left(\gamma_{ij}^k (\Phi_{ij}^k - M_{ij}^k) - \frac{\nu A_{ij}^k}{\omega_{ij}^k} \right) \mathbf{U}_j^k + \frac{V_i^{k-1}}{\Delta t} \mathbf{U}_i^{k-1} - \sum_{j \in \mathcal{N}_i} p_{ij}^k \mathbf{n}_{ij}^k A_{ij}^k + \Lambda V_i^k. \end{aligned} \quad (4.92)$$

Wir bezeichnen in (4.92) den Multiplikator vor \mathbf{U}_i^k mit C_i^k und die Terme vor \mathbf{U}_j^k mit C_j^k . D.h. es gilt

$$C_i^k = \frac{V_i^k}{\Delta t} + \sum_{j \in \mathcal{N}_i} (1 - \gamma_{ij}^k) (\Phi_{ij}^k - M_{ij}^k) + \frac{\nu A_{ij}^k}{\|\mathbf{x}_j^k - \mathbf{x}_i^k\|} \quad (4.93)$$

$$C_j^k = \gamma_{ij}^k (\Phi_{ij}^k - M_{ij}^k) - \frac{\nu A_{ij}^k}{\|\mathbf{x}_j^k - \mathbf{x}_i^k\|} \text{ für } j \in \mathcal{N}_i. \quad (4.94)$$

Damit nimmt (4.92) die Form

$$C_i^k \mathbf{U}_i^k = - \sum_{j \in \mathcal{N}_i} C_j^k \mathbf{U}_j^k + \frac{V_i^{k-1}}{\Delta t} \mathbf{U}_i^{k-1} - \sum_{j \in \mathcal{N}_i} p_{ij}^k \mathbf{n}_{ij}^k A_{ij}^k + \Lambda V_i^k. \quad (4.95)$$

an. Wir führen die Variablen $\tilde{\mathbf{U}}_i$ und \tilde{p}_i ein, so dass

$$p_i^k = p_i^{k-1} + \tilde{p}_i \quad (4.96)$$

gilt. Mit (4.96) und $\tilde{\mathbf{U}}_i$ erhalten wir aus (4.95)

$$\begin{aligned} & C_i^k \left(\mathbf{U}_i^k - \tilde{\mathbf{U}}_i + \tilde{\mathbf{U}}_i \right) \\ &= - \sum_{j \in \mathcal{N}_i} C_j^k \mathbf{U}_j^k + \frac{V_i^{k-1}}{\Delta t} \mathbf{U}_i^{k-1} - \sum_{j \in \mathcal{N}_i} (p_{ij}^{k-1} + \tilde{p}_{ij}) \mathbf{n}_{ij}^k A_{ij}^k + \Lambda V_i^k. \end{aligned} \quad (4.97)$$

Wir spalten (4.97) in zwei Gleichungen

$$C_i^k \tilde{\mathbf{U}}_i = - \sum_{j \in \mathcal{N}_i} C_j^k \mathbf{U}_j^k + \frac{V_i^{k-1}}{\Delta t} \mathbf{U}_i^{k-1} - \sum_{j \in \mathcal{N}_i} p_{ij}^{k-1} \mathbf{n}_{ij}^k A_{ij}^k + \Lambda V_i^k \quad (4.98)$$

$$C_i^k \left(\mathbf{U}_i^k - \tilde{\mathbf{U}}_i \right) = - \sum_{j \in \mathcal{N}_i} \tilde{p}_{ij} \mathbf{n}_{ij}^k A_{ij}^k \quad (4.99)$$

auf. Die Summe von (4.98) und (4.99) liefert (4.97). Durch eine Umstellung nach \mathbf{U}_i^k erhält man aus (4.99)

$$\mathbf{U}_i^k = \tilde{\mathbf{U}}_i - \frac{1}{C_i^k} \sum_{j \in \mathcal{N}_i} \tilde{p}_{ij} \mathbf{n}_{ij}^k A_{ij}^k. \quad (4.100)$$

Wir interpolieren \mathbf{U}_i^k auf die Flächenschwerpunkte und setzen diese in die diskrete Kontinuitätsgleichung (4.86) ein. Dies führt zu

$$\sum_{j \in \mathcal{N}_i} \left(\tilde{\mathbf{U}}_i - \frac{1}{C_i^k} \sum_{j \in \mathcal{N}_i} \tilde{p}_{ij} \mathbf{n}_{ij}^k A_{ij}^k \right)_{ij} \cdot \mathbf{n}_{ij}^k A_{ij}^k = 0. \quad (4.101)$$

Anwendung des „diskreten“ Gaußschen Satzes liefert

$$\sum_{j \in \mathcal{N}_i} \tilde{p}_{ij} \mathbf{n}_{ij}^k A_{ij}^k = \nabla \tilde{p}_i V_i^k. \quad (4.102)$$

Wir setzen (4.102) in (4.101) und erhalten

$$\sum_{j \in \mathcal{N}_i} \left(\tilde{\mathbf{U}}_i - \frac{1}{C_i^k} \nabla \tilde{p}_i V_i^k \right)_{ij} \cdot \mathbf{n}_{ij}^k A_{ij}^k = 0. \quad (4.103)$$

Wir interpolieren $\tilde{\mathbf{U}}_i$, $\frac{V_i^k}{C_i^k}$ und $\nabla \tilde{p}_i$ auf die Flächenmittelpunkte und erhalten damit aus (4.103)

$$\sum_{j \in \mathcal{N}_i} \left(\frac{V_i^k}{C_i^k} \right)_{ij} \nabla \tilde{p}_{ij} \cdot \mathbf{n}_{ij}^k A_{ij}^k = \sum_{j \in \mathcal{N}_i} \tilde{\mathbf{U}}_{ij} \cdot \mathbf{n}_{ij}^k A_{ij}^k. \quad (4.104)$$

Die Gleichung (4.104) entspricht der diskreten stationären Diffusionsgleichung mit dem Quellterm $\nabla \tilde{\mathbf{U}}$ und dem Diffusionskoeffizienten $\frac{V^k}{C^k}$. Diese Gleichung würde die Form

$$\nabla \cdot \frac{V^k}{C^k} \nabla \tilde{p} = \nabla \tilde{\mathbf{U}} \quad (4.105)$$

haben. Zusammenfassend erhalten wir mit (4.95), (4.96), (4.98), (4.100) und (4.104)

$$C_i^k \mathbf{U}_i^k + \sum_{j \in \mathcal{N}_i} C_j^k \mathbf{U}_j^k = \frac{V_i^{k-1}}{\Delta t} \mathbf{U}_i^{k-1} - \sum_{j \in \mathcal{N}_i} p_{ij}^k \mathbf{n}_{ij}^k A_{ij}^k + \boldsymbol{\Lambda} V_i^k \quad (4.106)$$

$$\tilde{\mathbf{U}}_i = \left(- \sum_{j \in \mathcal{N}_i} C_j^k \mathbf{U}_j^k + \frac{V_i^{k-1}}{\Delta t} \mathbf{U}_i^{k-1} - \sum_{j \in \mathcal{N}_i} p_{ij}^{k-1} \mathbf{n}_{ij}^k A_{ij}^k + \boldsymbol{\Lambda} V_i^k \right) / C_i^k \quad (4.107)$$

$$\sum_j \left(\frac{V_i^k}{C_i^k} \right)_{ij} \nabla \tilde{p}_{ij} \cdot \mathbf{n}_{ij}^k A_{ij}^k = \sum_j \tilde{\mathbf{U}}_{ij} \cdot \mathbf{n}_{ij}^k A_{ij}^k \quad (4.108)$$

$$p_i^k = p_i^{k-1} + \tilde{p}_i \quad (4.109)$$

$$\mathbf{U}_i^k = \tilde{\mathbf{U}}_i - \frac{1}{C_i^k} \sum_{j \in \mathcal{N}_i} \tilde{p}_{ij} \mathbf{n}_{ij}^k A_{ij}^k. \quad (4.110)$$

Die Gleichung in (4.108) ist die Poisson-Gleichung für den Druck. Durch Ersetzung von p_i^k und \mathbf{U}_i^k durch die „iterativen“ Größen erhalten wir den Algorithmus 4.1. Diese iterativen Größen werden im Algorithmus in **rot** gekennzeichnet. Ein Nachteil des Algorithmus 4.1 ist, dass die Erfüllung der Kontinuitätsgleichung für die Geschwindigkeit durch die Interpolationsfehler im Schritt 4. beeinträchtigt wird. Es ist daher wichtig in diesem Schritt eine möglichst genauere Interpolationsschema zu verwenden.

In der Literatur werden meistens im Zusammenhang mit dem SIMPLE-Algorithmus solche Begriffe wie Prädiktor und Korrektor verwendet. Im Algorithmus 4.1 entsprechen die Schritte 1-2 dem Prädiktor und die Schritte 3-6 dem Korrektor. Da die durch den Prädiktor berechnete Geschwindigkeit nicht divergenzfrei ist, empfiehlt es sich mehrere Iterationen des Korrektors durchzuführen.

Bemerkung 4.1:

Im Vergleich zu der SIMPLE-Vorschrift im Unterabschnitt 4.6.1 wurde in der hier verwendeten Variante des SIMPLE-Algorithmus die Impulsgleichung an den Diagonalelementen der betreffenden Matrix aufgespalten (vergleiche (4.70)-(4.74) und (4.106)-(4.110)).

Algorithmus 4.1 SIMPLE-Algorithmus

Seien \mathbf{U}_i^{k-1} , p_i^{k-1} und $M_{ij}^k = \mathbf{W}_{ij}^k \cdot \mathbf{n}_{ij}^k A_{ij}^k$ für alle Zell- bzw. Flächenmittelpunkte gegeben.
Wir suchen \mathbf{U}_i^k und p_i^k für $i = 1, 2, \dots, N$, wobei N die Anzahl der Zellen ist.
Setze für $i = 1, 2, \dots, N$

$$\mathbf{U}_i^k = \mathbf{U}_i^{k-1} \quad (4.111)$$

$$p_i^k = p_i^{k-1} \quad (4.112)$$

while Toleranzkriterium nicht erfüllt ist **do**

1. Interpoliere \mathbf{U}_i^k auf die Flächenmittelpunkte und werte für $i = 1, 2, \dots, N$ aus:

$$\Phi_{ij}^k = \mathbf{U}_{ij}^k \cdot \mathbf{n}_{ij}^k A_{ij}^k \quad (4.113)$$

$$C_i^k = \frac{V_i^k}{\Delta t} + \sum_{j \in \mathcal{N}_i} (1 - \gamma_{ij}^k) (\Phi_{ij}^k - M_{ij}^k) + \frac{\nu A_{ij}^k}{\|\mathbf{x}_j^k - \mathbf{x}_i^k\|} \quad (4.114)$$

$$C_j^k = \gamma_{ij}^k (\Phi_{ij}^k - M_{ij}^k) - \frac{\nu A_{ij}^k}{\|\mathbf{x}_j^k - \mathbf{x}_i^k\|} \text{ für } j \in \mathcal{N}_i. \quad (4.115)$$

2. Löse für $i = 1, 2, \dots, N$ das Gleichungssystem nach \mathbf{U}_i^k auf:

$$C_i^k \mathbf{U}_i^k + \sum_{j \in \mathcal{N}_i} C_j^k \mathbf{U}_j^k = \frac{V_i^{k-1}}{\Delta t} \mathbf{U}_i^{k-1} + \Lambda V_i^k - \sum_{j \in \mathcal{N}_i} p_{ij}^k \mathbf{n}_{ij}^k A_{ij}^k. \quad (4.116)$$

3. Berechne für $i = 1, 2, \dots, N$

$$\tilde{\mathbf{U}}_i = \left(- \sum_{j \in \mathcal{N}_i} C_j^k \mathbf{U}_j^k + \frac{V_i^{k-1}}{\Delta t} \mathbf{U}_i^{k-1} - \sum_{j \in \mathcal{N}_i} p_{ij}^{k-1} \mathbf{n}_{ij}^k A_{ij}^k + \Lambda V_i^k \right) / C_i^k \quad (4.117)$$

4. Interpoliere $\tilde{\mathbf{U}}_i$, C_i^k und V_i^k auf die Flächenschwerpunkte

5. Löse für $i = 1, 2, \dots, N$ das Gleichungssystem nach \tilde{p}_i auf:

$$\sum_j \left(\frac{V_i^k}{C_i^k} \right)_{ij} \nabla \tilde{p}_{ij} \cdot \mathbf{n}_{ij}^k A_{ij}^k = \sum_j \tilde{\mathbf{U}}_{ij} \cdot \mathbf{n}_{ij}^k A_{ij}^k \quad (4.118)$$

6. Berechne für $i = 1, 2, \dots, N$

$$p_i^k = p_i^{k-1} + \tilde{p}_i \quad (4.119)$$

$$\mathbf{U}_i^k = \tilde{\mathbf{U}}_i - \frac{1}{C_i^k} \sum_{j \in \mathcal{N}_i} \tilde{p}_{ij} \mathbf{n}_{ij}^k A_{ij}^k \quad (4.120)$$

end while

4.6.4 Gitterqualitätskriterien im Bezug auf die FVM

In diesem Abschnitt werden die wichtigsten Kriterien definiert, die maßgeblich für die Qualität der Gitter sind. Darüber hinaus wird kurz beschrieben, welchen Einfluss diese Kriterien auf die bei der räumlichen Diskretisierung verwendeten Diskretisierungsschemata haben. Dabei werden die Schemata zweiter Ordnung untersucht. Wir bedienen uns dabei der Literatur [37]. Die in diesem Unterabschnitt verwendeten Größen beziehen sich auf die Abbildung 4.2.

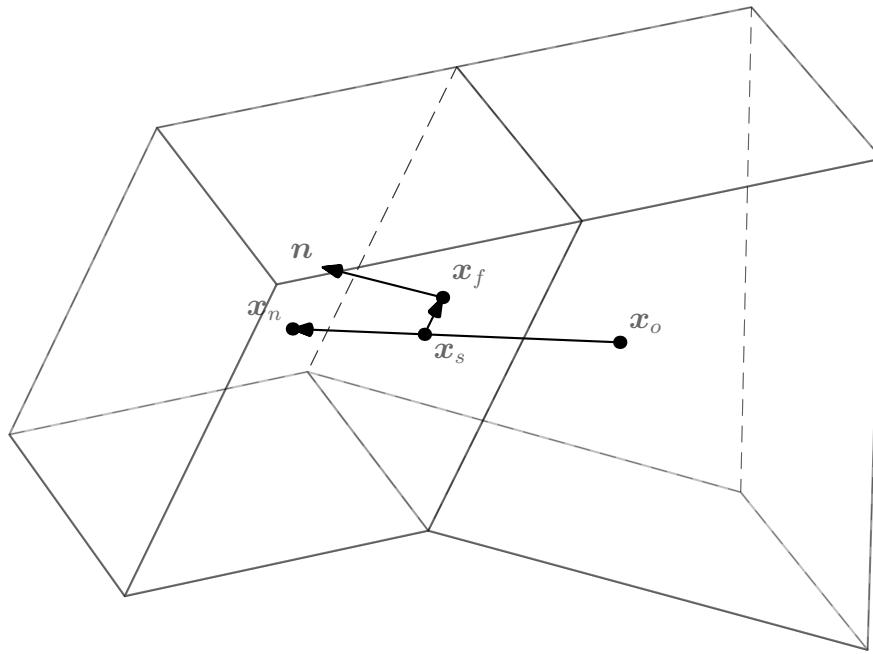


Abbildung 4.2: Verbindungslinie der Zellenschwerpunkte im Bezug zu dem Flächenschwerpunkt. x_o und x_n sind die Schwerpunkte der benachbarten Zellen. x_f ist der Schwerpunkt der Fläche. n ist die normierte Flächennormale. D.h. $\|n\| = 1$. Außerdem gilt $u := x_n - x_o$ und $v := x_f - x_s$.

- **Nicht-Orthogonalität (Non-orthogonality)** Das Maß dafür ist ein zwischen den Vektoren u und n eingeschlossener Winkel α . D.h. es gilt

$$\cos(\alpha) = \frac{u \cdot n}{\|u\|}. \quad (4.121)$$

- **Netzschiefe (Mesh skewness)** Wenn der Vektor u nicht durch x_f verläuft, bezeichnet man das darunterliegende Gitter als schief. Die charakteristische Größe ist

$$\beta = \frac{\|v\|}{\|u\|}. \quad (4.122)$$

- **Homogenität (Uniformity)** Das Netz ist homogen, wenn der Schnittpunkt x_s der Mittelwert von x_n und x_o ist. Definiert wird diese Größe durch

$$\gamma = \frac{\|x_s - x_o\|}{\|u\|}. \quad (4.123)$$

Bei homogenen Gittern muss γ stets 0.5 sein.

In [37] wurden Abschätzungen für die Fehler gemacht, die bei der Diskretisierung der Ableitungsoperatoren mittels FVM entstehen. Beim Interpolieren einer Funktion ϕ von $\phi(\mathbf{x}_o)$ und $\phi(\mathbf{x}_n)$ auf $\phi(\mathbf{x}_f)$ entsteht der Fehler

$$\phi(\mathbf{x}_f) - \tilde{\phi}(\mathbf{x}_f) \quad (4.124)$$

$$= \frac{\|\mathbf{u}\|^2}{2} \left(\gamma(\gamma-1) \hat{\mathbf{u}}^T \frac{\partial^2 \phi(\mathbf{x}_s)}{\partial \mathbf{x}^2} \hat{\mathbf{u}} - \beta \|\mathbf{u}\| \sum_i \hat{v}_i \hat{\mathbf{u}}^T \frac{\partial^3 \phi(\mathbf{x}_s)}{\partial x_i \partial \mathbf{x}^2} \hat{\mathbf{u}} + \beta^2 \hat{\mathbf{v}}^T \frac{\partial^2 \phi(\mathbf{x}_s)}{\partial \mathbf{x}^2} \hat{\mathbf{v}} \right), \quad (4.125)$$

wobei $\hat{\mathbf{u}} = \frac{\mathbf{u}}{\|\mathbf{u}\|}$ und $\hat{\mathbf{v}} = \frac{\mathbf{v}}{\|\mathbf{v}\|}$. Anhand von (4.125) kann man feststellen, dass dieser Fehler am kleinsten ist, wenn $\beta = 0$ ist. D.h. wenn die Verbindungsline von $\phi(\mathbf{x}_o)$ und $\phi(\mathbf{x}_n)$ direkt durch den Flächenmittelpunkt \mathbf{x}_f verläuft. Die Diskretisierung des Divergenzoperators sowie des Gradienten wird auf die Interpolation von den Zellmittelpunkten auf die Flächenmittelpunkte zurückgeführt. D.h. hier ist die gleiche Abhängigkeit wie in (4.125) zu erwarten.

Die räumliche Diskretisierung des Laplace-Operators führt zu der Auswertung der Richtungsableitung

$$\int_V \Delta \phi(\mathbf{x}) d\mathbf{x} = \int_{\partial V} \nabla \phi(\mathbf{x}) \cdot \mathbf{n} d\mathbf{x}. \quad (4.126)$$

Dabei entsteht der Fehler

$$\nabla \phi(\mathbf{x}_f) \cdot \mathbf{n} - \nabla \tilde{\phi}(\mathbf{x}_f) \cdot \mathbf{n} \quad (4.127)$$

$$= -\frac{\|\mathbf{u}\|(2\gamma-1)}{2\cos(\alpha)} \hat{\mathbf{u}}^T \frac{\partial^2 \phi(\mathbf{x}_s)}{\partial \mathbf{x}^2} \hat{\mathbf{u}} \quad (4.128)$$

$$- \frac{\|\mathbf{u}\|^2((1-\gamma)^3 + \gamma^3)}{6\cos(\alpha)} \sum_i \hat{u}_i \hat{\mathbf{u}}^T \frac{\partial^3 \phi(\mathbf{x}_s)}{\partial x_i \partial \mathbf{x}^2} \hat{\mathbf{u}} \quad (4.129)$$

$$- \frac{\|\mathbf{u}\|^2 \sin(\alpha) \gamma (1-\gamma)}{2\cos(\alpha)} \sum_i \hat{w}_i \hat{\mathbf{u}}^T \frac{\partial^3 \phi(\mathbf{x}_s)}{\partial x_i \partial \mathbf{x}^2} \hat{\mathbf{u}}, \quad (4.130)$$

wobei $\hat{\mathbf{u}} = \frac{\mathbf{u}}{\|\mathbf{u}\|}$, $\hat{\mathbf{v}} = \frac{\mathbf{v}}{\|\mathbf{v}\|}$, $\mathbf{w} = \mathbf{n} - \frac{\hat{\mathbf{u}}}{\cos(\alpha)}$, $\hat{\mathbf{w}} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$. Außerdem gilt $\mathbf{w} \cdot \mathbf{n} = 0$. Anhand des ersten Summanden kann man erkennen, dass bei $\gamma \neq 0.5$ der Diskretisierungsfehler erster Ordnung entsteht. Umgekehrt fällt der Term $(2\gamma-1)$ weg, und man erhält die Approximation zweiter Ordnung. Die nicht orthogonale Gitter führen dazu, dass $|\cos(\alpha)| < 1$.

