Technische Universität München
Lehrstuhl für Computation in Engineering

Generierung von Netzen für Finite Elemente hoher Ordnung in zwei und drei Raumdimensionen

Dipl.-Ing. Univ. Christian G. Sorger, M.Sc.

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

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr.-Ing., Dr.-Ing. habil. G. H. Müller
Prüfer der Dissertation:	1.	Univ.-Prof. Dr. rer. nat. E. Rank
	2.	Univ.-Prof. Dr.-Ing. habil. A. Düster, Technische Universität Hamburg-Harburg

Die Dissertation wurde am 28.09.2012 bei der Technischen Universität München eingereicht und durch die Fakultät für Bauingenieur- und Vermessungswesen am 04.12.2012 angenommen.

Zusammenfassung

Die vorliegende Arbeit befasst sich mit der Generierung von Netzen, die für eine Berechnung mit der Finite-Elemente-Methode hoher Ordnung geeignet sind. Dabei werden Verfahren zur Vernetzung gekrümmter Flächenmodelle in Dreiecks- und Viereckselemente sowie zur Vernetzung von unterschiedlichen Volumenmodellen in Tetraeder- und Hexaeder-Elemente präsentiert.
Zunächst wird eine Einführung in die wesentlichen Aspekte des geometrischen Modellierens sowie in die Finite-Elemente-Methode hoher Ordnung gegeben und ein Konzept zur möglichst exakten Erfassung der Geometrie mithilfe gekrümmter Elemente dargelegt. Anschließend wird der Aufbau einer Rahmen-Software erläutert, welche als Fundament für die entwickelten Vernetzungstechniken dient. Die Vernetzung beliebig gekrümmter Oberflächenmodelle mit qualitativ hochwertigen Elementen erfolgt durch eine Kopplung der rekursiven Gebietsteilungstechnik mit differentialgeometrischen Berechnungsmethoden. Darauf aufbauend wird eine Strategie zur Vernetzung indirekter Volumenmodelle in Tetraeder-Elemente aufgezeigt. Abschließend wird ein Verfahren zur Generierung von Hexaeder-Elementen für komplexe dünnwandige Strukturen dargelegt und seine Praxistauglichkeit an zahlreichen Beispielen getestet.

Abstract

This thesis addresses the generation of meshes which are well suited for computations with high order finite element methods. It presents approaches for the generation of triangular and quadrilateral elements on curved surface models and for the generation of tetrahedral and hexahedral elements on solid models of different types.
Firstly it introduces the fundamental aspects of geometric modelling and the finite element method of high order. It focuses on a concept for describing curved elements to best cover the exact shape of the underlying geometry. Subsequently, the architecture of a framework-software which serves as a foundation for the developed meshing techniques is outlined. The generation of finite element meshes with high element quality on arbitrary curved surface models is achieved by coupling the recursive subdivision technique with mathematical methods of differential geometry. Based on this a strategy for generation of tetrahedral elements for indirect solid models is shown. Finally, a method for generation of hexahedral elements on complex thin-walled structures is presented and its suitability for practical use is demonstrated in several examples.

Vorwort

Im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Computation in Engineering der Technischen Universität München entstand diese Arbeit zwischen August 2007 und September 2012. Sie wurde als Projekt in Teilen durch die SOFiSTiK AG und durch das Elitenetzwerk Bayern gefördert.

Mein erster Dank gilt Herrn Prof. Dr. rer. nat. Ernst Rank für sein entgegengebrachtes Vertrauen, die großzügige Förderung dieses Projektes und die hervorragenden Arbeitsbedingungen während meiner gesamten Promotionszeit. Sein großes persönliches Interesse an dieser Arbeit war für mich immer ein besonderer Ansporn, und seine Vorschläge und Anregungen haben die Richtung dieser Arbeit maßgeblich beeinflusst.

Herrn Prof. Dr.-Ing. habil. Alexander Düster danke ich für das Interesse an dieser Arbeit und die Übernahme des zweiten Gutachtens. Seine Vorarbeiten auf dem Gebiet der Finite-Elemente-Methode hoher Ordnung waren die Grundlage für das Projekt.