4.6.5 Motivation zur Verwendung der Centroidal-Voronoi-Gitter

In diesem Unterabschnitt wird gezeigt, dass die räumliche Diskretisierung mittels FVM auf den Centroidal-Voronoi-Gittern die kleinsten Approximationsfehler unter allen möglichen Gittern besitzt. Wir überlegen uns eine alternative Art der Diskretisierung, bei der man die Generatoren als diskrete Stützpunkte nimmt. Bei homogener Zelldichte trifft das stets zu, da die Generatoren mit den geometrischen Schwerpunkten übereinstimmen. Außerdem nehmen wir an, dass alle Voronoi-Gebiete konvex sind. In diesem Fall gelten folgende Aussagen:

- Die CV-Gitter sind absolut orthogonal, weil die Verbindungsline zwischen zwei Generatoren orthogonal zu der jeweiligen Fläche ist. D.h. es gilt für jedes Flächenelement

$$\cos(\alpha) = 1.$$

- Außerdem sind Centroidal-Voronoi-Gitter homogen, da für ein Generator \mathbf{x}_i per Definition $\|\mathbf{x} - \mathbf{x}_i\| < \|\mathbf{x} - \mathbf{x}_j\|$ gilt. Dabei sind \mathbf{x}_j alle restlichen Generatoren. Damit liegt \mathbf{x}_s auf dem Bisektor zwischen \mathbf{x}_n und \mathbf{x}_o . D.h. es gilt stets

$$\gamma = \frac{\|\mathbf{x}_s - \mathbf{x}_o\|}{\|\mathbf{x}_n - \mathbf{x}_o\|} = 0.5.$$

- Experimentell wurde in dieser Arbeit festgestellt, dass bei homogener Zelldichte alle Pyramiden einer inneren Centroidal-Voronoi-Zelle regulär sind. D.h. alle Kanten der Mantelfläche sind gleich lang. Damit muss $\|\mathbf{v}\| = 0$ und folglich auch

$$\beta = 0.$$

Wir haben festgestellt, welche Vorteile es mit sich bringt, bei CV-Gittern die Generatoren als Stützpunkte zu verwenden. Der größte Nachteil davon wäre, dass die Approximation der Volumenintegrale durch (4.79) den Abbruchsfehler erster Ordnung haben würde. Man könnte jedoch durch die Verwendung einer entsprechenden Quadraturformel auch eine höhere Approximationsordnung der Volumenintegrale erreichen. Dies würde jedoch einen wesentlich höheren Rechenaufwand verursachen.

Bei der Verwendung der geometrischen Schwerpunkte als Stützpunkte sind CV-Gitter immerhin optimal, da diese Diagramme eine Art Gleichgewicht zwischen dem Erfüllen der ZelldichteVerteilung und der isotropen Verteilung der Generatoren im Raum erzielen. Das bedeutet der Abstand zwischen dem geometrischen Schwerpunkt und dem mit der Zelldichte gewichteten Schwerpunkt (=Generator) ist minimal. D.h. bei einer vorgegebenen ZelldichteVerteilung fallen bei CV-Diagrammen alle drei Netzqualitätskriterien optimal aus.

4.7 Berechnung der zulässigen Netzflüsse

Dieser Abschnitt stellt einen nicht unwesentlichen Teil des wissenschaftlichen Beitrags dieser Arbeit dar. Hier wird beschrieben, wie die Netzflüsse bestimmt werden können, damit *Volume Conservation Law* erfüllt wird. Im diskreten Fall muss die Gleichung in (4.88) erfüllt sein. Der im Rahmen dieser Arbeit entwickelte Algorithmus zur Gitterbewegung führt zu einer topologischen Änderung der Flächenelemente. Dadurch gibt es keine Zuordnung der Knoten zwischen dem alten und dem neuen Gittern. Daher ist es unmöglich die Gittergeschwindigkeiten an den Flächenelementen aus Knotenverschiebungen zu rekonstruieren, wie das klassischerweise gemacht wird (siehe die Literatur [22, 28, 31, 32, 35]).

4.7.1 Korrektur der geschätzten Netzflüsse

Wir gehen davon aus, dass es eine Schätzung für die Netzflüsse gibt, die beispielsweise durch eine Interpolation der Geschwindigkeiten der Zellmittelpunkte auf die Flächenmittelpunkte bestimmt wurde. Weiterhin nehmen wir an, dass die Netzflüsse der Randflächen gegeben sind, da die Randverschiebungen durch den Anwender definiert werden sollten. Damit lautet die Aufgabenstellung: Wir suchen solche Netzflüsse

$$M_{ij}^k = \mathbf{W}_{ij}^k \cdot \mathbf{n}_{ij}^k A_{ij}^k, \quad (4.131)$$

dass die Gleichungen

$$\frac{V_i^k - V_i^{k-1}}{\Delta t} = \sum_{j \in \mathcal{N}_i} M_{ij}^k + \sum_{j \in \mathcal{B}_i} M_{ij}^k \quad \text{für } i = 1, 2, \dots, n \quad (4.132)$$

erfüllt sind. Dabei ist n die Anzahl der Volumenelente und \mathcal{N}_i bzw. \mathcal{B}_i ist die Menge der Indizes der Nachbarn bzw. der Randflächen des i -ten Elements. Die Indizierung der Randflächen fängt dabei mit n an, d.h. wenn bei einem M_{ij}^k $j > n$ ist, dann handelt es sich bei M_{ij}^k um einen Randnetzfluss. V_i^{k-1} bzw. V_i^k ist das alte bzw. das neue Volumen des i -ten Elements. Δt ist die verstrichene Zeit. Die Normale \mathbf{n}_{ij}^k zeigt vom V_i zum V_j und \mathbf{n}_{ji}^k zeigt wiederum in die umgekehrte Richtung, d.h. es gelten

$$\mathbf{n}_{ij}^k = -\mathbf{n}_{ji}^k \quad (4.133)$$

$$M_{ij}^k = -M_{ji}^k. \quad (4.134)$$

Wir führen globale Netzflüsse F_l^k und globale Flächennormalen \mathbf{N}_l^k ein, die für alle inneren Flächenelemente einheitlich definiert sind. Sei $m \in \mathbb{N}$ die Anzahl der internen Flächen, dann gilt für $l = 1, \dots, m$

$$F_l^k = F_{ij}^k = F_{ji}^k = \begin{cases} M_{ij}^k & i < j \\ -M_{ij}^k & \text{sonst} \end{cases} \quad (4.135)$$

$$\mathbf{N}_l^k = \mathbf{N}_{ij}^k = \mathbf{N}_{ji}^k = \begin{cases} \mathbf{n}_{ij}^k & i < j \\ -\mathbf{n}_{ij}^k & \text{sonst.} \end{cases} \quad (4.136)$$

Angenommen es gibt eine Näherung $\mathbf{f} \in \mathbb{R}^m$ für F_l^k , so dass (4.132) noch nicht erfüllt ist. Sei $\mathbf{y} \in \mathbb{R}^m$ der Vektor mit allen F_l^k , die der Gleichung (4.132) genügen. Seien außerdem $\mathbf{C} \in \mathbb{R}^{n \times m}$ und $\mathbf{c} \in \mathbb{R}^n$, so dass gilt

$$\mathbf{C}\mathbf{y} = \mathbf{c}. \quad (4.137)$$

D.h. die i -te Zeile von \mathbf{C} bzw. \mathbf{c} enthält eine Bedingung aus (4.132). Dabei gilt

$$C_{ij} = \begin{cases} 1 & j\text{-te Fläche gehört zu der }i\text{-ten Zelle, } \mathbf{N}_j^k \text{ zeigt nach außen} \\ -1 & j\text{-te Fläche gehört zu der }i\text{-ten Zelle, } \mathbf{N}_j^k \text{ zeigt nach innen} \\ 0 & \text{sonst} \end{cases} \quad (4.138)$$

$$c_i = \frac{V_i^k - V_i^{k-1}}{\Delta t} - \sum_{j \in \mathcal{B}_i} M_{ij}^k. \quad (4.139)$$

Offensichtlich gibt es unendlich viele Lösungen \mathbf{y} von (4.137). Wir suchen ein \mathbf{y} , das sowohl den kleinsten euklidischen Abstand zu der Näherung \mathbf{f} als auch die Lösung von (4.137) ist. Damit muss

$$\|\mathbf{y} - \mathbf{f}\|^2 \rightarrow \min \quad (4.140)$$

$$\mathbf{C}\mathbf{y} = \mathbf{c}. \quad (4.141)$$

gelten. Die Funktion in (4.140) kann wie folgt umgeformt werden.

$$\|\mathbf{y} - \mathbf{f}\|^2 = (\mathbf{y} - \mathbf{f})^T(\mathbf{y} - \mathbf{f}) = \mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T\mathbf{f} + \mathbf{f}^T\mathbf{f} \quad (4.142)$$

Da solche Operationen wie die Addition einer Konstante und die Multiplikation mit einer Konstante die Minimalstelle einer Funktion nicht verändern, erhalten wir mit (4.142) die zu (4.140)–(4.141) äquivalente Minimierungsaufgabe

$$\frac{1}{2}\mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{f} \rightarrow \min \quad (4.143)$$

$$\mathbf{C}\mathbf{y} = \mathbf{c}. \quad (4.144)$$

Kapitel 4. Nutzung des Verfahrens mit der FVM

In Anlehnung an [82] ist die Aufgabe (4.143)-(4.144) identisch zu

$$\mathbf{y} + \mathbf{C}^T \mathbf{w} = \mathbf{f} \quad (4.145)$$

$$\mathbf{C} \mathbf{y} = \mathbf{c}, \quad (4.146)$$

wobei $\mathbf{w} \in \mathbb{R}^n$ der zugehörige Lagrange-Multiplikator ist. Man setzt (4.145) in (4.146) ein und es ergibt sich

$$\mathbf{C}(\mathbf{f} - \mathbf{C}^T \mathbf{w}) = \mathbf{c} \quad \Rightarrow \quad \mathbf{C}\mathbf{C}^T \mathbf{w} = \mathbf{C}\mathbf{f} - \mathbf{c}. \quad (4.147)$$

Mit (4.145) und (4.147) erhält man die Lösung für \mathbf{y} durch

$$\mathbf{C}\mathbf{C}^T \mathbf{w} = \mathbf{C}\mathbf{f} - \mathbf{c} \quad (4.148)$$

$$\mathbf{y} = \mathbf{f} - \mathbf{C}^T \mathbf{w}. \quad (4.149)$$

Nun betrachten wir die Struktur der Matrix $\mathbf{B} := \mathbf{C}\mathbf{C}^T$. Für die ij -Komponente von \mathbf{B} gilt

$$B_{ij} = \sum_{k=1}^m C_{ik} C_{jk}. \quad (4.150)$$

Laut Definition von \mathbf{C} muss folgendes gelten:

$$B_{ij} = \begin{cases} -1 & \text{falls } i\text{-te und } j\text{-te Zellen eine gemeinsame Fläche haben} \\ |\mathcal{N}_i| & \text{wenn } i = j \text{ ist, und } |\mathcal{N}_i| \text{ ist die Anzahl der Flächen der } i\text{-ten Zelle} \\ 0 & \text{sonst.} \end{cases} \quad (4.151)$$

Wir sehen, dass die Matrix \mathbf{B} dünnbesetzt ist. \mathbf{B} hat die gleiche Struktur wie die Matrix des Laplace-Operators. Außerdem sollte \mathbf{B} symmetrisch positiv definit (s.p.d.) sein, denn laut [84] ist jede Matrix der Form $\mathbf{C}\mathbf{C}^T$ s.p.d., wenn \mathbf{C} vollen Zeilenrang hat. D.h. das Gleichungssystem in (4.148) kann sehr effizient mithilfe von solchen iterativen Verfahren wie die Methode der Konjugierten-Gradienten gelöst werden.

Es stellt sich heraus, dass \mathbf{C} jedoch keinen vollen Zeilenrang hat, d.h. ihre Zeilen sind linear abhängig. Es ist dadurch bedingt, dass n Bedingungen der diskreten Form vom *Volume Conservation Law* (VCL) in (4.132) a priori linear abhängig sind. Dazu stellen wir uns ein Gitter vor, dass nur aus zwei Hexaedern Ω_0 und Ω_1 besteht. Es gibt eine einzelne interne Fläche, die Ω_0 und Ω_1 trennt, d.h. es gibt nur einen unbekannten Netzfluss. Auf der anderen Seite gibt es zwei Gleichungen aus VCL für jedes Volumenelement. Wir haben ein überbestimmtes Problem, bei dem offensichtlich die Gleichungen aus VCL linear abhängig sind, d.h. bei N Volumenelementen ist immer eine VCL-Bedingung überflüssig.

Das beschriebene Problem wird in dieser Arbeit dadurch gelöst, dass man die VCL-Bedingung der Zelle Ω_l mit dem größten Volumeninhalt ($=V_l$) dem Gleichungssystem in (4.137) entnimmt. Es wird vorausgesetzt, dass bei gegebenen Randnetzflüssen das VCL auch für das gesamte Rechengebiet erfüllt ist. Nach der Lösung von (4.148)-(4.149) ist das VCL für Ω_l bis auf den Fehler \mathcal{E}_i erfüllt (siehe das Lemma 4.1). Der Fehler \mathcal{E}_i setzt sich aus dem Rundungsfehler und aus dem Fehler in der VCL-Bedingung für das gesamte Rechengebiet zusammen. Die Wahl der Zelle mit dem größten Volumen V_l ist dadurch bedingt, dass \mathcal{E}_i für diese Zelle den kleinsten relativen Wert hat, und damit auch vernachlässigbar ist.

Lemma 4.1. Sei eine Tessellierung $\{\Omega_i\}_{i=1}^N$ eines abgeschlossenen Gebietes $\Omega \subset \mathbb{R}^3$ gegeben. Wir betrachten zwei Zustände dieser Tessellierung $\{\Omega_i^{k-1}\}_{i=1}^N$ und $\{\Omega_i^k\}_{i=1}^N$, wobei Ω_i^{k-1} bzw. Ω_i^k für jedes i zum Zeitpunkt t^{k-1} bzw. t^k vorliegt. Beim Übergang vom Zustand t^{k-1} zum Zustand t^k können sich bei jedem Element Ω_i sowohl seine Nachbarn als auch sein Volumen ändern. Seien die Netzflüsse der Randflächen der Tessellierung $\{\Omega_i\}_{i=1}^N$ vorgegeben. Sei auch das VCL wie in (4.132) sowohl für die Volumenelemente Ω_i mit $i = 1, \dots, N$ und $i \neq l$ als auch für das Gesamtgebiet erfüllt, d.h. es gilt

$$\frac{V_i^k - V_i^{k-1}}{\Delta t} = \sum_{j \in \mathcal{N}_i} M_{ij}^k + \sum_{j \in \mathcal{B}_i} M_{ij}^k \quad \text{für } i \in \{1, \dots, N\} \setminus l, \quad (4.152)$$

$$\frac{V^k - V^{k-1}}{\Delta t} = \sum_{j \in \mathcal{B}} M_{ij}^k, \quad (4.153)$$

wobei V_i den Volumina von Ω_i entsprechen und V^{k-1} bzw. V^k bezeichnet das Volumen des gesamten Gebiets Ω zum Zeitpunkt t^{k-1} bzw. t^k . \mathcal{B} ist die Indexmenge aller Randflächenelemente. Der Index i in (4.153) entspricht dem Volumenelement, das das Randflächenelement mit dem Index j enthält. Dann ist das VCL auch für Ω_l erfüllt, d.h. es gilt

$$\frac{V_l^k - V_l^{k-1}}{\Delta t} = \sum_{j \in \mathcal{N}_l} M_{lj}^k + \sum_{j \in \mathcal{B}_l} M_{lj}^k. \quad (4.154)$$

Beweis. Seien die Voraussetzungen wie in dem Lemma 4.1 erfüllt. Das VCL für das Gesamtgebiet lautet

$$\frac{V^k - V^{k-1}}{\Delta t} = \sum_{j \in \mathcal{B}} M_{ij}^k, \quad (4.155)$$

Da es sich bei $\{\Omega_i\}_{i=1}^N$ um eine Tessellierung handelt, setzt sich das Volumen V aus der Summe einzelner Volumina V_i . Damit erhält man aus (4.155)

$$\frac{1}{\Delta t} \left[V_l^k + \sum_{i=0, i \neq l}^N V_i^k - \left(V_l^{k-1} + \sum_{i=0, i \neq l}^N V_i^{k-1} \right) \right] = \sum_{j \in \mathcal{B}} M_{ij}^k. \quad (4.156)$$

Durch eine einfache Umformung erhalten wir aus (4.156)

$$\frac{V_l^k - V_l^{k-1}}{\Delta t} = \frac{1}{\Delta t} \left(\sum_{i=0, i \neq l}^N V_i^{k-1} - \sum_{i=0, i \neq l}^N V_i^k \right) + \sum_{j \in \mathcal{B}} M_{ij}^k. \quad (4.157)$$

Da die zwei Summen in (4.157) über die gleiche Indexmenge laufen, gilt

$$\frac{V_l^k - V_l^{k-1}}{\Delta t} = \sum_{i=0, i \neq l}^N \left(\frac{V_i^{k-1} - V_i^k}{\Delta t} \right) + \sum_{j \in \mathcal{B}} M_{ij}^k. \quad (4.158)$$

Ein Einsetzen von (4.152) in (4.158) liefert

$$\frac{V_l^k - V_l^{k-1}}{\Delta t} = \sum_{i=0, i \neq l}^N \left(\sum_{j \in \mathcal{N}_i} M_{ij}^k + \sum_{j \in \mathcal{B}_i} M_{ij}^k \right) + \sum_{j \in \mathcal{B}} M_{ij}^k. \quad (4.159)$$

Ein inneres Flächenelement gehört zu jeweils zwei Volumenelementen. Da $M_{ij}^k = -M_{ji}^k$, heben sich in (4.159) diese Terme für alle Flächenelemente außer der Flächenelemente von Ω_l auf. Damit erhalten wir aus (4.159)

$$\frac{V_l^k - V_l^{k-1}}{\Delta t} = \sum_{j \in \mathcal{N}_l} M_{lj}^k + \sum_{j \in \mathcal{B}_l} M_{lj}^k. \quad (4.160)$$

Die Gleichung (4.160) bestätigt die Annahme. \square

Das VCL für das Rechengebiet

Dem Lemma 4.1 kann man entnehmen, dass der Fehler in der VCL-Bedingung (4.153) für das ganze Rechengebiet sich direkt in der VCL-Bedingung der Zelle mit dem größten Volumen V_l widerspiegelt. Es ist daher zwingend notwendig, dass die in jedem Zeitschritt verwendeten Randnetzflüsse der Gleichung in (4.153) genügen. Wie im Abschnitt 4.5 bereits erwähnt wurde, führt die Verletzung vom VCL zur Störung in der Massenbilanz. Das beschriebene Phänomen spielt eine entscheidende Rolle bei den Strömungssimulationen mit inkompressiblen Fluiden, bei denen schon die kleinsten Störungen in der Massenbilanz zu großen Druckschwankungen führen können.

In [27] wurde ein Algorithmus vorgestellt, mit dem die VCL erfüllende Netzflüsse berechnet werden können, die durch die im Raum verschobene Vielecke zurückgelegt werden. Da die Vielecke der Randflächen im allgemeinen nicht eben sind, werden diese Vielecke in Dreiecke zerlegt. Bei der Berechnung der Volumina von Polyedern werden die vieleckigen Flächenelemente auch in Dreiecke zerlegt. Um (4.153) zu erfüllen, muss bei den beiden Verfahren dieselbe Zerlegungsvorschrift gewählt werden. D.h. die Transformation eines Dreiecks muss der Volumenänderung des dieses Dreiecks enthaltenden Tetraeders entsprechen. Denn die exakte Form eines vieleckigen nicht ebenen Flächenelements ist erst durch eine Zerlegung in Dreiecke gegeben.

In der Praxis hat es sich herausgestellt, dass auch bei der Verwendung des in [27] beschriebenen Algorithmus kleine Abweichung der VCL-Bedingung in (4.153) entsteht. Die Ursache dafür liegt im Aufsummieren der Rundungsfehler. Zur Lösung dieser Problematik bietet sich folgende Strategie an. Es können entweder die Netzflüsse oder die Volumina der Randzellen angepasst werden. Bei einer vorgegebenen Volumenänderung kann der Fehler in (4.153) auf alle Randnetzflüsse abhängig von ihrem Betrag verteilt werden. Die im Raum fixierten Randflächenelemente müssen jedoch aus dieser Berechnung ausgenommen werden, weil ihre Netzflüsse gleich 0 bleiben müssen.

Stabilisierung des Algorithmus

Bei der Bewertung einiger Proberechnungen hat es sich herausgestellt, dass der durch (4.148)-(4.149) definierte Algorithmus bei den Gittern mit stark variierender Zelldichte instabil ist. Hierbei können die Rundungsfehler zu vollkommen unphysikalischen Netzflüssen führen. Dieses Problem kann wie folgt behoben werden. Zu den bereits vorhandenen Bezeichnungen definieren wir eine mit den Flächeninhalten skalierte Matrix \tilde{C} , sodass

$$\tilde{C}_{ij} = \begin{cases} A_{ij} & j\text{-te Fläche gehört zu der } i\text{-ten Zelle, } \mathbf{N}_j^k \text{ zeigt nach außen} \\ -A_{ij} & j\text{-te Fläche gehört zu der } i\text{-ten Zelle, } \mathbf{N}_j^k \text{ zeigt nach innen} \\ 0 & \text{sonst} \end{cases} \quad (4.161)$$

gilt. Außerdem soll auch $\mathbf{A} \in \mathbb{R}^{m \times m}$ eine Diagonalmatrix sein, deren Elemente den Flächeninhalten entsprechen. Weiterhin sei $\tilde{\mathbf{f}} \in \mathbb{R}^m$ ein Vektor, der alle geschätzten $\mathbf{W}_{ij}^k \cdot \mathbf{N}_{ij}^k$ enthält. D.h. es gilt

$$\mathbf{f} = \mathbf{A}\tilde{\mathbf{f}}. \quad (4.162)$$

Mit den eingeführten Bezeichnungen kann der Vektor \mathbf{y} mit korrigierten Netzflüssen durch

$$\tilde{\mathbf{C}}\tilde{\mathbf{C}}^T\tilde{\mathbf{w}} = \tilde{\mathbf{C}}\tilde{\mathbf{f}} - \mathbf{c} = \mathbf{C}\mathbf{f} - \mathbf{c} \quad (4.163)$$

$$\mathbf{y} = \mathbf{A} \left(\tilde{\mathbf{f}} - \tilde{\mathbf{C}}^T \tilde{\mathbf{w}} \right) = \mathbf{f} - \mathbf{A}\tilde{\mathbf{C}}^T\tilde{\mathbf{w}}. \quad (4.164)$$

berechnet werden. Der durch (4.163)-(4.164) gewonnene Lagrange-Multiplikator $\tilde{\mathbf{w}}$ bezieht sich nicht auf die mit Flächen skalierten Netzflüsse $M_{ij}^k = \mathbf{W}_{ij}^k \cdot \mathbf{n}_{ij}^k A_{ij}^k$ sondern auf die Normalgeschwindigkeiten $\mathbf{W}_{ij}^k \cdot \mathbf{n}_{ij}^k$. Die bei der Berechnung von $\tilde{\mathbf{w}}$ entstehenden Rundungsfehler beziehen sich damit auch auf $\mathbf{W}_{ij}^k \cdot \mathbf{n}_{ij}^k$. Es ist damit ausgeschlossen, dass die Netzflüsse kleiner Flächen stark durch die Rundungsfehler des Gesamtproblems (4.148) beeinflusst werden. Mit anderen Worten ist die Stabilität des durch (4.163)-(4.164) definierten Algorithmus unabhängig vom darunterliegenden Gitter.

Hintergründe zur Aufstellung der Matrix \mathbf{CC}^T in (4.148)-(4.149)

Beim Leser könnte die Frage auftreten, warum das Problem in (4.140)-(4.141) nicht direkt als lineares Least-Square-Problem mit linearen Restriktionen gelöst wird. Denn die Multiplikation der Matrizen \mathbf{C} und \mathbf{C}^T in (4.148) führt zur Quadrierung der Konditionszahl der Matrix \mathbf{C}^T , d.h. $\kappa_2(\mathbf{CC}^T) = (\kappa_2(\mathbf{C}^T))^2$ (siehe [84]). Im weiteren betrachten wir Möglichkeiten zur Lösung des Problems in (4.140)-(4.141) und erläutern, warum wir uns für die Aufstellung der Matrix \mathbf{CC}^T in (4.148)-(4.149) entschieden haben. In den folgenden Untersuchungen wird das im Kapitel 5 beschriebene Modell des Propellers verwendet. Die betreffende Matrix \mathbf{C}^T zum Zeitpunkt $t = 0$ besitzt 3647406 Zeilen, 579971 Spalten und 7294796 Nichtnulleinträge. Alle Rechnungen wurden auf dem Prozessor „Intel(R) Core(TM) i5-3450 CPU @ 3.10GHz“ durchgeführt. Wenn es nicht extra angegeben wird, wurde die Rechnung nur auf einem Thread durchgeführt.

- Einer der ersten Ansätze wäre die sogenannte QR-Zerlegung der Matrix \mathbf{C}^T in (4.148) (siehe [7, 84]). Damit könnte das Problem in (4.148) auf

$$\mathbf{CC}^T\mathbf{w} = (\mathbf{QR})^T\mathbf{QR}\mathbf{w} = \mathbf{R}^T\mathbf{Q}^T\mathbf{QR}\mathbf{w} = \mathbf{R}^T\mathbf{R}\mathbf{w} = \mathbf{C}\mathbf{f} - \mathbf{c} \quad (4.165)$$

zurückgeführt werden, bei dem die orthogonale Matrix \mathbf{Q} nicht benötigt wird. Zur Berechnung der QR-Zerlegung der Matrix \mathbf{C}^T wurde die kommerzielle Software MATLAB eingesetzt. Um den sogenannten Fill-In-Effekt zu reduzieren, wurde die frei verfügbare Bibliothek „METIS - Serial Graph Partitioning and Fill-reducing Matrix Ordering“ verwendet (siehe [1]). Mit dem Permutationsvektor \mathbf{p} wurde die Matrix \mathbf{R} in Matlab durch den Aufruf

```
tic; R = qr(C'(:,p), 0); toc
```

berechnet. Dieser Rechenschritt hat ca. 212 Sekunden gedauert. Dabei wurden alle 4 Prozessorkerne vollständig ausgelastet. D.h. auf einem Prozessorkern würde die Rechnung etwa 800 Sekunden dauern. Die Cholesky-Zerlegung der Matrix \mathbf{CC}^T

```
tic; B = C * C'; R = chol(B(p,p)); toc
```

hat hierbei nur 23 Sekunden gedauert. D.h. nur ein Zehntel der Rechenzeit für die QR-Zerlegung. Dieser positive Effekt ist darauf zurückzuführen, dass beim Produkt \mathbf{CC}^T

beinahe kein Fill-In entsteht. Die Matrix $\mathbf{C}\mathbf{C}^T$ enthält lediglich 7874751 Nichtnulleinträge. Obwohl die Konditionszahl der Matrix $\mathbf{C}\mathbf{C}^T$ $3.4828e+10$ beträgt, löst die Methode der Konjugierten Gradienten in Kombination mit dem Präkonditionier unvollständige Cholesky-Zerlegung das gleiche Problem in 5 Sekunden und braucht dabei 123 Iterationen, um die absolute Toleranz 10^{-12} zu erreichen. Die hohe Rechenzeit beim direkten Lösen des Least-Square-Problems ist auf den Fill-In-Effekt zurückzuführen. Die Matrix \mathbf{R} enthält bei der QR-Zerlegung 239227332 Nichtnulleinträge, was ca. dem 30-fachen Specheraufwand der ursprünglichen Matrix \mathbf{C}^T entspricht.

- Andererseits gibt es auch wissenschaftliche Arbeiten [14, 16, 40, 46, 53], die sich mit iterativen Methoden zur Lösung von linearen Least-Square-Problemen beschäftigen. Diese Verfahren basieren jedoch auf den klassischen iterativen Methoden wie CG, MINRES oder GMRES angewandt auf die Normalengleichung. D.h. ein Problem des Typs $\min_x \|\mathbf{b} - \mathbf{Ax}\|_2$ wird als $\mathbf{A}^T \mathbf{Ax} = \mathbf{A}^T \mathbf{b}$ gelöst. Das Ziel der meisten wissenschaftlichen Arbeiten ist die Verbesserung der Konvergenzeigenschaften der iterativen Löser. Bei einer höheren Konditionszahl der Matrix $\mathbf{C}\mathbf{C}^T$ könnte man auf diese Verfahren zurückgreifen. Während der ganzen transienten Simulation des im Kapitel 5 beschriebenen Propellers hat die PCG-Methode bei der Lösung des Problems (4.163)-(4.164) nicht mehr als 150 Iterationen benötigt, um die absolute Toleranz 10^{-12} zu erreichen. Das lässt darauf schließen, dass die Konditionszahl der entsprechenden Matrix $\mathbf{C}\mathbf{C}^T$ in diesem Beobachtungsraum beinahe konstant war und damit höchstwahrscheinlich in der gleichen Größenordnung 10^{10} lag.
- Da der in dieser Arbeit entwickelte Code eine Erweiterung von OpenFOAM ist, galt es im Rahmen der Promotion festzustellen, ob die in OpenFOAM eingebauten Löser für lineare Gleichungssysteme für das Problem (4.140)-(4.141) verwendet werden können. Zum einen werden diese Löser auf verteilten Systemen mittels OpenMPI (Message-Passing-Interface) ausgeführt und besitzen meist sehr gute Skalierbarkeit. D.h. beispielsweise bei Verdopplung der Anzahl der Rechenprozesse erhält man beinahe halbierte Rechenzeiten. Zum anderen bietet OpenFOAM das sogenannte Mehrgitterverfahren zur Lösung von linearen Gleichungssystemen an. Beim untersuchten Problem brauchte dieses Verfahren 37 Iterationen, um die zweite Norm des Residuums 10^{-12} zu erreichen. Die Rechenzeit betrug dabei ca. 5 Sekunden. Im Vergleich zur präkonditionierten CG-Methode stellt dies keine wirkliche Verbesserung dar. Ein großer Vorteil des in OpenFOAM eingebauten Mehrgitterverfahrens ist jedoch seine geringere Abhängigkeit der Konvergenzgeschwindigkeit von der Konditionszahl der betreffenden Matrix. Dabei ist das eine rein auf der Erfahrung basierende Aussage.

4.7.2 Eine gute Näherung der Netzflüsse

In dem Unterabschnitt 4.7.1 wurde eine Methode beschrieben, mit deren Hilfe jede Näherung der Netzflüsse so modifiziert werden kann, dass *Volume Conservation Law* erfüllt wird. In diesem Kontext wäre es auf den ersten Blick nicht verkehrt, die Geschwindigkeiten der Zellmittelpunkte (geometrische Schwerpunkte) auf die Flächenmittelpunkte zu interpolieren, deren Skalarprodukt mit $\mathbf{n}_{ij} A_{ij}$ die Netzflüsse liefern würde. Es hat sich jedoch in der Praxis herausgestellt, dass diese Methode künstliche Wirbeln des modellierten Fluids verursacht. Dies hat den folgenden Grund.

Die durch die beschriebene Methode gewonnenen Netzflüsse führen zu großen Abweichung von VCL. Die in dem Unterabschnitt 4.7.1 vorgestellte Korrektur erzeugt viel zu große Änderungen dieser geschätzten Netzflüsse. Da die Netzflüsse direkt in die Impulsgleichung eingehen, hat dies die zusätzlichen und unnatürlichen Geschwindigkeiten des Fluids zur Folge. Es ist damit wichtig eine Näherung der Netzflüsse zu finden, die sehr "geringfügig" das VCL verletzt.

In [79, 80] wurde eine Methode vorgestellt, mit deren Hilfe die Geschwindigkeiten der Flächenschwerpunkte in Richtung der Flächennormale aus den Geschwindigkeiten der Generatoren ermittelt werden können. In den angegebenen Quellen wird es auch behauptet, dass die dadurch errechneten Werte exakt sind. Dabei ist die mathematische Herkunft der meisten vorgestellten Gleichungen nicht zu finden. Es wird hier eine entsprechende Herleitung angeboten und gezeigt, dass die durch die angebotene Methode erhaltenen Geschwindigkeiten der Flächenschwerpunkte lediglich als eine Näherung verwendet werden können.

In diesem Unterabschnitt gelten abweichende Bezeichnungen. \mathbf{x}_i ist ein Generator und \mathbf{x}_j ist einer seiner Nachbarn. Sei ferner \mathbf{x}_f der geometrische Schwerpunkt der Trennfläche ∂V_{ij} von \mathbf{x}_i und \mathbf{x}_j . Wir suchen die Geschwindigkeit ω_f von \mathbf{x}_f , denn laut (4.83) gilt

$$\int_{\partial V_{ij}} \boldsymbol{\omega}(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x}) d\mathbf{x} = \boldsymbol{\omega}(\mathbf{x}_f) \cdot \mathbf{n}(\mathbf{x}_f) A_{ij} + \int_{\partial V_{ij}} \mathcal{O}(\|\mathbf{x} - \mathbf{x}_f\|^2) d\mathbf{x} \approx \boldsymbol{\omega}_f \cdot \mathbf{n}_{ij} A_{ij}. \quad (4.166)$$

D.h. der Abbruchfehler in (4.166) hat die Ordnung 2. Da die Geschwindigkeit die zeitliche Ableitung der Verschiebung ist, sind diese Größen proportional. Wir zeigen hier, wie man die Verschiebung des Flächenmittelpunkts berechnen kann. Dabei nehmen wir an, dass diese Verschiebung infinitesimal klein ist. D.h. $\Delta t \rightarrow 0$.

Seien damit \mathbf{u}_i und \mathbf{u}_j die Verschiebungen der Generatoren \mathbf{x}_i und \mathbf{x}_j . Wir suchen also die Verschiebung \mathbf{u}_m des Flächenmittelpunkts \mathbf{x}_f . Diese Größe kann man in zwei Komponenten zerlegen. Das sind die Verschiebung \mathbf{u}_m des Punktes

$$\mathbf{x}_m = \frac{\mathbf{x}_i + \mathbf{x}_j}{2}. \quad (4.167)$$

und die Rotation \mathbf{u}_r um den Punkt \mathbf{x}_m . Es gilt also

$$\mathbf{u}_f = \mathbf{u}_m + \mathbf{u}_r. \quad (4.168)$$

Seien $\tilde{\mathbf{x}}_i$, $\tilde{\mathbf{x}}_j$, $\tilde{\mathbf{x}}_m$ die Positionen der Punkte \mathbf{x}_i , \mathbf{x}_j , \mathbf{x}_m nach der Verschiebung durch \mathbf{u}_i und \mathbf{u}_j . Die Hervorhebung (\sim) soll im weiteren Verlauf die transformierten (verschobenen) Größen betonen. Mit (4.167) erhält man damit

$$\mathbf{u}_m = \tilde{\mathbf{x}}_m - \mathbf{x}_m = \frac{\tilde{\mathbf{x}}_i + \tilde{\mathbf{x}}_j}{2} - \frac{\mathbf{x}_i + \mathbf{x}_j}{2} = \frac{(\tilde{\mathbf{x}}_i - \mathbf{x}_i) + (\tilde{\mathbf{x}}_j - \mathbf{x}_j)}{2} = \frac{\mathbf{u}_i + \mathbf{u}_j}{2}. \quad (4.169)$$

Um die Rotationskomponente \mathbf{u}_r zu berechnen, müssen wir laut (4.168) die Verschiebung $\mathbf{u}_m = \tilde{\mathbf{x}}_m - \mathbf{x}_m$ von allen verschobenen Größen wie $\tilde{\mathbf{x}}_m$, $\tilde{\mathbf{x}}_i$ und $\tilde{\mathbf{x}}_j$ sowie von \mathbf{u}_i , \mathbf{u}_j abziehen. Wir erhalten damit die folgenden Werte:

$$\hat{\mathbf{x}}_m = \tilde{\mathbf{x}}_m - (\tilde{\mathbf{x}}_m - \mathbf{x}_m) = \mathbf{x}_m \quad (4.170)$$

$$\hat{\mathbf{u}}_i = \mathbf{u}_i - (\tilde{\mathbf{x}}_m - \mathbf{x}_m) = \tilde{\mathbf{x}}_i - \mathbf{x}_i - \left(\frac{\tilde{\mathbf{x}}_i + \tilde{\mathbf{x}}_j}{2} - \frac{\mathbf{x}_i + \mathbf{x}_j}{2} \right) \quad (4.171)$$

$$= \frac{2\tilde{\mathbf{x}}_i - 2\mathbf{x}_i - \tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j + \mathbf{x}_i + \mathbf{x}_j}{2} = \frac{\tilde{\mathbf{x}}_i - \mathbf{x}_i - \tilde{\mathbf{x}}_j + \mathbf{x}_j}{2} = \frac{\mathbf{u}_i - \mathbf{u}_j}{2} \quad (4.172)$$

$$\hat{\mathbf{u}}_j = \mathbf{u}_j - (\tilde{\mathbf{x}}_m - \mathbf{x}_m) = \tilde{\mathbf{x}}_j - \mathbf{x}_j - \left(\frac{\tilde{\mathbf{x}}_i + \tilde{\mathbf{x}}_j}{2} - \frac{\mathbf{x}_i + \mathbf{x}_j}{2} \right) \quad (4.173)$$

$$= \frac{2\tilde{\mathbf{x}}_j - 2\mathbf{x}_j - \tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j + \mathbf{x}_i + \mathbf{x}_j}{2} = \frac{\tilde{\mathbf{x}}_j - \mathbf{x}_j - \tilde{\mathbf{x}}_i + \mathbf{x}_i}{2} = \frac{\mathbf{u}_j - \mathbf{u}_i}{2} \quad (4.174)$$

Aus (4.171)-(4.174) folgt, dass

$$\hat{u}_i = -\hat{u}_j. \quad (4.175)$$

Das Ergebnis der beschriebenen Transformation durch $(-\boldsymbol{u}_m)$ ist in der Abbildung 4.3 zu sehen. Die Punkte \boldsymbol{x}_m und $\hat{\boldsymbol{x}}_m$ sind jetzt gleich. Weitere Bezeichnungen sind der Abbildung 4.3 zu entnehmen. Es ist gut sichtbar, dass die zu \boldsymbol{x}_{ij} parallelen Komponenten \hat{u}_i^p und \hat{u}_j^p sich gegenseitig

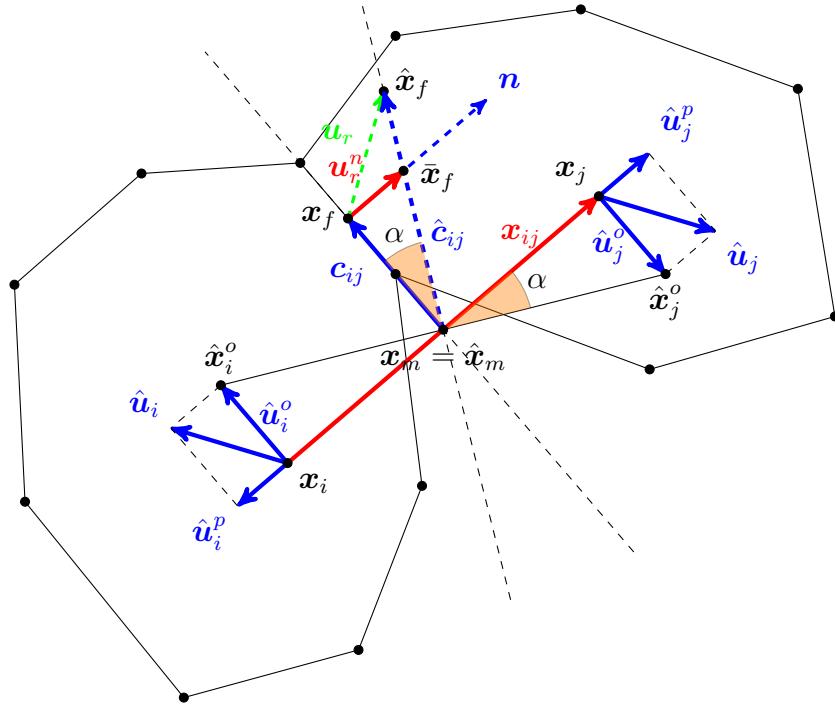


Abbildung 4.3: Berechnung der Rotationskomponente der Verschiebung. Die Punktbezeichnungen werden schwarz dargestellt. Die Vektoren werden in Farbe abgebildet.

aufheben, und somit keinen Beitrag zu \boldsymbol{u}_r leisten. D.h. wir interessieren uns nur für die zu \boldsymbol{x}_{ij} orthogonalen Komponenten \hat{u}_i^o und \hat{u}_j^o . Dadurch wird der Vektor \boldsymbol{c}_{ij} auch um den Winkel α gedreht, und wir erhalten $\hat{\boldsymbol{c}}_{ij}$. Der transformierte Flächenschwerpunkt $\hat{\boldsymbol{x}}_f$ muss nicht unbedingt auf der Normale \boldsymbol{n} liegen. Denn die Position von $\hat{\boldsymbol{x}}_f$ hängt nicht nur von \boldsymbol{x}_i bzw. \boldsymbol{x}_j sondern auch von den anderen Generatoren ab. Das hat sich auch in der Praxis bestätigt. Im allgemeinen befindet sich der verschobene Flächenschwerpunkt auf dem Bisektor zwischen $\hat{\boldsymbol{x}}_i^o$ und $\hat{\boldsymbol{x}}_j^o$. In der Abbildung 4.3 entspricht dieser Bisektor der Gerade durch den Vektor $\hat{\boldsymbol{c}}_{ij}$. Da bei der Berechnung des Netzflusses die Gittergeschwindigkeit auf die Flächennormale projiziert wird, können wir gleich hier davon Gebrauch machen. Wir bestimmen eine Näherung für die Projektion von \boldsymbol{u}_r auf \boldsymbol{n} durch

$$\boldsymbol{u}_r^n := (\boldsymbol{u}_r \cdot \boldsymbol{n}) \boldsymbol{n} \approx \bar{\boldsymbol{x}}_f - \boldsymbol{x}_f. \quad (4.176)$$

$\bar{\boldsymbol{x}}_f$ ist der Schnittpunkt des Bisektors zwischen $\hat{\boldsymbol{x}}_i^o$ und $\hat{\boldsymbol{x}}_j^o$ mit der Geraden durch die Flächennormale \boldsymbol{n} (siehe die Abbildung 4.3). Damit gilt

$$\tan(\alpha) = \frac{\|\boldsymbol{u}_r^n\|}{\|\boldsymbol{c}_{ij}\|}. \quad (4.177)$$

Da \boldsymbol{x}_{ij} und \boldsymbol{n} per Definition eines Voronoi-Gebietes parallel sind, müssen auch \boldsymbol{c}_{ij} und $\hat{\boldsymbol{u}}_j^o$ parallel sein. Damit erhalten wir

$$\hat{\boldsymbol{u}}_j^o = \left(\hat{\boldsymbol{u}}_j \cdot \frac{\boldsymbol{c}_{ij}}{\|\boldsymbol{c}_{ij}\|} \right) \frac{\boldsymbol{c}_{ij}}{\|\boldsymbol{c}_{ij}\|}. \quad (4.178)$$