Bei Herrn Prof. Dr.-Ing. Casimir Katz und der SOFiSTiK AG bedanke ich mich für die finanzielle Unterstützung meiner Arbeit und die hervorragende Zusammenarbeit. Mein besonderer Dank gilt Herrn Dr.-Ing. Andreas Niggl für die zahlreichen Diskussionen und Fachgespräche, die meine Arbeit über die Jahre stets unterstützend begleitet haben.

Ein ganz besonderer Dank gilt meinem langjährigen Kommilitonen, Zimmergenossen und Freund Dr. Iason Papaioannou, der mich immer in jeder Hinsicht unterstützt hat und während der vielen gemeinsamen Jahre an der Uni mein wichtigster Wegbegleiter geworden ist.

Für die geradezu familiäre Atmosphäre und das freundschaftliche Miteinander am Lehrstuhl bedanke ich mich bei allen Kollegen. Ein spezieller Dank gilt dabei meinem Gruppenleiter Dr. Stefan Kollmannsberger und meinen Kollegen Angelika Kneidl, Jovana Knežević, Hanne Cornils, Prof. Dr. André Borrmann und Felix Frischmann, die für mich in der gemeinsamen Zeit mehr Freunde wie Kollegen waren und dies auch heute noch sind.

Meiner Schwester Sabina Sorger danke ich für ihre Ermunterungen und Anstöße während der Anfertigung dieser Arbeit, und ganz besonders muss ich mich bei meinem Vater Walter Sorger für sein aufmerksames Lesen und die zahlreichen daraus resultierenden Hinweise bedanken, die dieser Arbeit erst den letzten Schliff gegeben haben.

Dass sie mir mit ihrer toleranten und großzügigen Erziehung erst den Weg für meinen Werdegang geebnet haben, dafür schulde ich meinen Eltern Walter Sorger und Dr. Elisabeth Sorger großen Dank.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Mit der Entwicklung von leistungsstarken Rechnern in der zweiten Hälfte des letzten Jahrhunderts begann die Erforschung von numerischen Berechnungsverfahren für die computergestützte Bearbeitung von komplexen, ingenieurrelevanten Fragestellungen der Struktur- und Strömungsmechanik. In den 50er und 60er Jahren des 20ten Jahrhunderts wurde die Methode der finiten Elemente (FEM), von den Professoren Argyris (Universitäten Stuttgart und London) und Clough (Universität Berkeley, Kalifornien) unabhängig voneinander entwickelt. Diese Methode stellt bis heute das wichtigste und am häufigsten eingesetzte Verfahren zur Berechnung von mechanischen Systemantworten dar.

In den Anfangsjahren der numerischen Simulation wurden zunächst nur kleine Systeme für stark vereinfachte geometrische Modelle angesetzt. Die Ursache hierfür lag einerseits in der mangelnden Rechenleistung und andererseits am zu geringen Speichervolumen der damaligen Rechensysteme. Durch die enorme Steigerung dieser beiden Faktoren stiegen auch die Größe der zu berechnenden Systeme sowie die Komplexität der geometrischen Aufgabenstellungen bis zum heutigen Zeitpunkt kontinuierlich an. In den vergangenen zwei Jahrzehnten war besonders im Bereich des computergestützten Konstruierens (CAD) ein deutlicher Leistungssprung von einfachen zweidimensionalen Modellierungstechniken hin zu Modellierungstechniken für die Erstellung von komplexen dreidimensionalen Freiformmodellen zu beobachten. Dies belegen die zahlreichen neu erschienenen oder weiter entwickelten CAD-Anwendungen der heutigen Zeit.

Mit steigender Komplexität stiegen auch die Anforderungen an die Techniken zur Diskretisierung des geometrischen Modells, was bis heute einer der wichtigsten und gleichzeitig problematischsten Schritte der Finite-Elemente-Analyse ist. Die Diskretisierung erfolgt über die näherungsweise Beschreibung des Berechnungsgebietes mit einem Finite-Elemente-Netz. Dieses besteht im Fall von zweidimensionalen Berechnungsgebieten aus einer endlichen Zahl von Dreiecks- oder Viereckselementen. Bei dreidimensionalen Berechnungsgebieten werden Volumenelemente wie Tetraeder oder Hexaeder angesetzt. Für die Qualität der Berechnungsergebnisse spielt die Größe, bei manchen strömungsmechanischen Problemen auch die

Ausrichtung der einzelnen Elemente eine entscheidende Rolle [69]. Die vollautomatische Generierung von geeigneten Finite-Elemente-Netzen ist daher eine Schlüsselaufgabe bei der Simulation geometriebasierter physikalischer Phänomene.