Da die Dreiecke durch $(\mathbf{x}_m, \mathbf{x}_f, \bar{\mathbf{x}}_f)$ und $(\mathbf{x}_m, \mathbf{x}_j, \hat{\mathbf{x}}_j^o)$ ähnlich sind, gilt auch

$$\tan(\alpha) = \frac{\|\hat{\mathbf{u}}_j^o\|}{\|\mathbf{x}_{ij}\|/2} = 2 \frac{\left\| \left(\hat{\mathbf{u}}_j \cdot \frac{\mathbf{c}_{ij}}{\|\mathbf{c}_{ij}\|} \right) \frac{\mathbf{c}_{ij}}{\|\mathbf{c}_{ij}\|} \right\|}{\|\mathbf{x}_{ij}\|} = 2 \frac{|\hat{\mathbf{u}}_j \cdot \mathbf{c}_{ij}|}{\|\mathbf{x}_{ij}\| \|\mathbf{c}_{ij}\|} \quad (4.179)$$

Wir setzen (4.177) in (4.179) ein, und erhalten

$$\frac{\|\mathbf{u}_r^n\|}{\|\mathbf{c}_{ij}\|} = 2 \frac{|\hat{\mathbf{u}}_j \cdot \mathbf{c}_{ij}|}{\|\mathbf{x}_{ij}\| \|\mathbf{c}_{ij}\|} \Rightarrow \|\mathbf{u}_r^n\| = 2 \frac{|\hat{\mathbf{u}}_j \cdot \mathbf{c}_{ij}|}{\|\mathbf{x}_{ij}\|} \quad (4.180)$$

Um das Vorzeichen von $(\mathbf{u}_r \cdot \mathbf{n})$ zu bestimmen, machen wir Gebrauch von dem Wert $(\hat{\mathbf{u}}_j \cdot \mathbf{c}_{ij})$. Bei der in der Abbildung 4.3 dargestellten Situation ist

$$\mathbf{u}_r \cdot \mathbf{n} > 0, \text{ wobei } \hat{\mathbf{u}}_j \cdot \mathbf{c}_{ij} < 0. \quad (4.181)$$

Damit gilt also

$$\mathbf{u}_r^n = -2 \frac{\hat{\mathbf{u}}_j \cdot \mathbf{c}_{ij}}{\|\mathbf{x}_{ij}\|} \mathbf{n}. \quad (4.182)$$

In (4.173)-(4.174) wurde gezeigt, dass $\hat{\mathbf{u}}_j = \frac{\mathbf{u}_j - \mathbf{u}_i}{2}$ ist. Außerdem gilt auch $\mathbf{n} = \frac{\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|}$. Damit erhalten wir aus (4.182)

$$\mathbf{u}_r^n = \frac{(\mathbf{u}_i - \mathbf{u}_j) \cdot \mathbf{c}_{ij}}{\|\mathbf{x}_{ij}\|} \frac{\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|}. \quad (4.183)$$

Mit (4.168), (4.169) und (4.183) erhält man die zu \mathbf{n} parallele Gesamtverschiebung von \mathbf{x}_f durch

$$\mathbf{u}_f^n = \left(\frac{\mathbf{u}_i + \mathbf{u}_j}{2} \cdot \frac{\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|} + \frac{(\mathbf{u}_i - \mathbf{u}_j) \cdot \mathbf{c}_{ij}}{\|\mathbf{x}_{ij}\|} \right) \frac{\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|}. \quad (4.184)$$

Die Gültigkeit der Gleichung (4.184) hat eine Schranke

$$-\frac{\pi}{2} < \alpha < \frac{\pi}{2}, \quad (4.185)$$

wobei eine Rotation um $|\alpha| > \frac{\pi}{2}$ schwer vorstellbar ist. Bei der in dieser Arbeit entwickelten Methode zur Gitterbewegung wird diese Schranke nicht überschritten, da der Schwerpunkt (der neue Generator) einer Konvexen Zelle immer innerhalb dieser Zelle liegt. Dies sorgt dafür, dass (4.185) immer erfüllt ist (Siehe die Abbildung 4.3). Die angenäherte Geschwindigkeit ω_f^n des Flächenschwerpunktes \mathbf{x}_f in Richtung der Flächennormale \mathbf{n} kann durch die Ableitung von \mathbf{u}_f^n nach der Zeit bestimmt werden. Dabei werden die Größen \mathbf{x}_{ij} und \mathbf{c}_{ij} als konstant angesehen. Damit erhält man für ω_f^n

$$\omega_f^n = \left(\frac{\omega_i + \omega_j}{2} \cdot \frac{\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|} + \frac{(\omega_i - \omega_j) \cdot \mathbf{c}_{ij}}{\|\mathbf{x}_{ij}\|} \right) \frac{\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|}. \quad (4.186)$$

Dabei sind ω_i und ω_j die Geschwindigkeiten von \mathbf{x}_i und \mathbf{x}_j . Mit (4.186) nimmt das *Volume Conservation Law* (4.82) folgende Gestalt an:

$$\frac{d}{dt} \int_{V_i} d\mathbf{x} = \sum_{j \in \mathcal{N}_i \partial V_{ij}} \int \left(\left(\frac{\omega_i + \omega_j}{2} \cdot \frac{\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|} + \frac{(\omega_i - \omega_j) \cdot \mathbf{c}_{ij}}{\|\mathbf{x}_{ij}\|} \right) \frac{\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|} \right) \cdot \mathbf{n} d\mathbf{x} \quad (4.187)$$

$$= \sum_{j \in \mathcal{N}_i \partial V_{ij}} \int \left(\frac{\omega_i + \omega_j}{2} \cdot \frac{\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|} + \frac{(\omega_i - \omega_j) \cdot \mathbf{c}_{ij}}{\|\mathbf{x}_{ij}\|} \right) d\mathbf{x}. \quad (4.188)$$

Diskretisierung von (4.187)-(4.188) mit dem impliziten Euler-Verfahren führt zu

$$\frac{V_i^k - V_i^{k-1}}{\Delta t} \approx \sum_{j \in \mathcal{N}_i} \left(\frac{\omega_i^k + \omega_j^k}{2} \cdot \frac{\mathbf{x}_{ij}^k}{\|\mathbf{x}_{ij}^k\|} + \frac{(\omega_i^k - \omega_j^k) \cdot \mathbf{c}_{ij}^k}{\|\mathbf{x}_{ij}^k\|} \right) A_{ij}^k. \quad (4.189)$$

Der Approximationsfehler in (4.189) setzt sich aus folgenden Termen zusammen:

- Der Fehler durch die Näherung von ω_f^n ;
- Der Diskretisierungsfehler durch das implizite Euler-Verfahren. Dieser ist proportional zum Quadrat der Zeitschrittweite Δt ;
- Der quadratische Fehler bei der Approximation der Flächenintegrale $\int_{\partial V_{ij}}$ (siehe die Gleichung (4.166)).

Mit diesen Überlegungen können wir M_{ij}^k aus (4.189)

$$M_{ij}^k = \left(\frac{\omega_i^k + \omega_j^k}{2} \cdot \frac{\mathbf{x}_{ij}^k}{\|\mathbf{x}_{ij}^k\|} + \frac{(\omega_i^k - \omega_j^k) \cdot \mathbf{c}_{ij}^k}{\|\mathbf{x}_{ij}^k\|} \right) A_{ij}^k \quad (4.190)$$

als eine gute Schätzung für die Netzflüsse verwenden. Die in dem Unterabschnitt 4.7.1 beschriebene Korrektur führt dazu, dass auch die diskrete Form von *Volume Conservation Law* erfüllt ist.

4.8 Verteilung der Zelldichte bzw. der Kantenlänge

In diesem Abschnitt wird beschrieben, wie die Verteilung der in dem Kapitel 3 definierten Zelldichte berechnet werden kann. Die Zelldichte wird bei der Berechnung der gewichteten Schwerpunkte benötigt. Wir verwenden folgende Bezeichnungen: $\mathcal{L}(\mathbf{x})$ - die Kantenlänge; $\rho(\mathbf{x})$ - die Zelldichte. Einen Zusammenhang zwischen \mathcal{L} und ρ stellt die Gleichung (3.67) aus dem Unterabschnitt 3.2.3 her. Diese hat die Form

$$\rho(\mathbf{x}) = \frac{1}{(\mathcal{L}(\mathbf{x}))^5}. \quad (4.191)$$

Wir werden daher zuerst die Verteilung der Kantenlänge bestimmen. Die Zelldichte wird dann mithilfe von (4.191) berechnet.

Die in diesem Abschnitt beschriebene Methode basiert darauf, dass zu jedem Zeitpunkt ein Gitter vorhanden ist. In der Regelfall ist die Kantenlänge auf dem Rand des Modells durch den Benutzer vorgegeben. Da die Kantenlänge eine stetig differenzierbare Funktion sein muss, wird diese in dieser Arbeit als Lösung der stationären Diffusionsgleichung bestimmt. Denn die unter dem Laplace-Operator stehende Funktion muss zwingend zweifach stetig differenzierbar sein. Es ist die partielle Differentialgleichung

$$\nabla \cdot (\delta(\mathbf{x}) \nabla \mathcal{L}(\mathbf{x})) = 0 \quad (4.192)$$

zu lösen, wobei $\delta(\mathbf{x})$ der Diffusionskoeffizient ist. $\delta(\mathbf{x})$ dient dazu, die Änderungsrate der Elementgröße zu beeinflussen. Die Änderungsrate bezieht sich auf die Richtung vom Rand zum Inneren. Für die Lösung der Differentialgleichung in (4.192) verwenden wir die Finite-Volumen-Methode. Dabei wird die Gleichung auf dem Gitter diskretisiert, das von dem vorherigen Zeitschritt stammt. Bei der Lösung von (4.192) werden keine besonders hohen Anforderungen an

Genauigkeit gesetzt. Denn die Darstellung der Kantenlänge als die Lösung der Diffusionsgleichung lediglich ein von uns gewähltes Modell ist.

Daher können wir zur Approximation der Richtungsableitung die in (4.84) angegebene Vorschrift verwenden. Denn für allgemeine Gitter ist diese Näherung bis auf den Abbruchfehler erster Ordnung genau. Die Diskretisierung von (4.192) mithilfe von FVM führt damit zu

$$\int_{\partial V_i} \delta(\mathbf{x}) \nabla \mathcal{L}(\mathbf{x}) \cdot \mathbf{n} d\mathbf{x} = \sum_{j \in \mathcal{N}_i} \delta_{ij} \frac{\mathcal{L}(\mathbf{x}_j) - \mathcal{L}(\mathbf{x}_i)}{\|\mathbf{x}_j - \mathbf{x}_i\|} A_{ij} = 0 \quad \text{für } i = 1, 2, \dots, N, \quad (4.193)$$

wobei \mathcal{N}_i die Menge der Nachbarindizes des i -ten Elements ist. δ_{ij} sind die Werte von δ an den Flächenmittelpunkten zwischen \mathbf{x}_i und \mathbf{x}_j . Bei den Randzellen muss die Richtungsableitung ($\nabla \mathcal{L}(\mathbf{x}) \cdot \mathbf{n}$) auf eine abweichende Weise behandelt werden. Siehe dafür [37]. Nach der Lösung des Gleichungssystems in (4.193) erhalten wir die Kantenlänge für die Zellmittelpunkte. Um die Volumenintegrale in (3.69) und die Flächenintegrale in (3.100)-(3.101) auszuwerten, muss die Kantenlänge zusätzlich an den Flächenmittelpunkten und an den Eckpunkten des Gitters gegeben sein. Diese Werte werden durch lineare Interpolation bestimmt.

Es hat sich im Laufe der Arbeit herausgestellt, dass der Abstand zum nächsten Flächenmittelpunkt auf dem Gebietsrand ($=: B_d(\mathbf{x})$) zur Bestimmung von δ herangezogen werden kann. Wir verwenden die Gleichung

$$\delta(\mathbf{x}) := \frac{1}{(B_d(\mathbf{x}))^\kappa}, \quad (4.194)$$

um $\delta(\mathbf{x})$ zu berechnen. Dabei ist κ ein konstanter Exponent, mit dessen Hilfe ein zusätzlicher Einfluss auf die Änderungsrate der Kantenlänge gewonnen werden kann. Um diese Abhängigkeit zu untersuchen, wurde ein einfaches Modell erzeugt (siehe die Abbildung 4.4b). Hierbei wurden für zwei gegenüberliegende Ränder unterschiedliche Kantenlängen definiert, sodass deren Verhältnis 1 zu 10 ist. Die Abbildung 4.4a zeigt die Lösungen des Gleichungssystems in (4.193) bei verschiedenen κ . Bei der Wahl von κ sind folgende Fälle zu unterscheiden:

- $\kappa > 0$: Im Bereich des Randes wird $\delta(\mathbf{x})$ wesentlich höher als im Inneren sein, so dass die kleinen Elementgrößen vom Rand ins Innere besser propagieren beziehungsweise diffundieren kann. Für größere κ erhält man kleinere Wachstumsraten der Kantenlänge.
- $\kappa = 0$: Hierbei verschwindet der Diffusionskoeffizient $\delta(\mathbf{x})$, so dass die Lösung einer Wärmeverteilung im homogenen Medium entspricht. Der Graph der Kantenlänge von einem zum anderen Rand hat beinahe einen linearen Verlauf.
- $\kappa < 0$: Wird in dieser Arbeit nicht benutzt. Dies würde eine umgekehrte Wirkung haben, d.h. die kleinere Kantenlänge würde schneller als linear wachsen. Dies würde zu sehr verzerrten Gittern führen.

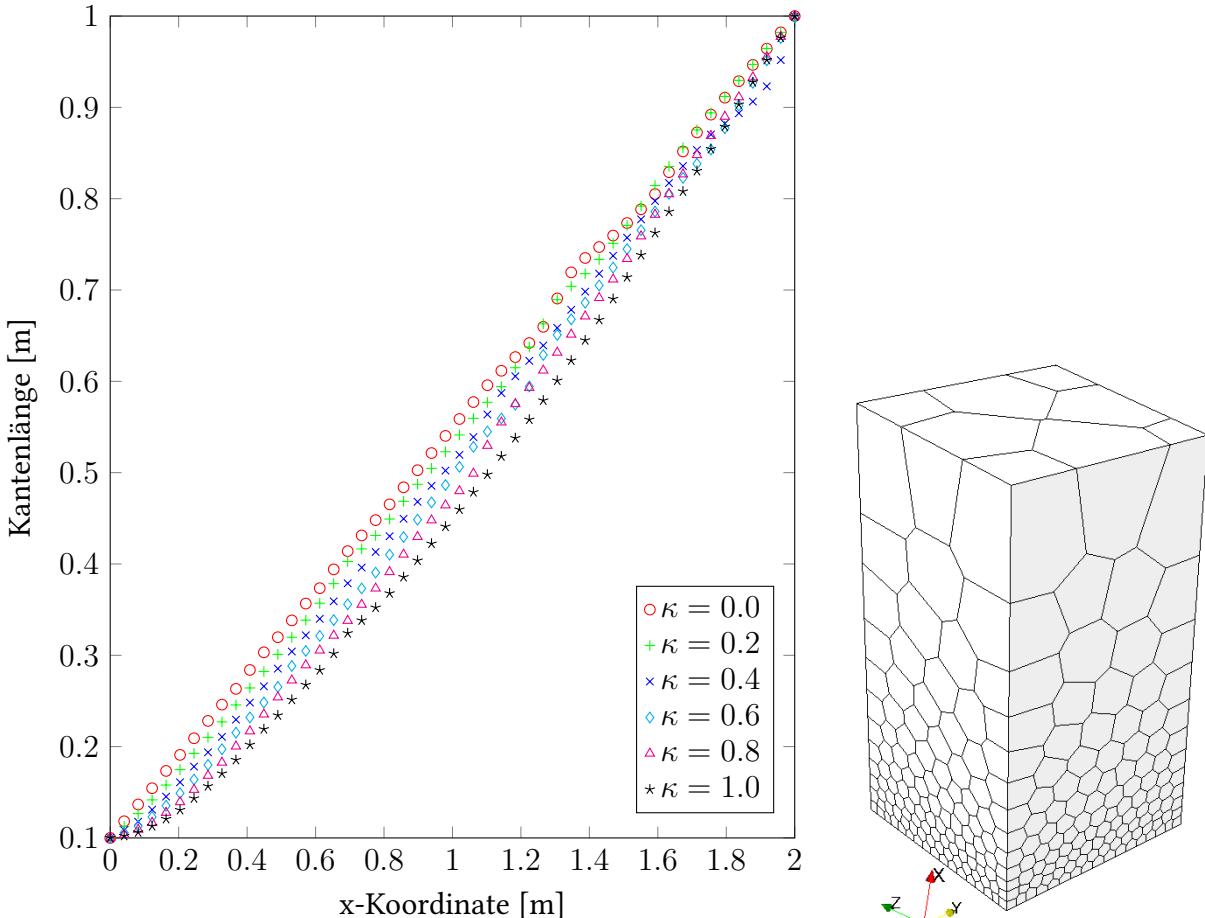
Demzufolge kann die Wachstumsrate der Kantenlänge mit einem einzigen Skalar κ beeinflusst werden.

Es ist notwendig, dass die Diffusionsgleichung (4.192) für die Kantenlänge gelöst wird, und nicht für die Zelldichte. Dies hat folgende Motivation. Wenn man die Lösung der Diffusionsgleichung mit einem verschwindenden Diffusionskoeffizienten entlang einer Normale zum Rand auswertet, stellt man fest, dass diese einen linearen Verlauf hat. Die Abbildung 4.4 bestätigt diese Aussage. Hätten wir die Diffusionsgleichung für die Zelldichte gelöst, würden wir für diese einen "linearen" Verlauf erhalten. Dies widerspricht der Definition der Zelldichte, die als Kehrwert der Funktion fünfter Ordnung bestimmt wird.

Aus dem gleichen Grund muss unbedingt die Kantenlänge und nicht die Zelldichte an den Flächenmittelpunkten und an den Eckpunkten des Gitters durch lineare Interpolation bestimmt werden. Anschließend folgt die Anwendung der Gleichung in (4.191). Die beschriebenen Schritte sind im Algorithmus 4.2 zusammengefasst.

Algorithmus 4.2 Berechnung der Verteilung der Zelldichte

1. Definiere die Kantenlänge für jedes Flächenelement auf dem Gebietsrand.
2. Berechne den Abstand zum nächstliegenden Rand für jeden Zellmittelpunkt: $B_d(\mathbf{x})$.
3. Interpoliere $B_d(\mathbf{x})$ von den Zellmittelpunkten auf die Flächenmittelpunkte.
4. Wähle ein entsprechendes κ und berechne: $\delta(\mathbf{x}) := \frac{1}{(B_d(\mathbf{x}))^\kappa}$.
5. Löse die Laplace-Gleichung $\nabla \cdot (\delta(\mathbf{x}) \nabla \mathcal{L}(\mathbf{x})) = 0$ für jeden Zellmittelpunkt.
6. Interpoliere $\mathcal{L}(\mathbf{x})$ von den Zellmittelpunkten auf die Flächenmittelpunkte und auf die Eckpunkte des Gitters.
7. Berechne $\rho(\mathbf{x}) = \frac{1}{(\mathcal{L}(\mathbf{x}))^5}$ für die Zellmittelpunkte, die Flächenmittelpunkte und die Eckpunkte des Gitters.



(a) Kantenlänge entlang der Gerade $(0.0, 0.5, 0.5)-(2.0, 0.5, 0.5)$. Der Startpunkt der Geraden liegt im Mittelpunkt der unteren Fläche des Modells (Abbildung 4.4b). Der Endpunkt der Geraden ist im Mittelpunkt der oberen Fläche.

(b) Das verwendete Modell.
x-Achse zeigt nach oben.
z-Achse zeigt nach links.

Abbildung 4.4: Abhängigkeit der Änderungsrate der Elementgröße vom κ .

5 Validierung in OpenFOAM

5.1 Integration in OpenFOAM

OpenFOAM ist ein frei verfügbares Paket von Strömungslösern, das sich durch seine stark verwurzelte Klassenstruktur sehr einfach erweitern lässt. Die hierbei verwendete Programmiersprache ist C++. Der entwickelte Netzbewegungslöser stellt dabei eine Klasse dar, die von der Parent-Klasse `dynamicFvMesh` abgeleitet wurde. Ein Kompilieren dieser Klasse liefert eine zusätzliche Bibliothek. Die in OpenFOAM eingebauten Löser wie `pimpleDyMFoam` bzw. `rhoPimpleDyMFoam` können in Zusammenspiel mit dieser Bibliothek nicht verwendet werden, da diese Löser von einer konstanten Topologie des darunterliegenden Gitters ausgehen. Um die entwickelte Bibliothek zu verwenden, muss jeder Löser umgeschrieben werden. Alle in dem betreffenden Löser auftretenden Felder `surfaceScalarField` bzw. `surfaceVectorField` müssen nach jedem Aktualisieren des Gitters neu initialisiert werden. Beim `pimpleDyMFoam` reicht es aus, die Massenflüsse

$$\Phi_{ij}^k = \mathbf{U}_{ij}^k \cdot \mathbf{n}_{ij}^k A_{ij}^k \quad (5.1)$$

neu zu berechnen. Siehe die Gleichung (4.90) im Kapitel 4. Die Netzflüsse

$$M_{ij}^k = \mathbf{W}_{ij}^k \cdot \mathbf{n}_{ij}^k A_{ij}^k \quad (\text{Gl. (4.89)}) \quad (5.2)$$

werden in der Klasse des Netzbewegungslösers berechnet. In `interDyMFoam` muss zusätzlich die Größe `rhoPhi` durch eine Interpolation aus `rho` neu berechnet werden. In der OpenFOAM-Schreibweise erhält man `rho` und `rhoPhi` durch

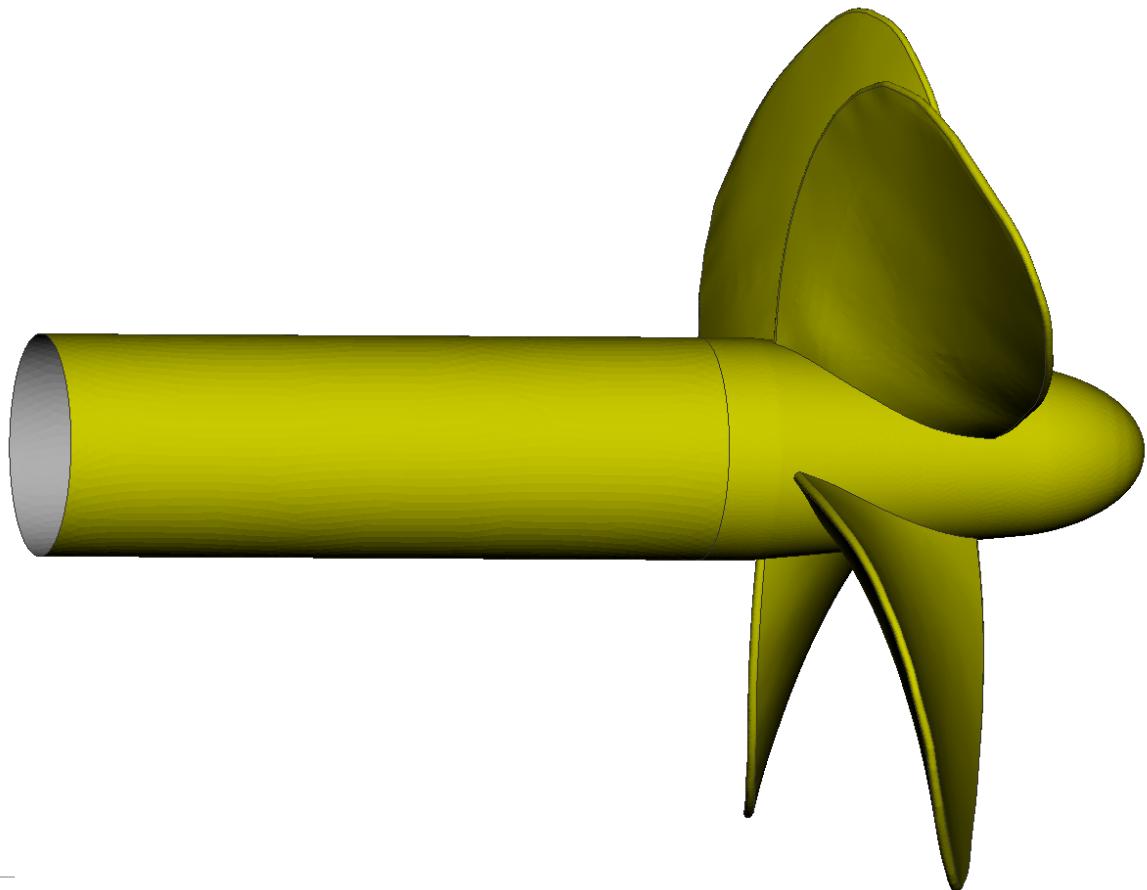
```
phi = fvc::interpolate(U, "cubic") & mesh.Sf();  
rhoPhi = fvc::interpolate(rho, "cubic") * phi;
```

In OpenFOAM gibt es Klassen, in denen Instanzen des Typs `surfaceScalarField` bzw. `surfaceVectorField` definiert werden, ohne diese zu der sogenannten `objectRegistry` hinzuzufügen. Dadurch sind diese Felder gewöhnlicherweise in den restlichen Klassen nicht sichtbar. Dadurch ist bei der Verwendung einer in OpenFOAM eingebauten Randbedingung, die weder Neumann-Randbedingung noch Dirichlet-Randbedingung ist, eine besondere Vorsicht geboten.

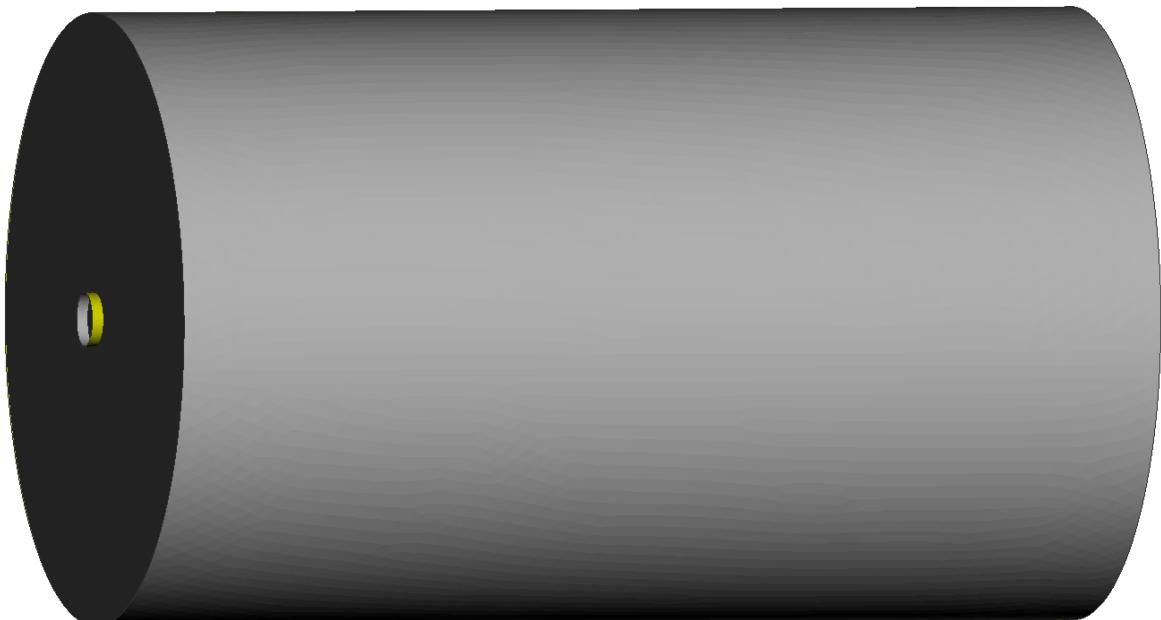
In einer unserer früheren Arbeiten [83] haben wir die Tauglichkeit des entwickelten Algorithmus zur Netzbewegung anhand einer Rechnung, die sich mit den experimentell gewonnenen Daten vergleichen lässt, bestätigt. Dabei wurde eine in einem viskosen Fluid sinkende Stahlkugel modelliert. Der Vorteil so einer Simulation liegt darin, dass deren Ergebnisse nicht nur mit den experimentellen Daten, sondern auch mit den Resultaten, die durch eine stationäre Rechnung gewonnen wurden, verglichen werden können. In beiden Fällen wurde eine sehr gute Übereinstimmung festgestellt.

5.2 Propeller im Wasser

In diesem Abschnitt werden wir uns mit einer Strömungssimulation beschäftigen, bei der ein Propeller im Wasser mit einer konstanten Winkelgeschwindigkeit $158 \frac{\text{rad}}{\text{s}}$ gedreht wird. Die Abbildung 5.1 zeigt die Randflächen der simulierten Geometrie. In der Beschriftung sind auch die



(a) STL-Oberfläche des Rotors



(b) STL-Oberfläche des Zylinders

Abbildung 5.1: Triangulierte Oberflächen zur Simulation des Propellers. Die Drehachse des Rotors wird etwas verlängert und ragt aus dem äußeren Zylinder heraus. Der Durchmesser der Drehachse ist 0.05278 Meter. Der größte Umfang der Rotorflügel beträgt ca. 0.22 Meter. Die Länge des Rotors beträgt 0.271 Meter. Der Umfang des äußeren Zylinders ist 0.6 Meter. Die Länge des Zylinders beträgt 1 Meter.

Geometrie-Abmessungen angegeben. Die untersuchte Geometrie wurde einem der Tutorials von OpenFOAM entnommen. Dabei wurden alle nicht konvexen scharfen Kanten abgerundet, so dass der entwickelte Netzbewegungslöser keine konkaven Polyeder erzeugen kann. Dadurch werden die in OpenFOAM definierten Kriterien zur Gitterqualität erfüllt. Im Bezug auf die Abbildung 5.1 befindet sich der Einlass links und der Auslass rechts. Beim Einlass wird eine einströmende Geschwindigkeit von $5 \frac{m}{s}$ definiert. Da der entwickelte Algorithmus zur Netzbewegung mit den in OpenFOAM integrierten Turbulenzmodellen zur Zeit noch nicht kompatibel ist, werden alle Rechnungen mit der Einstellung "laminar" simuliert. Zum Zweck der Validierung wurden insgesamt vier transiente Rechnungen gemacht. Diese sind in der Tabelle 5.1 aufgelistet. Im weiteren werden diese Rechnungen detailliert beschrieben.

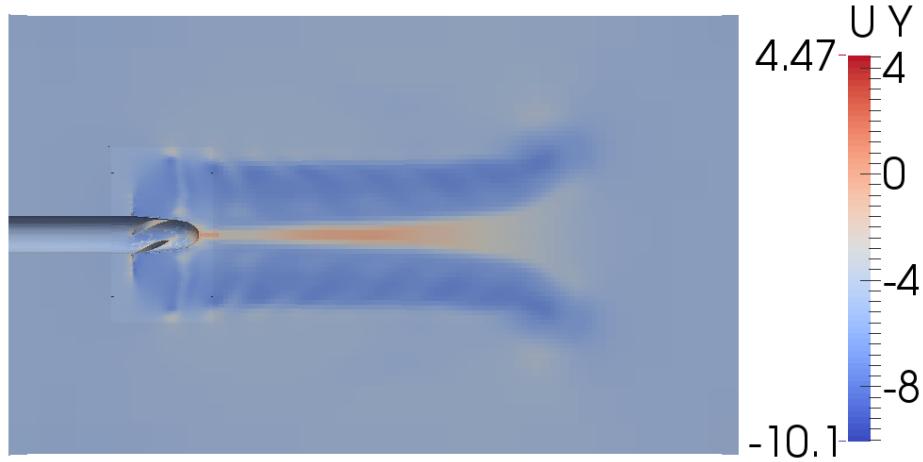
	Rechenfall	Typ des Gitters	Anzahl der Zellen
1	snappyHexMesh mit der AMI Randbedingung	Im Inneren Hexaeder; An dem Rand durch Modifikation der Hexaeder erhaltenen Polyeder	525586
2	ANSA-Netz mit der AMI Randbedingung	Zu Polyedern konvertierte Tetraeder	494592
3	Der entwickelte Netzbewegungslöser auf Basis von CVT	Centroidal-Voronoi-Zellen	579972
4	ANSA-Netz, bei dem alle Knoten rotiert werden.	Zu Polyedern konvertierte Tetraeder	923968

Tabelle 5.1: Rechenfälle zur Validierung des entwickelten Netzbewegungslösers

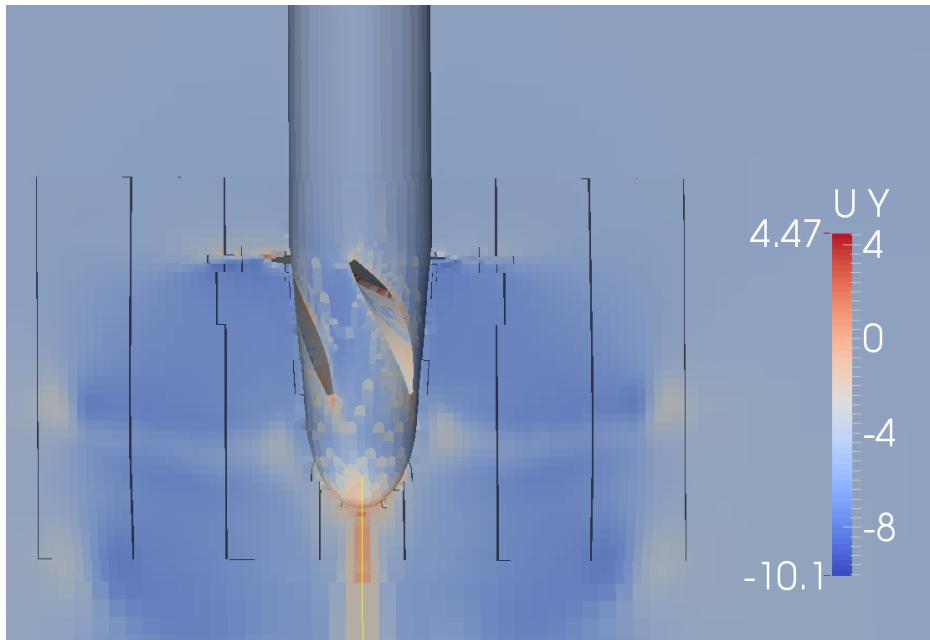
1. Zuerst wurde das in OpenFOAM hinterlegte Tutorial mit vorgegebenen Einstellungen simuliert. Es handelt sich um den Ansatz, bei dem zwei Teilvolumina unterschiedlich vernetzt werden. Um den Rotor wird ein bewegtes Teilvolumen erzeugt. Das zwischen diesem und dem Zylinder eingeschlossene Volumen wird getrennt vernetzt und nicht bewegt. Der Informationsaustausch findet durch Interpolation zwischen den dafür speziell vordefinierten Rändern statt. Die entsprechende Randbedingung für diese Ränder heißt in OpenFOAM AMI (Engl. Arbitrary Mesh Interface). Die Hauptidee des interpolierenden Algorithmus besteht in der Bestimmung der überlappenden Flächeninhalten zwischen den angrenzenden Flächenelementen und der Verwendung der zu diesen Flächenelementen zugehörigen Zellen als direkte Nachbarn, d.h. diese überlappenden Flächeninhalte tragen zu den Koeffizienten der Lösungsmatrix bei.

Offenbar besteht ein großer Nachteil dieser Methode zum einen in der Existenz infinitesimal kleiner überlappenden Flächeninhalte und dem Vorhandensein einer großen Anzahl der Nachbarn pro Zelle. Die Nachbarschaften über die Flächen mit den kleinen Inhalten können vernachlässigt werden, wodurch jedoch die Bilanz der Erhaltungsgrößen teilweise gestört werden kann. Zum anderen kann das Gitterqualitätskriterium Orthogonalität sehr schlecht ausfallen. Die Abbildung 5.3 zeigt zwei relativ flache Zellen und eine kleine Überlappungsfläche. Der Winkel zwischen den Vektoren \mathbf{u} und \mathbf{n} grenzt an 90° , wodurch $\cos(\alpha) = \frac{\mathbf{n} \cdot \mathbf{u}}{\|\mathbf{u}\|} \approx 0$. Das führt dazu, dass der im Abschnitt 4.6.4 abgeschätzte Approximationfehler bei der Diskretisierung des Laplace-Operators unendlich große Werte annehmen

kann (vergleiche die Gleichung (4.127)). Entscheidend ist hierbei, dass der Druck beim verwendeten SIMPLE-Algorithmus als Lösung der Laplace-Gleichung errechnet wird (siehe (4.105) bzw. (4.108)).



(a) Y-Komponente der Geschwindigkeit im ganzen Zylinder



(b) Y-Komponente der Geschwindigkeit im Bereich des Rotors

Abbildung 5.2: Das Geschwindigkeitsfeld bei dem unter OpenFOAM abgelegten Tutorial ‘‘propeller’’. Die Anzahl der Zellen beträgt 525586. Das Netz wurde mit dem Tool snappyHexMesh erzeugt. Dadurch sind die meisten Zellen Hexaeder.

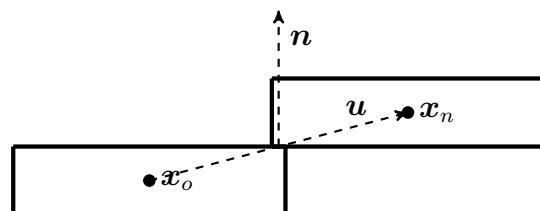


Abbildung 5.3: Ungünstige Überlappung von zwei Flächenelementen. $\cos(\alpha) = \frac{\mathbf{n} \cdot \mathbf{u}}{\|\mathbf{u}\|} \approx 0$.

Bei einer ungünstigen Überlappung von zwei Gittern können dadurch die beiden beschriebenen Probleme zu sehr großen Oszillation des Druckes führen (siehe die Abbildung 5.12).

Die Abbildung 5.2 zeigt das Ergebnis der Simulation des in OpenFOAM hinterlegten Rechenfalles auf Basis der AMI-Randbedingung. Das Geschwindigkeitsfeld wurde an dem Zeitpunkt $t = 0.1s$ erfasst. Wie bereits schon erwähnt wurde, werden alle in diesem Abschnitt untersuchten Rechenfälle in dem "laminaren" Modus simuliert. Diese Einstellung hat den Vorteil, dass nach einem gewissen Zeitpunkt sich ein konstantes Strömungsprofil bildet, bei dem sowohl das Geschwindigkeitsfeld als auch das Druckfeld keine großen Änderung über die Zeit erfahren. Der Betrachtungspunkt ist dabei relativ zu dem drehenden Propeller. Es hat sich herausgestellt, dass bei dem untersuchten Rechenfall, dieses konstante Strömungsprofil schon ab dem Zeitpunkt $t = 0.1s$ vorliegt. In den weiteren Rechenschritten waren nur sehr geringe Schwankungen der zu lösenden Größen zu beobachten. Diese Tatsache führt dazu, dass die hier untersuchten transienten Rechnungen nur an einem bestimmten simulierten Zeitpunkt verglichen werden können.

2. Der in diesem Punkt beschriebene Rechenfall wurde in dem kommerziellen Preprozessor ANSA aufgesetzt, der durch die Firma „BETA CAE Systems S.A.“ entwickelt und vertrieben wird. Sowohl das rotierende Teilvolumen als auch das ruhende Teilvolumen wurden mit Polyedern vernetzt. Im Falle polyedrischer Gitter ist es bei der AMI-Randbedingung darauf zu achten, dass alle zu den beiden AMI-Rändern gehörenden Zellen dem gleichen Prozessor zugeordnet werden. Die Abbildung 5.4 zeigt das Teilnetz, das die beiden AMI-Patches enthält. Es wird sich später herausstellen, dass die AMI-Patches polyedrischer Gitter im Bezug auf die Stetigkeit der Strömungsfelder wesentlich bessere Eigenschaften als die AMI-Patches hexaedrischer Gitter haben.



Abbildung 5.4: Das Teilnetz des Prozessors, der AMI-Randbedingung enthält

3. Hierbei handelt es sich um den entwickelten Ansatz auf Basis der Centroidal-Voronoi-Tessellierung. Bei der Simulation wird die triangulierte Randfläche des Propellers mit der vorgegebenen Winkelgeschwindigkeit rotiert. Die Generatoren der an diese Randfläche angrenzenden Zellen werden dem entsprechenden Dreieck des Randgitters zugewiesen und mitbewegt. Die neuen Positionen dieser Generatoren sind aus den neuen Koordinaten der zugehöriger Dreiecke zu bestimmen. Dabei bleibt die relative Position jedes Generators

zu allen drei Dreieckspunkten konstant. Dieses Vorgehen entspricht der Randbedingung `moveWithSurface` aus dem Unterabschnitt 3.7.3. Die an den Einlass angrenzenden Zellen werden mit der Randbedingung `projectOnToSurface` behandelt. Für die restlichen auf dem Zylinder liegenden Zellen wird jeweils die Randbedingung `moveWithSurface` gesetzt.

4. Der in diesem Punkt untersuchte Rechenfall hat eine höhere Anzahl von Volumenelementen und wird durch die Rotation des gesamten Gitters simuliert. Die Geschwindigkeit auf der Zylinderwand wird auf Null gesetzt. Aufgrund dieses Lösungsansatzes eignet sich dieser Rechenfall als eine Referenzlösung zum Abgleich anderer Rechenfälle.