Die Unterteilung von geometrischen Flächenmodellen in qualitativ hochwertige Dreiecks- und Viereckselemente war besonders in den 80er und 90er Jahren des vergangenen Jahrhunderts im Fokus vieler Forschungsprojekte und ist heute durch zahlreiche grundlegende Vernetzungstechniken möglich [72]. Die Grenzen der heutigen Vernetzungssysteme liegen dabei hauptsächlich in der Vernetzung von topologisch komplexen oder geometrisch stark gekrümmten Freiformflächen, welche als Folge der gesteigerten Modellierungsmöglichkeiten immer häufiger werden. Durch eine Kopplung der Vernetzungstechniken mit differentialgeometrischen Darstellungsmethoden [107] sind solche Probleme oft einfach und effektiv zu lösen.

Die Vernetzung von dreidimensionalen Berechnungsgebieten ist im Vergleich zur Flächenvernetzung wesentlich aufwendiger und bis heute zentraler Bestandteil vieler Forschungsvorhaben. Der am einfachsten zu generierende dreidimensionale Elementtyp ist das Tetraeder-Element, für das eine ausreichende Zahl guter Vernetzungsalgorithmen existiert [72]. Deutlich komplizierter ist die Vernetzung mit Hexaeder-Elementen. Allerdings besitzen Hexaeder-Elemente im Vergleich zu Tetraeder-Elementen mathematische Vorteile [21, 36]. Eine Reihe von Vernetzungsprogrammen ist zwar auf die Generierung von Hexaeder-Elementen spezialisiert, doch bis zum heutigen Zeitpunkt ist noch kein Verfahren bekannt, welches eine allgemeingültige Lösung mit ausreichend guten Elementen für beliebige geometrische Problemstellungen liefert.

Die generierten Elemente unterscheiden sich nicht nur nach der Dimension und nach dem Elementtyp, sondern besitzen je nach Finite-Elemente-Ansatz auch unterschiedliche Anforderungen an die Diskretisierung. In der klassischen Finite-Elemente-Methode [19, 100] beschreiben die Elemente einfache Interpolationspolynome und werden durch gemeinsame Knoten miteinander verbunden. Die Auflösung des Berechnungsgebietes wird dabei durch die Anzahl der Elemente gesteuert. In der p-Version der Finite-Elemente-Methode [15, 40, 81] übernehmen die Elemente am Rand die exakte Form des Berechnungsgebietes und werden durch polynomiale Funktionsräume hoher Ordnung diskretisiert. Daher sind nur vergleichsweise wenige Elemente für eine exakte Beschreibung des geometrischen Modells notwendig.

Die vorliegende Arbeit beschäftigt sich mit der Generierung von Finite-Elemente-Netzen mit unterschiedlichen Elementtypen, die für eine Berechnung mit klassischen Ansätzen sowie mit der p-Version der Finite-Elemente-Methode geeignet sind.

Die Arbeit gliedert sich wie folgt:

Grundlagen

- Kapitel 2 legt die Grundlagen der geometrischen Modellierung dar. Zunächst werden die mathematischen Darstellungen einfacher geometrischer Kurven und Flächen sowie der Aufbau von komplexen Volumenmodellen beschrieben. Darauf aufbauend gibt das Kapitel Einblick in die wichtigsten Grundlagen der Differentialgeometrie und erläutert ausgewählte Aspekte der topologischen Algebra.
- Kapitel 3 widmet sich der Beschreibung der Finite-Elemente-Methode. Diese beginnt mit der Einführung eines allgemeinen Randwertproblems und seiner schwachen Formulierung. Im Anschluss daran wird die eigentliche Methode mit Hauptaugenmerk auf der Lösung des Integrationsproblems erklärt. Abschließend werden der klassische Ansatz und die p-Version der Finite-Elemente-Methode vorgestellt und miteinander verglichen.