Die Abbildung 5.6 zeigt das Geschwindigkeitsfeld aller vier untersuchten Rechenfälle. Es ist zu sehen, dass die errechneten Geschwindigkeitsfelder in etwa ähnlichen Wertebereichen liegen. Man erkennt hierbei, dass das Centroidal-Voronoi-Gitter im Bereich der Hauptströmung am schlechtesten aufgelöst ist. Die Ursache dafür liegt darin, dass die meisten Zellen sich im Bereich der Stirnfläche der Propellerflügel befinden. Für diese Fläche wurde eine wesentlich höhere Zelldichte definiert, um die konkaven Volumenelemente an dieser Stelle zu vermeiden. Das Ergebnis dieser Einstellung ist in der Abbildung 5.5 dargestellt. Alleine auf der Stirnfläche der Propellerflügel befinden sich 262710 Zellen. Das entspricht etwa der Hälfte der gesamten Anzahl der Volumenelemente. Offenbar besitzt dieser Aspekt ein großes Optimierungspotential. Analog

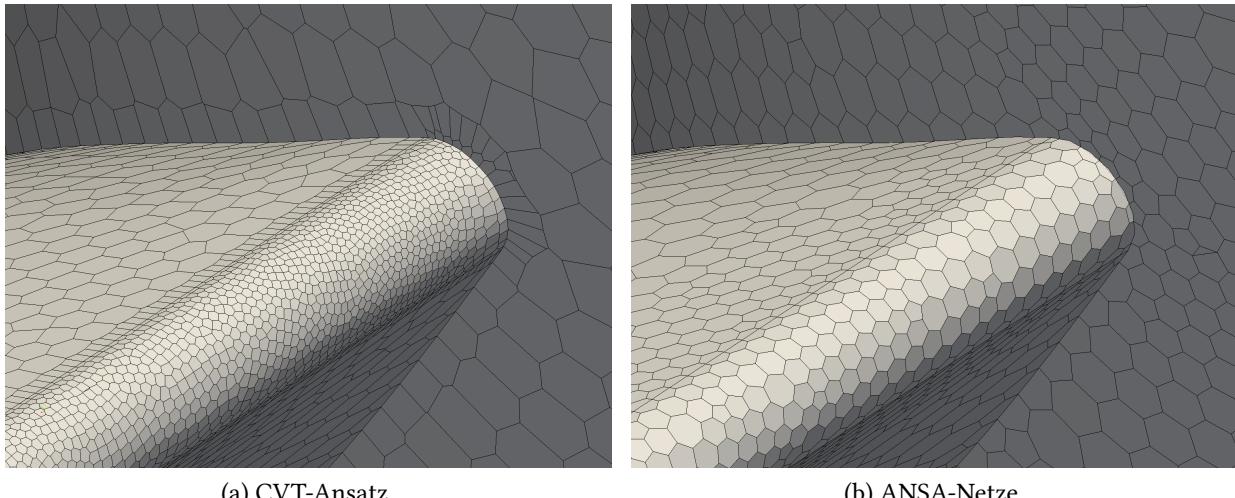
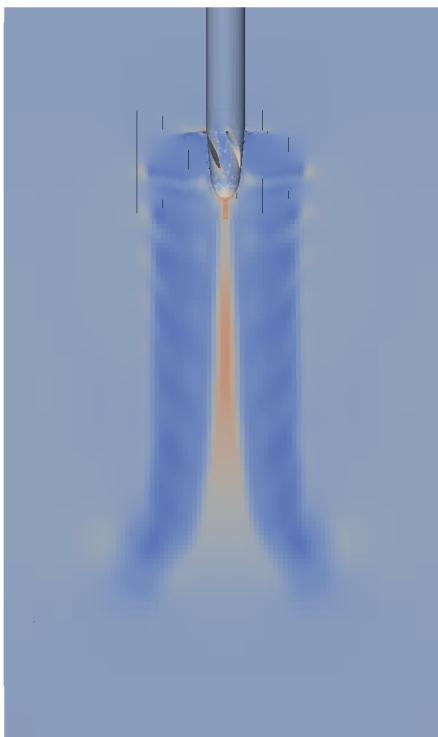


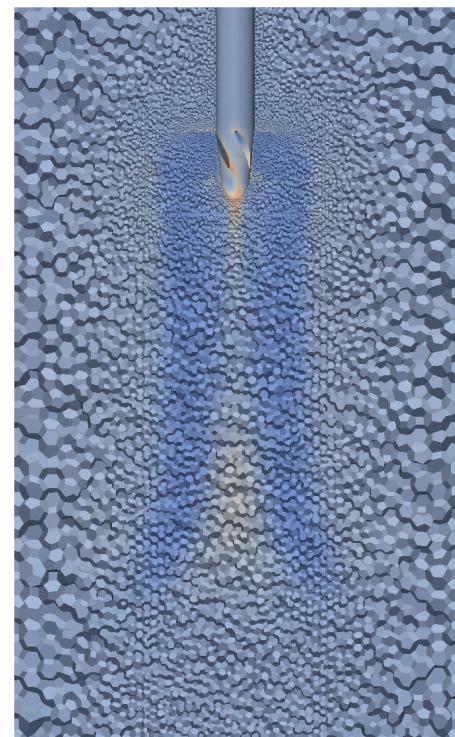
Abbildung 5.5: Stirnfläche der Propellerflügel in der Gitterdarstellung

zur Abb. 5.6 zeigt die Abbildung 5.7 die Rechenfälle von der gleichen Perspektive, jedoch wurde hier der Wertebereich durch das kleinste Intervall begrenzt. Um die Geschwindigkeitsfelder visuell miteinander vergleichen zu können, wurden diese an einer XY-Schnittebene ausgewertet. Die Abbildung 5.8 zeigt die Gitter aller 4 untersuchten Rechenfälle.

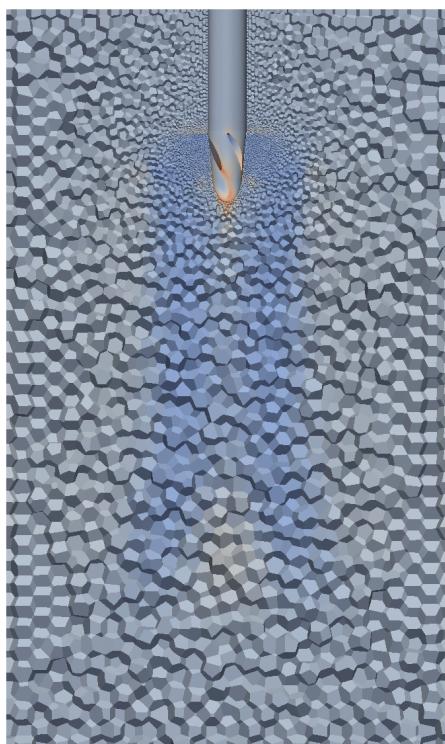
Die Abbildung 5.9 zeigt das Geschwindigkeitsfeld im Bereich des Rotors für die vier untersuchten Rechenfälle. Das Augenmerk fällt hier auf einen sehr homogenen Verlauf des Geschwindigkeitsfeldes im Bereich des Rotors gegenüber den restlichen Lösungen. Dies ist darauf zurückzuführen, dass die durch den CVT-Ansatz erzeugten Netze eine Art globale Optimalität besitzen (Siehe Abschnitt 3.2), wodurch diese Netze in Richtung der konstanten Zelldichte eine homogene Verteilung der Zellen aufweisen. Offensichtlich zählt dies zu einer wichtigen und positiven Eigenschaft der Centroidal-Voronoi-Gitter. Aufgrund der homogenen Zelldichte-verteilung beim CVT-Gitter ist der Verlauf der Wandschubspannung an den Rotorflügeln so gut wie gleichmäßig (siehe die Abbildung 5.10).



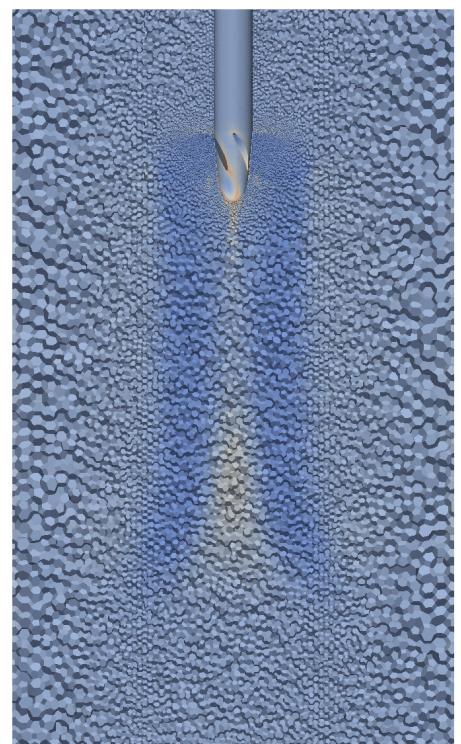
(a) SnappyHexMesh mit AMI



(b) ANSA-Netz mit AMI

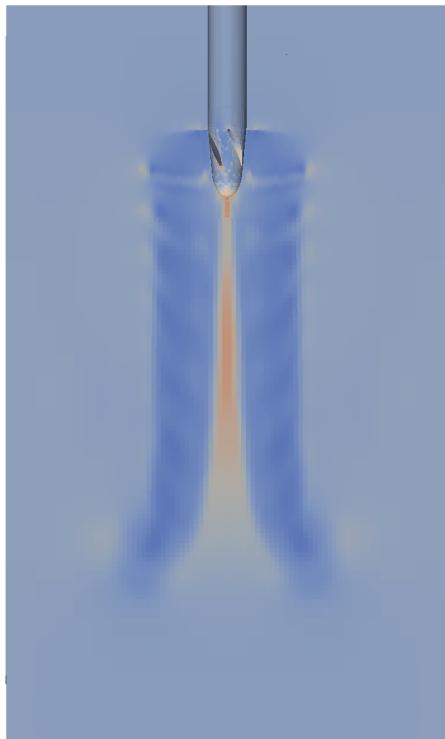


(c) CVT-Ansatz

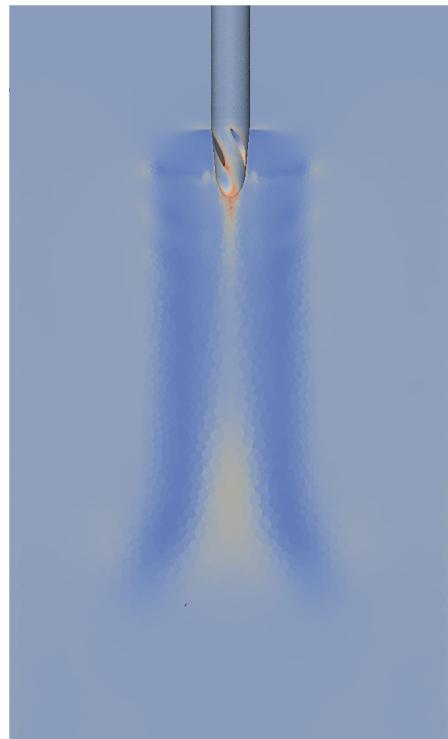


(d) ANSA-Netz mit Rotation

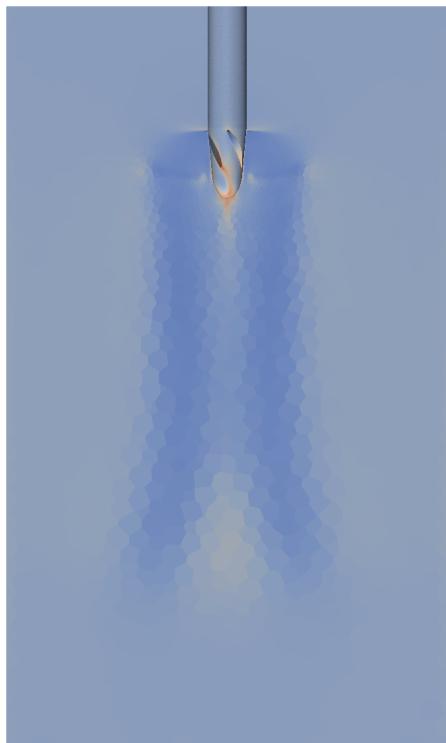
Abbildung 5.6: Simulation des drehenden Propellers. Y-Komponente des Geschwindigkeitsfeldes in der XY-Ebene. Die Lösung wird an den Zellmittelpunkten ausgewertet.



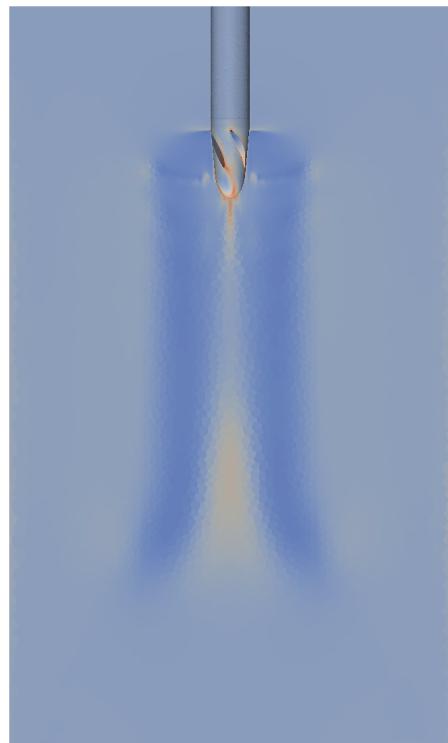
(a) SnappyHexMesh mit AMI



(b) ANSA-Netz mit AMI



(c) CVT-Ansatz



(d) ANSA-Netz mit Rotation

Abbildung 5.7: Simulation des drehenden Propellers. Y-Komponente des Geschwindigkeitsfeldes wird an der XY-Schnittebene ausgewertet. Der Wertebereich wurde durch das kleinste Intervall beschränkt.

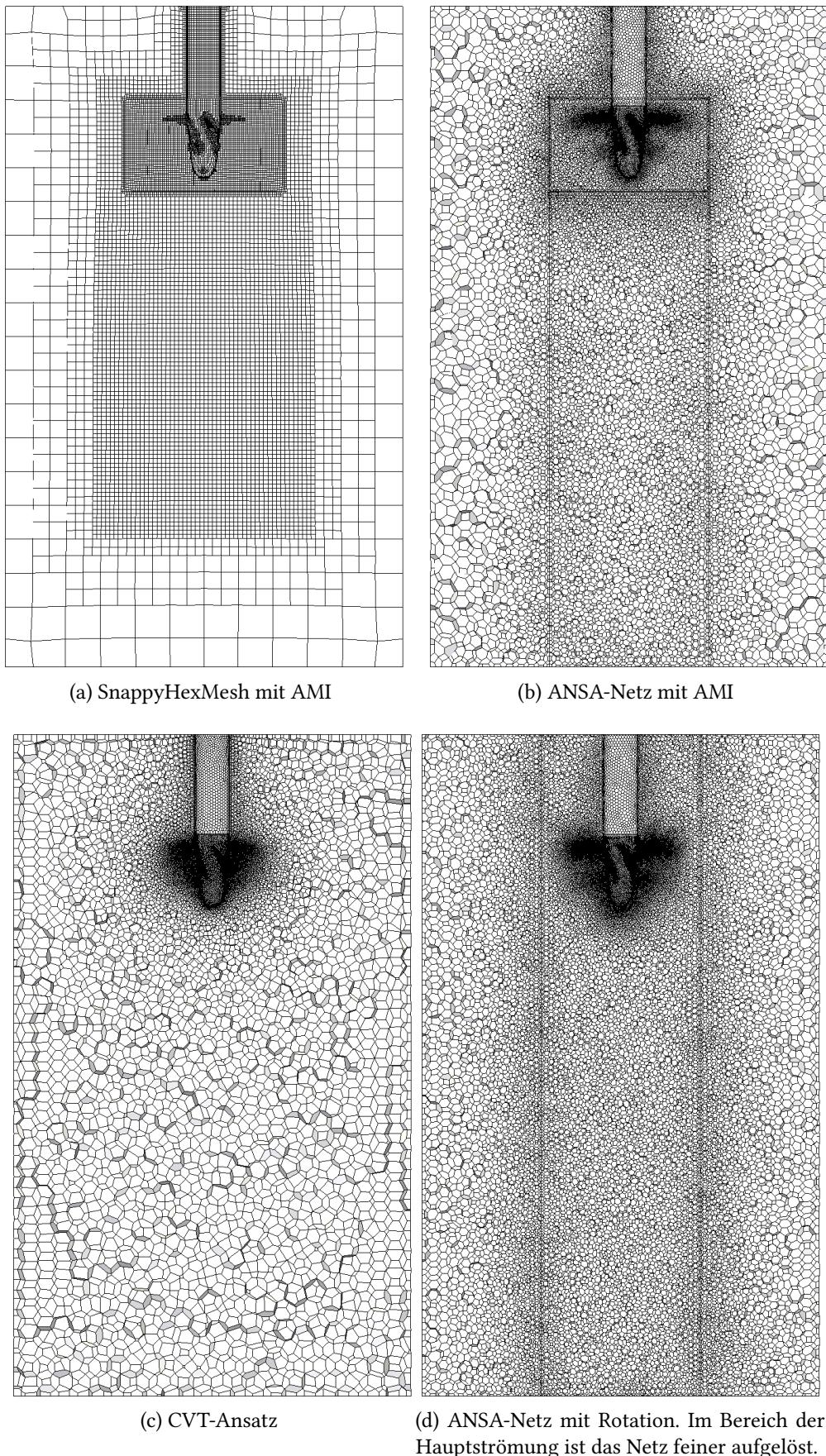


Abbildung 5.8: Simulation des drehenden Propellers. Die Netze aller 4 Rechenfälle.

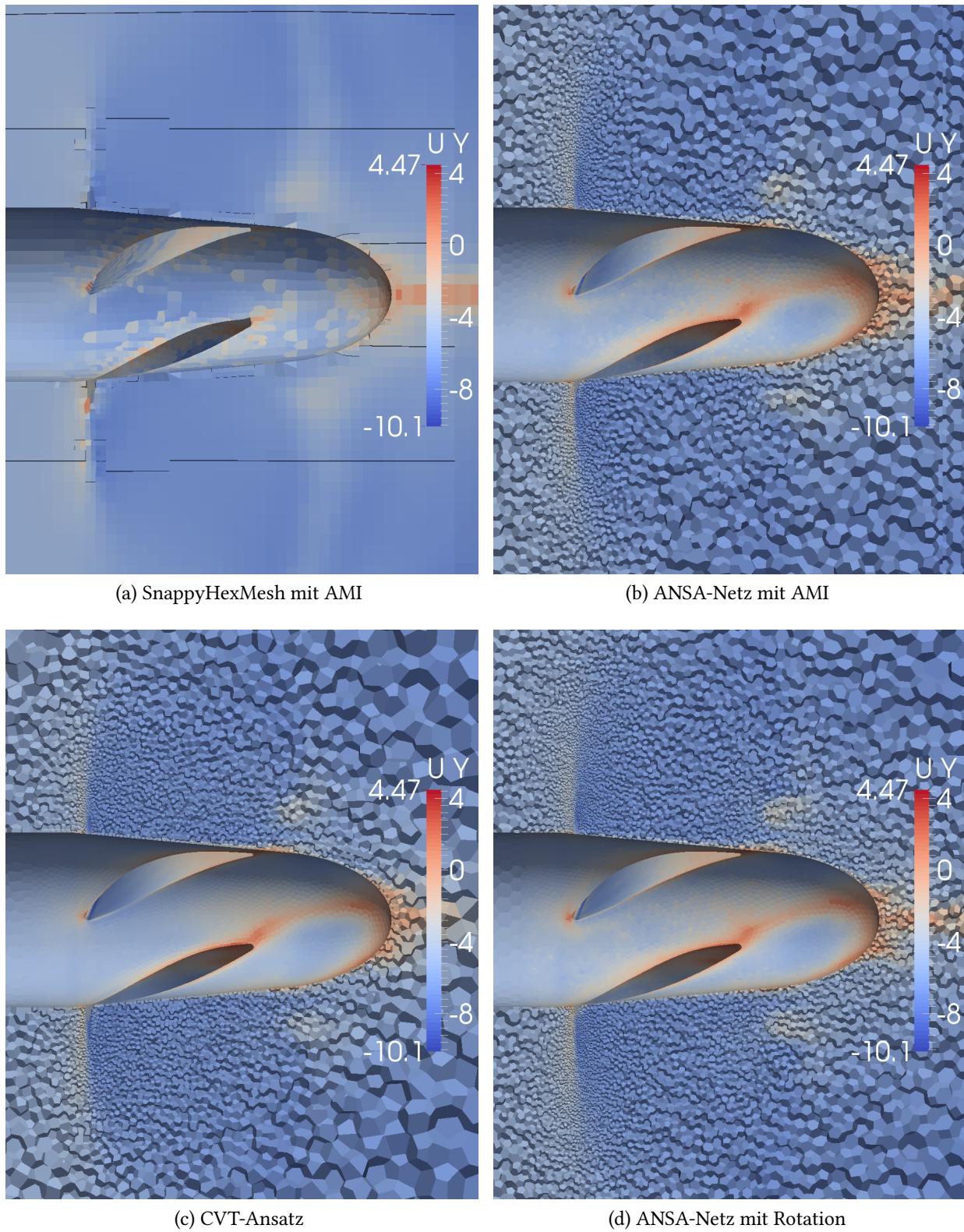


Abbildung 5.9: Simulation des drehenden Propellers. Y-Komponente des Geschwindigkeitsfeldes in der YZ-Ebene. Die Lösung wird an den Zellmittelpunkten ausgewertet. Der vergrößerte Bereich des Rotors. Der Wertebereich wurde durch das kleinste Intervall beschränkt.

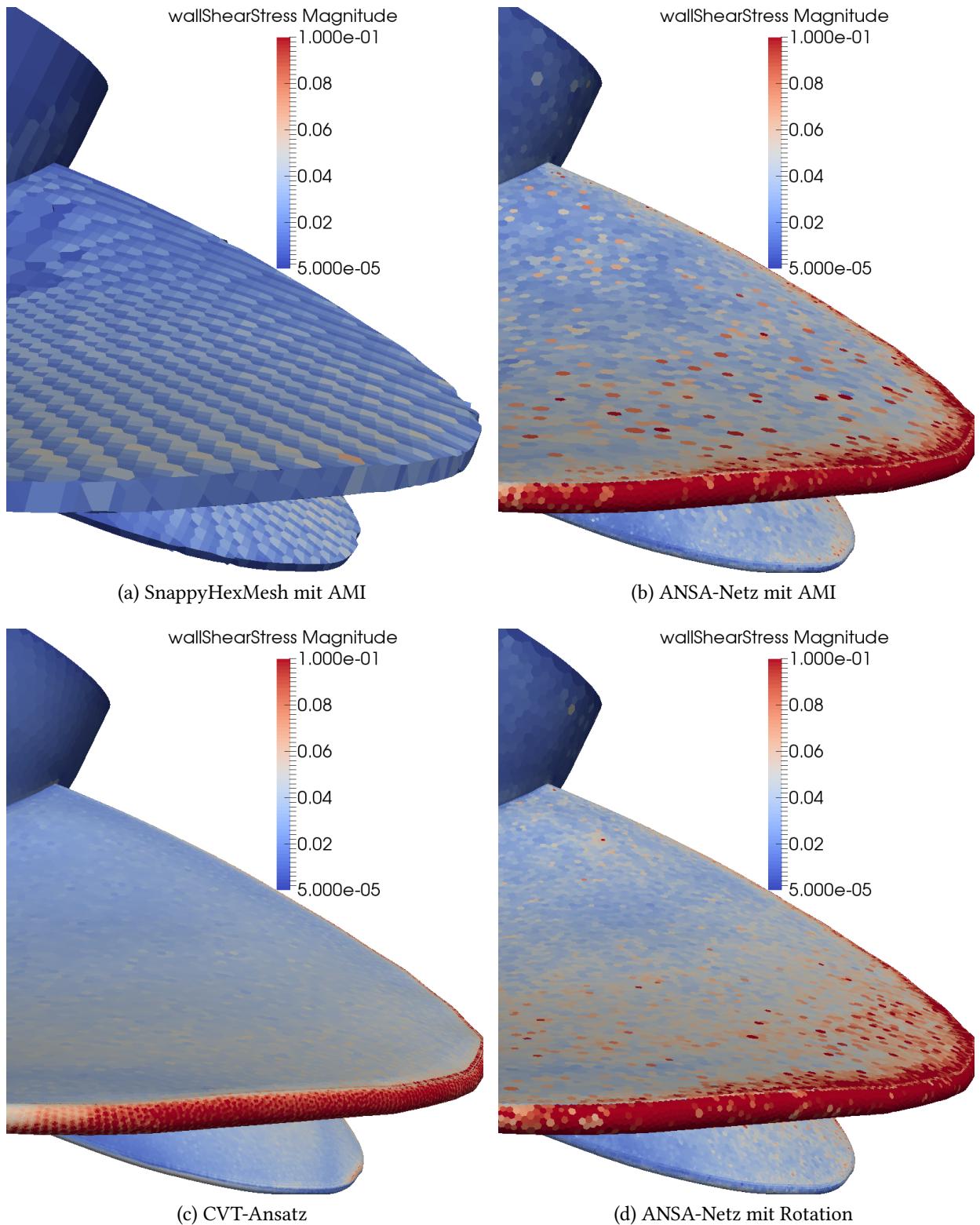


Abbildung 5.10: Simulation des drehenden Propellers. Der Betrag der Wandschubspannung.

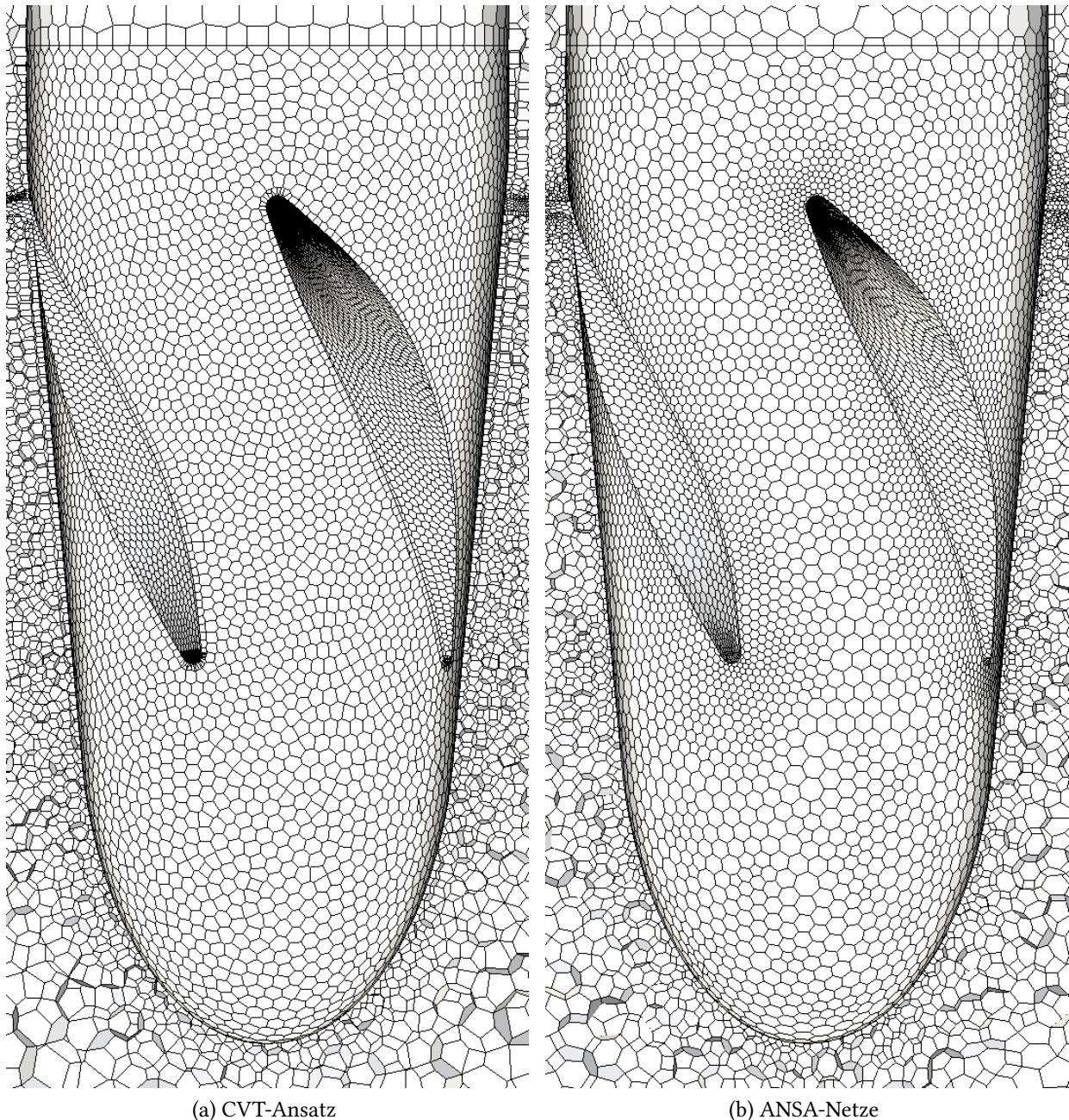


Abbildung 5.11: Rotor in der Gitterdarstellung

Um die durchgeführten Rechnungen quantitativ zu bewerten, wurde die Druckkraft am Einlass über 100 Zeitschritte ausgewertet. Die Abbildung 5.12 zeigt den zeitlichen Verlauf der Druckkraft der vier untersuchten Rechenfälle. Da die Rechnungen im laminaren Modus durchgeführt wurden, ist es zu erwarten, dass sich nach einem bestimmten Zeitschritt eine konstante Druckkraft am Einlass herrscht. Bei dem Rechenfall durch Rotation der Netzknoten trifft dies auch zu. Bei der Lösung durch den CVT-Ansatz sind vernachlässigbare Schwankungen der Druckkraft zu beobachten. Bei den beiden Lösungen mit der AMI-Randbedingung sind große Schwankungen der Druckkraft zu sehen. Das hexaedrische Gitter weist dabei die größten Oszillationen der Druckkraft auf.

Die Ursache liegt hierbei in einem strukturierten Aufbau dieser Gitter. Beim Abgleiten von zwei AMI-*Patches* aneinander kommt es gleichzeitig bei mehreren Flächenelementen zu sehr kleinen Überlappungsflächen. Wenn beide AMI-*Patches* in radiale Richtung gleiche Auflösung haben, sind von diesem Phänomen alle Flächenelemente betroffen, die auf der zylindrischen Wand liegen. Die Abbildung 5.13a zeigt einen Prozessorpatch zwischen zwei Teilnetzen. Dies veranschaulicht die beschriebene Problematik. Die Wahl der unterschiedlichen Zelldichten für die angrenzenden AMI-*Patches* könnte die auftretenden Oszillationen der Druckkraft reduzieren, jedoch niemals verschwinden lassen. Denn auch das polyedrische Gitter mit der AMI-Randbedingung weist diese Oszillationen auf.

Bei der Verwendung der AMI-Randbedingung haben die hexaedrischen Gitter gegenüber den polyedrischen Netzen einen großen Vorteil im Bezug auf paralleles Rechnen. Bei den hexaedrischen Gittern können sowohl die Volumenelemente als auch die AMI-*Patches* beinahe gleichmäßig auf mehrere Prozessoren verteilt werden. Dadurch wird auch die AMI-Interpolation parallel ausgeführt. Dafür müssen die beiden AMI-*Patches* die gleiche Struktur in Richtung der Drehachse besitzen. Dadurch kann beispielsweise das hexaedrische Netz des Propellers in Richtung der Drehachse so aufgeteilt werden, dass jeder Prozessor eine gewisse Anzahl von AMI-Flächenelementen zusammen mit ihren überlappenden AMI-Flächenelementen aus dem Nachbar-*Patch* erhält. Siehe dafür die Abbildung 5.13. In der *decomposeParDict* müssen dafür die folgenden Einstellungen vorgenommen werden:

```
method          hierarchical;
hierarchicalCoeffs
{
    n              (1 4 1);
    delta          0.001;
    order          xyz;
}
```

Hierbei ist y die Richtung der Drehachse. Durch diese Einstellung wird das Gitter in vier Teilnetze in y -Richtung zerlegt.

Die in diesem Abschnitt durchgeführte Untersuchung hat gezeigt, dass der im Rahmen dieser Arbeit entwickelte Netzbewegungslöser wesentlich bessere Eigenschaften im Bezug auf die Stetigkeit des Druckfeldes gegenüber der Verwendung von AMI-*Patches* besitzt. In den weiteren Schritten könnte dieser Ansatz zur Strömungssimulation der im Abschnitt 1.1 beschriebenen Verdrängerpumpen verwendet werden.

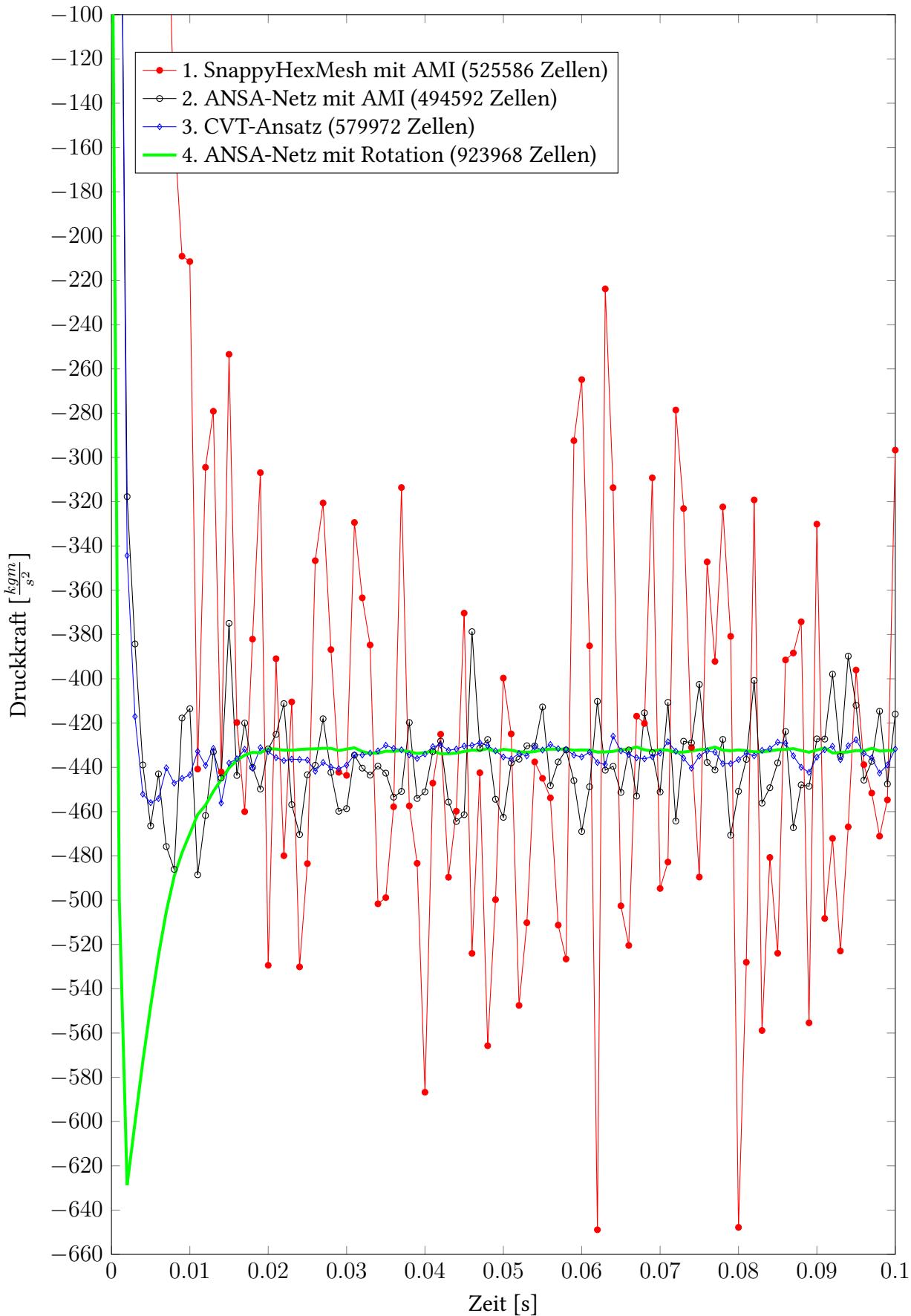


Abbildung 5.12: Druckkraft des Propellers

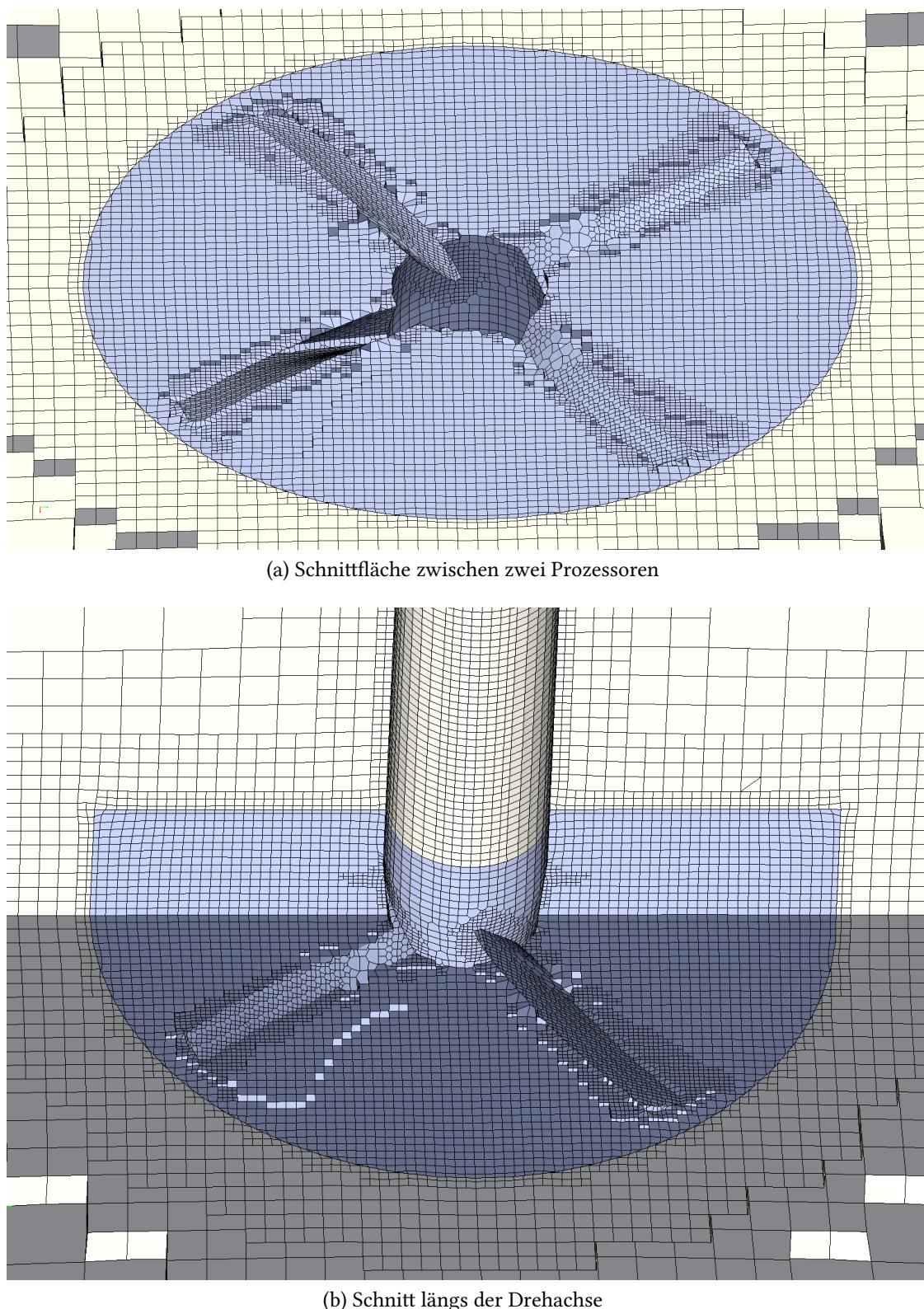


Abbildung 5.13: Teilnetz eines Prozessors vom Rechenfall 1

6 Fazit und Ausblick

- Das Ziel dieser Arbeit war die Entwicklung eines Netzbewegungslösers für unbegrenzt große Verformungen von volumetrischen Gittern. Anhand einer unserer früheren Arbeiten [83] und der Feststellungen im Kapitel 5 kann man feststellen, dass dieses Ziel erreicht wurde. Wie schon früher erwähnt wurde, benötigt dieser Netzbewegungslöser ein zumindest angenähertes Centroidal-Voronoi-Gitter. Für diesen Zweck kann der entwickelte Code auch verwendet werden. In der Praxis hat es sich jedoch herausgestellt, dass dieser Code nicht mit jedem Gitter umgehen kann. Bei nicht ausreichender Auflösung konkaver Randflächen können dabei nicht konvexe Zellen generiert werden. Im Abschnitt 2.5 wurde dieses Problem diskutiert. Der meist kritische Fall liegt vor, wenn eine Zelle in zwei Teilvolumina zerlegt wird. In diesem Fall kann man kein globales Gitter mehr erzeugen. Der entwickelte Code zur Netzgenerierung kann dadurch verbessert werden, dass die konkaven Zellen in der vorbereitenden Phase so zerlegt werden, dass es nur noch konvexe Zellen gibt. Wenn die Randzellen mit dem Randgitter verbunden werden, wird bei der Netzbewegung das beschriebene Problem nur sehr selten auftreten. Zur Ausnahme kommt es erst dann, wenn sich eine Zelle an den Rand anschließt.
- Durch Anbindung eines frei-verfügbaren Codes zur volumetrischen Vernetzung könnte der im Unterabschnitt 3.7.1 beschriebene Algorithmus zur Erzeugung des initialen CV-Gitters als ein automatisches OpenFOAM-Tool umgesetzt werden. Hierfür könnte auch der in dieser Arbeit entwickelte Algorithmus zur Aktualisierung der Voronoi-Diagramme so erweitert werden, dass dieser als ein Vernetzer verwendet werden kann. Dafür muss ein vereinfachtes Voronoi-Diagramm erzeugt werden, das beispielsweise nur aus Hexaedern bestehen kann.
- Beim Verschneiden des Voronoi-Diagramms des einhüllenden Quaders mit dem Randgitter wurde das Problem kritischer Punkte noch nicht gelöst. Es handelt sich dabei um den Fall, wenn das Randgitter exakt durch ein Voronoi-Knoten verläuft. Oder auch wenn eine Voronoi-Kante exakt auf dem Randgitter liegt.
- Außerdem ist der entwickelte Netzbewegungslöser mit den in OpenFOAM eingebauten Turbulenzmodellen nicht kompatibel. Wir nehmen an, dass es der Reinitialisierung einiger Felder bedarf. Denn unser Algorithmus erzeugt neue Flächenelemente. D.h. die Felder des Typs `surfaceScalarField` bzw. `surfaceVectorField` müssen in dem Code neu zugewiesen werden.
- Eine weitere nicht weniger interessante Entwicklung könnte in die Richtung der Untersuchung des Einflusses von konkaven Voronoi-Regionen auf die Finite-Volumen-Methode gehen. In erster Linie auch deshalb, da es an den scharfen konkaven Kanten wesentlich einfacher ist, nicht konvexe Zellen zu generieren, als diese zu zerlegen. Als erster Schritt sollte die Berechnung sowohl der Schwerpunkte als auch der Volumeninhalte für die konkaven Zellen entsprechend angepasst werden. Denn in OpenFOAM sind die konkaven Volumenelemente *a priori* nicht zugelassen.