Rahmen-Software

- Kapitel 4 stellt die im Zuge dieser Arbeit entwickelte Rahmen-Software *TUM.GeoFrame* vor. Diese bildet die Basis für alle implementierten Algorithmen. Der Schwerpunkt der Beschreibung liegt auf dem Aufbau und dem Design der Rahmen-Software, sowie dem Zusammenspiel zwischen den algorithmischen Prozeduren und der Datenbasis.

Netzgenerierung

- Kapitel 5 diskutiert die Generierung von Finite-Elemente-Netzen auf geometrischen Flächenmodellen. Die gezeigte Methode stellt eine Erweiterung des Netzgenerators *DoMesh* [82, 91] dar und wurde in den Vernetzungskern von *TUM.GeoFrame* integriert. Die Verbesserung basiert auf der Einbeziehung von wichtigen differentialgeometrischen Eigenschaften von Kurven und Flächen. Dadurch lassen sich qualitativ hochwertige Dreiecks- oder Viereckselemente auf beliebig geformten Freiformflächen erzeugen.
- Kapitel 6 widmet sich der Generierung von Volumenelementen auf dreidimensionalen Berechnungsgebieten. Die Generierung von Tetraedern erfolgt durch die Anbindung des Netzgenerators *Netgen* [90]. Eine anschließende Diskretisierung der Tetraeder-Oberflächen ermöglicht die Beschreibung des Netzes für die p-Version der Finite-Elemente-Methode. Die Generierung von Hexaeder-Netzen basiert auf einfachen Abbildungstechniken. Der finale Volumenkörper wird dabei aus einem dimensionsreduzierten Flächenmodell entwickelt. Im Blickfeld der neu entwickelten Methode steht die Bearbeitung unterschiedlicher Szenarien, bei denen sich einzelne dimensionsreduzierte Referenzflächen parallel, senkrecht, oder schiefwinklig berühren oder schneiden.

Der im Rahmen dieser Arbeit entwickelte Algorithmus ist speziell für dünnwandige Strukturen gut geeignet. Die Entwicklung eines allgemeingültigen Verfahrens zur Generierung von Hexaeder-Elementen auf beliebigen Volumenmodellen bleibt weiterhin ein Themengebiet mit vielen noch ungelösten Problemen. Das gezeigte Verfahren ist jedoch in der Lage, Netze für eine Reihe von geometrischen Modellen zu generieren, für die bisher noch keine geeignete Vernetzungsmethode bekannt war. Kapitel 7 nimmt dazu Stellung und wagt einen Ausblick auf mögliche zukünftige Arbeiten.

Kapitel 2

Grundlagen der geometrischen Modellierung

Die computergestützte Beschreibung der Form geometrischer Objekte spielt eine zentrale Rolle in der Berechnung von physikalischen Phänomenen im konstruktiven Maschinenbau, im Bauingenieurwesen und in vielen anderen Ingenieursdisziplinen. Hierfür existiert eine Vielzahl von unterschiedlichen Modellierungskonzepten, wobei die Wahl der jeweiligen Methode stark von der spezifischen Problemstellung sowie von den zu modellierenden Objekten abhängt. Dieses Kapitel möchte die für das Verständnis dieser Arbeit notwendigen Grundlagen der geometrischen Modellierung vermitteln. Für detaillierte Ausführungen wird auf die Literatur [7, 8, 47] verwiesen.

2.1 Geometrische Beschreibung von Kurven

Geometrische Kurven sind eindimensionale Objekte, die entweder in der euklidischen Ebene $\mathbb{R}^2$ oder im euklidischen Raum $\mathbb{R}^3$ liegen. Die einfachste Form einer Kurve ist die Gerade, die durch einen Start- und einen Endpunkt eindeutig bestimmt wird. Ist die Form der Kurve komplexer, existieren dafür verschiedene mathematische Darstellungsweisen. Dieser Abschnitt behandelt die mathematischen Beschreibungen der wichtigsten geometrischen Kurventypen. Die Auswahl umfasst dabei den Standardsatz eines jeden computergestützten Konstruktionsprogramms (CAD).