Kapitel 6. Fazit und Ausblick

- Bei der Behandlung konkaver Zellen könnte man sich eine alternative Finite-Volumen-Diskretisierung überlegen, bei der als Definitionspunkt nicht der geometrische Schwerpunkt sondern der Generator des entsprechenden Voronoi-Gebiets herangezogen werden würde. Hierdurch müssten die Volumenintegrale für solche Elemente auf eine unkonventionelle Weise durch eine entsprechende Quadraturformel angenähert werden. Ansonsten würde dabei der Approximationsfehler der ersten Ordnung entstehen, was zu Inkonsistenzen der Erhaltungssätze führen könnte.
- Ein gleichermaßen wichtiges Potenzial liegt im Umsetzen des Konzepts des verteilten Rechnens auf unabhängigen Prozessen mittels Message-Passing-Interface (MPI). Dies ist der in OpenFOAM inhärente Ansatz. Zur Zeit werden nur die Berechnung der Schwerpunkte und die Voronoi-Zerlegung des einhüllenden Quaders parallel ausgeführt. Dies wurde mit OpenMP umgesetzt. D.h. paralleles Ausführen wird mithilfe zusätzlicher Threads erreicht. Die Lösung von den partiellen Differentialgleichungen kann demnach nur auf einem Thread durchgeführt werden. Die Berechnung des im Kapitel 5 beschriebenen Falles hat beispielsweise auf dem Intel-Core-i5-3450 fünf Tage gedauert.

Literatur

- [1] 2015. URL: <http://glaros.dtc.umn.edu/gkhome/metis/metis/overview>.
- [2] März 2016. URL: <http://www.cfx-berlin.de/software/stroemungsmechanik/twinmesh-fuer-verdraengermaschinen.html>.
- [3] März 2016. URL: <http://www.cfx-berlin.de>.
- [4] März 2016. URL: http://doc.cgal.org/latest/Triangulation_3/.
- [5] A. W. Date. "Complete pressure correction algorithm for solution of incompressible Navier-Stokes equations on a nonstaggered grid". In: *Numerical Heat Transfer, Part B: Fundamentals: An Internal Journal of Computation and Methodology* 29 (2007), S. 441–458.
- [6] A. W. Date. "Solution of Navier-Stokes equations on nonstaggered grid at all speeds". In: *Numerical Heat Transfer, Part B: Fundamentals: An Internal Journal of Computation and Methodology* 33 (2007), S. 451–467.
- [7] Alfio Quarteroni Riccardo Sacco Fausto Saleri. *Numerical Mathematics*. Heidelberg: Springer-Verlag, 2007.
- [8] Allen Gersho. "Asymptotically Optimal Block Quantization". In: *IEEE Transactions On Information Theory* 25.4 (1979).
- [9] Benjamin de Foy and William Dawes. "Unstructured pressure-correction solver based on a consistent discretization of the Poisson equation". In: *International journal for numerical methods in fluids* 34 (1999), S. 463–478.
- [10] Benjamin de Foy and William Dawes. "Unstructured pressure-correction solver based on a consistent discretization of the Poisson equation". In: *International Journal for Numerical Methods in Fluids* 34 (2000), S. 463–478.
- [11] C. Farhat, C. Degand, B. Koobus, M. Lesoinne. "Torsional springs for two-dimensional dynamic unstructured fluid meshes". In: *Computer Methods in Applied Mechanics and Engineering* 163 (1-4 Sep. 1998), S. 231–245.
- [12] Carlo L. Bottasso, Davide Detomib, Roberto Serra. "The ball-vertex method: a new simple spring analogy method for unstructured dynamic meshes". In: *Computer Methods in Applied Mechanics and Engineering* 194 (39-41 Okt. 2005), S. 4244–4264.
- [13] Charles S. Peskin. "The immersed boundary method". In: *Acta Numerica* (2002), S. 479–517.
- [14] Sou-Cheng (Terrya) Choi. "ITERATIVE METHODS FOR SINGULAR LINEAR EQUATIONS AND LEAST-SQUARES PROBLEMS". DOCTOR OF PHILOSOPHY. INSTITUTE FOR COMPUTATIONAL und MATHEMATICAL ENGINEERING in Stanford, 2006.
- [15] Christoph Degand, Charbel Farhat. "A three-dimensional torsional spring analogy method for unstructured dynamic meshes". In: *Computers & Structures* 80 (3-4 Feb. 2002), S. 305–316.
- [16] David Fong and Michael Saunders. *LSMR: An iterative algorithm for least-squares problems*. Institute for Computational und Mathematical Engineering (icME) Stanford University, Apr. 2010.

Literatur

- [17] David Silvester. *Fast Solvers for Unsteady Incompressible Flow*. University of Manchester.
URL: <http://www.maths.manchester.ac.uk/~djs/>.
- [18] Dehong Zeng, C. Ross Ethier. "A semi-torsional spring analogy model for updating unstructured meshes in 3D moving domains". In: *Finite Elements in Analysis and Design* 41 (11-12 Juni 2005), S. 1118–1139.
- [19] Desheng Wang and Qiang Du. "Mesh optimization based on the centroidal voronoi tessellation". In: 2 (2005), S. 100–113.
- [20] Dong-Ming Yan, Wenping Wang, Bruno Levy, Yang Liu. "Efficient Computation of Clipped Voronoi Diagram for Mesh Generation". In: *Computer-Aided Design* 45 (Apr. 2013), S. 843–852.
- [21] E. Bänsch, P. Benner, J. Saak and H. K. Weichelt. *Riccati-Based Boundary Feedback Stabilization of Incompressible Navier-Stokes Flow*. Deutsche Forschungsgemeinschaft, 2013.
- [22] F. Harlow and J. Welch. "Numerical calculation of time-dependent viscous incompressible flow of fluid with free surfaces". In: *The Physics of Fluids* 10.1063 (1965).
- [23] F. R. Menter. "Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications". In: *AIAA JOURNAL* 32.8 (1994).
- [24] Franz Durst. *Eine Einführung in die Theorie der Strömungen von Fluiden*. Springer-Verlag Berlin Heidelberg, 2006.
- [25] Fue-Sang Lien. "A pressure-based unstructured grid method for all-speed flows". In: *International journal for numerical methods in fluids* 33 (1999), S. 355–375.
- [26] George A. Markou, Zacharias S. Mouroutis, Dimos C. Charmpis, Manolis Papadrakakis. "The ortho-semi-torsional (OST) spring analogy method for 3D mesh moving boundary problems". In: *Computer Methods in Applied Mechanics and Engineering* 196 (4-6 Jan. 2007), S. 747–765.
- [27] H. Zhang, M. Reggio, J. Y. Trepanier and R. Camarero. "Discrete form of the GCL for moving meshes and its implementation in CFD schemes". In: *Computers Fluid* 22 (1993), S. 9–23.
- [28] H. Zhang, M. Reggio, J. Y. Trepanier, and R. Camarero. "Discrete form of the GCL for moving meshes and its implementation in CFD schemes". In: *Computers Fluids* 22.9 (1993).
- [29] Heiko Weichert. *Die Navier-Stokes Gleichung als DAEs*. Technische Universität Chemnitz, 2010.
- [30] Hrvoje Jasak, Zeljko Tukovic. *Automatic mesh motion for the unstructured finite volume method*. Nov. 2006.
- [31] I. Demirdi and M. Peri. "Space conservation law in finite volume calculations of fluid flow". In: *International Journal for Numerical Methods in Fluids* 8.1037 (1988).
- [32] I. Demirdi and M. Peri. "Finite volume method for prediction of fluid flow in arbitrarily shaped domains with moving boundaries". In: *International Journal for Numerical Methods in Fluids* 10.771 (1990).
- [33] Hairer und Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*. 1. Aufl. Heidelberg: Springer, 1991.
- [34] J. Donea, Antonio Huerta, J.Ph. Ponthot and A. Rodriguez-Ferran. "Chapter 14, Arbitrary Lagrangian-Eulerian Methods". In: () .

-
- [35] J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin. *Numerical Grid Generation*. Mississippi State, Mississippi, 1985.
 - [36] John Burkardt. 2015. URL: <http://people.sc.fsu.edu/~jburkardt/>.
 - [37] Franjo Juretic. "Error Analysis in Finite Volume CFD". Thesis submitted for the Degree of Doctor of Philosophy. Department of Mechanical Engineering Imperial College London, Dez. 2004.
 - [38] Karl Strehmel, Rüdiger Weiner, Helmut Podhaisky. *Numerik gewöhnlicher Differentialgleichungen. Nichtsteife, steife und differential-algebraische Gleichungen*. Springer Spektrum, 2012.
 - [39] Kely Mooney, Sandeep Menon, David Schmidt. *Adaptive Tetrahedral Remeshing for Deforming Domain Simulations*. Multiphase Flow Simulation Laboratory University of Massachusetts Amherst, Juni 2012.
 - [40] Ken HAYAMI, Jun-Feng YIN, and Tokushi ITO. *GMRES Methods for Least Squares Problems*. National Institute of Informatics, Juli 2007.
 - [41] Konrad Königsberger. *Analysis 2*. 4. Aufl. Springer-Lehrbuch, 2002.
 - [42] Kumaresan Nallasamy, Kuru Ratnavelu, Bernardine R. Wong. "Optimal control for Navier-Stokes Takagi-Sugeno fuzzy equations using Simulink". In: Proceedings of the International Conference on Logic, Information & Computation. 2011.
 - [43] Kyle Mooney and Jacques Papper. *Implementation of a moving immersed boundary method on a dynamically refining mesh with automatic load balancing*. 10 OpenFOAM Workshop, Juli 2015.
 - [44] Lars Davidson. "A pressure correction method for unstructured meshes with arbitrary control volumes". In: *International journal for numerical methods in fluids* 22 (1996), S. 265–281.
 - [45] Long Chen. *Mesh smoothing schemes based on optimal delaunay triangulations*. Math Department, The Pennsylvania State University, State College.
 - [46] M. Bierlaire, Ph. L. Toint, and D. Tuyttens. "On Iterative Algorithms for Linear Least Squares Problems With Bound Constraints". In: *LINEAR ALGEBRA AND ITS APPLICATIONS* 143 (1991), S. 111–143.
 - [47] M. J. Berger and P. Colella. "Local Adaptive Mesh Refinement for Shock Hydrodynamics". In: *Journal of Computational Physics* 82 (1989), S. 64–84.
 - [48] Magnus Winter. "Benchmark and validation of Open Source CFD codes, with focus on compressible and rotating capabilities, for integration on the SimScale platform." Master's Thesis in Engineering Mathematics. Gothenburg, Sweden, 2013.
 - [49] Marcel Viktor Jankowiak. "Kalibrierung transienter, quasi-periodischer 3D Strömungsrechnungen mittels 2C2D PIV". Dissertation. 2015.
 - [50] Martin Engel. "Numerische Simulation von Strömungen in zeitabhängigen Gebieten und Anwendung auf Fluid-Struktur-Wechselwirkungsprobleme". Diplomarbeit. Mathematisch-Naturwissenschaftliche Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn, 2002.
 - [51] Martin Hanke-Bourgeois. *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens*. Stuttgart - Leipzig - Wiesbaden: B. G. Teubner, 2002.
 - [52] Matthias Günter. *Algorithmische Geometrie*. Der Helex-Matze Verlag, 2010.
-

Literatur

- [53] Michele Benzi and Miroslav Tuma. “A robust incomplete factorization preconditioner for positive definite matrices”. In: *NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS* 10 (2003), S. 385–400.
- [54] Mohamed S. Ebeida and Scott A. Mitchell. *Uniform Random Voronoi Meshes*. Sandia National Laboratories, P.O. Box 5800, Albuquerque, NM 87185-1318 msebeid@sandia.gov.
- [55] *OpenFOAM C++ Documentation*. 2015. URL: <http://foam.sourceforge.net/docs/cpp/>.
- [56] Petri Fast, Michael J. Shelley. “A moving overset grid method for interface dynamics applied to non-Newtonian Hele-Shaw flow”. In: *Journal of Computational Physics* 195 (2004), S. 117–142.
- [57] Pierre Alliez, David Cohen-Steiner, Mariette Yvinec, Mathieu Desbrun. “Variational Tetrahedral Meshing”. In: *ACM Transactions on Graphics. Proceedings of ACM SIGGRAPH 2005* 24 (Juli 2005), S. 617–625.
- [58] Prem K. Kythe, Michael R. Schäferkotter. *Handbook of Computational Methods for Integration*. Chapman & Hall/CRC, 2005.
- [59] Qiang Du and Desheng Wang. “Tetrahedral mesh generation and optimization based on centroidal Voronoi tessellation”. In: *International journal for numerical methods in engineering* 56 (2003), S. 1355–1373.
- [60] Qiang Du and Desheng Wang. “Anisotropic centroidal voronoi tessellations and their applications”. In: *SIAM Journal on Scientific Computing* 26.3 (2005), S. 737–761.
- [61] Qiang Du, Maria Emelianenko. “Acceleration schemes for computing centroidal Voronoi tessellations”. In: *Numerical linear algebra with applications* 0 (2005), S. 1–19.
- [62] Qiang Du, Maria Emelianenko, and Lili Ju. “Convergence of the Lloyd algorithm for computing centroidal voronoi tessellations”. In: *SIAM Journal Numerical Analysis* 44.1 (2006), S. 102–119.
- [63] Qiang Du, Max D. Gunzburger, and Lili Ju. “Constrained Centroidal Voronoi Tessellations For Surfaces”. In: *SIAM Journal on Scientific Computing* 24.5 (2003), S. 1488–1506.
- [64] Qiang Du, Vance Faber, Max Gunzburger. “Centroidal Voronoi Tessellations: Applications and Algorithms”. In: *SIAM REVIEW* 41.4 (1999), S. 637–676.
- [65] R. Altmann and J. Heiland. “FINITE ELEMENT DECOMPOSITION AND MINIMAL EXTENSION FOR FLOW EQUATIONS”. In: *ESAIM: Mathematical Modelling and Numerical Analysis* 49 (2015), S. 1489–1509.
- [66] R. Löhner, and C. Yang. “Improved ALE mesh velocities for moving bodies”. In: *Communications in Numerical Methods in Engineering* (1996).
- [67] Rajat Mittal, Gianluca Iaccarino. “Immersed Boundary Methods”. In: *Fluid Mechanics* 239 (2005), S. 37–61.
- [68] Robert Altmann and Jan Heiland. “Regularization and Rothe Discretization of Semi-Explicit Operators DAEs”. Preprint-Reihe des Instituts für Mathematik Technische Universität Berlin. Feb. 2016.
- [69] Rolf Klein. *Algorithmische Geometrie. Grundlagen, Methoden, Anwendungen*. Berlin Heidelberg: Springer-Verlag, 2005.
- [70] Rony Keppens. *Adaptive Mesh Refinement*. Centre for Plasma-Astrophysics, Belgium, Mai 2008.

-
- [71] Rony Keppens, Omar Ghattas, Georg Stadler, Lucas C. Wilcox. *Adaptive Mesh Refinement (AMR)*. Institute for Computational Engineering und Sciences (ICES), März 2009.
 - [72] Chris H. Rycroft. *Voro++: a three-dimensional Voronoi cell library in C++*. 2009.
 - [73] S. Jakobsson, O. Amoignon. “Mesh deformation using radial basis functions for gradient-based aerodynamic shape optimization”. In: *Computers & Fluids* 36 (2007), S. 1119–1136.
 - [74] Sandeep Menon and David P. Schmidt. *Conservative interpolation on unstructured polyhedral meshes: An extension of the supermesh approach to cell-centered finite-volume variables*. Department of Mechanical und Industrial Engineering, University of Massachusetts.
 - [75] Sandeep Menon and David P. Schmidt. *Adaptive Tetrahedral Remeshing for Multiphase Flow Simulations in OpenFOAM*. Multiphase Flow Simulation Laboratory University of Massachusetts Amherst, Juni 2009.
 - [76] Scott T. Miller, R. L. Campbell, C. W. Elsworth, J. S. Pitt, D. A. Boger. “An Overset Grid Method for Fluid-Structure Interaction”. In: *World Journal of Mechanics* 4 (2014), S. 217–237.
 - [77] Sina Arabi, Ricardo Camarero, Francois Guibault. “Unstructured meshes for large body motion using mapping operators”. In: *Mathematics and computers in simulation* (Mai 2012).
 - [78] Vaidehi Ambatipudi. *Simple solver for driven cavity flow problem*. Purdue University.
 - [79] Volker Springel. *Cosmological moving-mesh hydrodynamics with AREPO*. Max-Planck-Institut for Astrophysics, Nov. 2009.
 - [80] Volker Springel. *Galilean-invariant cosmological hydrodynamical simulations on a moving mesh*. Max-Planck-Institut für Astrophysik, 2009.
 - [81] Wilhelm Kley. *Numerische Hydrodynamik*. Institut für Astronomie & Astrophysik Universität Tübingen, Juli 2004.
 - [82] Witalij Wambold. “Umsetzung des Optimierungsalgorithmus von Goldfarb und Idnani mit CUDA zur Außenkonturbestimmung von Fahrzeugen aus 3D-Laserscannerdaten”. Masterarbeit. 2011.
 - [83] Witalij Wambold, Günter Bärwolff. “New Mesh Motion Solver for Large Deformations based on CVT”. In: *Elsevier* 82 (2014), S. 390–402.
 - [84] Wolfgang Dahmen, Arnold Reusken. *Numerik für Ingenieure und Naturwissenschaftler*. Heidelberg: Springer-Verlag Berlin, 2005.
 - [85] Xia-ping Zhang, Dai Zhou, Yan Bao. “Mesh motion approach based on spring analogy method for unstructured meshes”. In: *Journal of Shanghai Jiaotong University* (Feb. 2010).
 - [86] Xuan Zhou, Shuixiang Li. “A new mesh deformation method based on disk relaxation algorithm with pre-displacement and post-smoothing”. In: *Journal of Computational Physics* 235 (Feb. 2013).

Danksagung

Hiermit möchte ich mich bei allen Personen bedanken, die zum Erfolg dieser Arbeit beigetragen haben.

Insbesondere danke ich Prof. Dr. Günter Bärwolff sowohl für sein Interesse und Vertrauen als auch für seine unkomplizierte und unbürokratische Fern-Betreuung meiner Promotion. Des Weiteren danke ich ihm für die Bereitstellung eines Arbeitsplatzes in der TU Berlin, wodurch viele komplizierte mathematische Fragen geklärt werden konnten. Darüber hinaus bedanke ich mich für die Aufnahme in seinen Fachbereich als Promotionsstudent und für die Übernahme der Be-gutachtung.

Prof. Dr. Volker Mehrmann danke ich für die Bereiterklärung zur Übernahme des Zweitgutach-tens.

Meinem Betreuer seitens Volkswagen AG Herrn Dr. Andreas Gitt-Gehrke danke ich für das in mich gesetzte Vertrauen bei der Durchführung dieser Arbeit, für das von ihm gestellte Thema, das mich vom ersten Tag meiner Promotion fasziniert hat, für seine sehr kompetente fachliche Unterstützung in beinahe aussichtslosen Situationen sowie für seine Begeisterung über meine Fortschritte und für die Übernahme des Drittgutachtens.

Beim Prof. Dr. Ulrich Pinkall bedanke ich mich für die Übernahme des Vorsitzes im Promotions-ausschuss.

Mein Dank gilt auch vielen Kolleginnen und Kollegen der Volkswagen AG, die mich bei der täglichen Arbeit unterstützten und begleiteten. Ein besonderer Dank richtet sich dabei an Sabine Baumbach für ihre Hilfsbereitschaft, Kollegialität und ihre freundliche Art den Arbeitsalltag aufzuheitern. Ein großes Dankeschön geht an Marcel Jankowiak für die angenehme Zusammenarbeit und für viel nützliche Information über die gemessenen Strömungsphänomene.

Für die Unterstützung und den Rückhalt, den ich von meiner Frau Eugenia, meinen Eltern, Schwiegereltern und Geschwistern erhalten habe, möchte ich mich bei meiner ganzen Familie besonders herzlich bedanken.

Lebenslauf

Akademische Bildung

- 01/2012–08/2016 Promotionsstudent der Naturwissenschaften an der Technischen Universität Berlin
- 04/2010–11/2011 Master Mathematik Computational Engineering an der Beuth Hochschule für Technik Berlin, Abschluss mit Auszeichnung
- 10/2005–03/2010 Bachelor Mathematik an der Beuth Hochschule für Technik Berlin mit Schwerpunkt Technik
- 08/2003–06/2005 Abitur an der Cottbus-Kolleg Institut zur Erlangung der Hochschulreife
- 09/2000–07/2001 Studium der Radiotechnik an der staatlichen Technischen Universität Omsk
- 09/1990–06/2000 Mittelschule in Omsk mit Schwerpunkt Technik

Berufstätigkeit

- 10/2016– Mathematiker in der Produktvorentwicklung bei Gesellschaft für optische Messtechnik
- 09/2015–09/2016 Softwareentwickler im Bereich Preprocessing für Crash-Simulationen bei GNS mbH in Braunschweig
- 01/2012–06/2015 Doktorand bei Volkswagen AG Salzgitter
- 04/2011–10/2011 Masterarbeit bei Daimler AG in Ulm
- 10/2010–03/2011 Studentischer Mitarbeiter mit Tutoraufgaben beim Fraunhofer Institut Produktionsanlagen und Konstruktionstechnik
- 09/2009–09/2010 Studentischer Mitarbeiter beim Fraunhofer Institut Produktionsanlagen und Konstruktionstechnik
- 06/2009–08/2009 Praktikum beim Fraunhofer Institut Produktionsanlagen und Konstruktionstechnik