2.1.1 Darstellung von Kurven

Explizite Darstellung

Bei der *expliziten Darstellung* wird die Kurve C über eine Funktion

$$y = f(x) \tag{2.1a}$$

$$\text{im } \mathbb{R}^3 \text{ zusätzlich}: \quad z = g(x) \tag{2.1b}$$

beschrieben. Die Kurve durchläuft hierbei für die unabhängige Koordinate (x) ein vorgegebenes Definitionsintervall $I \subset \mathbb{R}$. Die abhängigen Koordinaten (y, z) entsprechen dabei in jedem Punkt des Definitionsintervalls einem Funktionswert.

Obwohl die *explizite Darstellung* aufgrund ihrer Eindeutigkeit analytisch direkt verwertbar ist, lässt sich damit nur eine sehr begrenzte Anzahl an Kurven beschreiben. Beispielsweise können keine geschlossenen Kurven (z.B. Kreis, Ellipse) oder Kurven mit Tangenten vertikal zur unabhängigen Koordinatenachse beschrieben werden. Daher wird die *explizite Darstellung* im Rahmen von Kurvenbetrachtungen eher selten benutzt.

Implizite Darstellung

Bei der *impliziten Darstellung* wird die Kurve C über die Lösungsmenge einer Funktion

$$f(x, y) = 0 \tag{2.2}$$

beschrieben und der Hinweis auf die Art der Koordinate ist unwesentlich.

Die *implizite Darstellung* erlaubt die Beschreibung von allen möglichen ebenen Kurven (z.B. Kreis, Ellipse) und liefert einen sehr effizienten Ansatz zur Bestimmung, ob ein Punkt auf einer Kurve liegt, bzw. auf welcher Seite er liegt und welchen Abstand er hat. Allerdings besitzt die *implizite Darstellung* oft mehr Lösungen als erwünscht (z.B. wird sowohl ein Kreis als auch ein Halbkreis durch die selbe implizite Funktionen $x^2+y^2-1=0$ beschrieben) und ist somit uneindeutig.

Parameterdarstellung

Bei der *Parameterdarstellung* der Kurve C werden die Koordinaten einzeln als Funktionen

$$x = f_x(t) \tag{2.3a}$$

$$y = f_y(t) \qquad \text{mit} \quad t_0 \leq t \leq t_1 \tag{2.3b}$$

$$\text{im } \mathbb{R}^3 \text{ zusätzlich}: \quad z = f_z(t) \tag{2.3c}$$

von einem Laufparameters t beschrieben. Der Laufparameter kann hierbei als Zeitachse interpretiert werden und durchläuft ein vorgegebenes Definitionsintervall $I \subset \mathbb{R}$. Die Koordinaten (x, y, z) werden zu jedem Zeitpunkt (t) einzeln über diese reellen Funktionen bestimmt.

In der *Parameterdarstellung* ist jede Steigung erlaubt und auch Überschneidungen sind möglich, allerdings verliert die Kurve vorteilhafte analytische Eigenschaften wie z.B. die direkte Differenzierbarkeit. Aufgrund ihrer Vielseitigkeit ist die *Parameterdarstellung* in der Computergrafik jedoch die am häufigsten verwendete Darstellungsart.

Kurventypen und ihre Darstellung

Explizite Kurven können ohne Schwierigkeit in eine *implizite Darstellung* und *implizite Kurven* jederzeit in eine *Parameterdarstellung* überführt werden. Umgekehrt ist das nicht für alle Kurventypen möglich. Daher basieren geometrische Algorithmen bevorzugt auf der *Parameterdarstellung*. Tabelle 2.1 gibt einen Überblick über die in dieser Arbeit behandelten Kurventypen und die jeweiligen Darstellungsmöglichkeiten.

Kurve	Explizit	Implizit	Parametrisch
Parabel	X	X	X
Hyperbel	X	X	X
Kreis	-	X	X
Ellipse	-	X	X
Bézier-Kurve	-	-	X
BSpline-Kurve	-	-	X

Tabelle 2.1: Kurventypen und ihre Darstellungsarten

2.1.2 Kegelschnitte

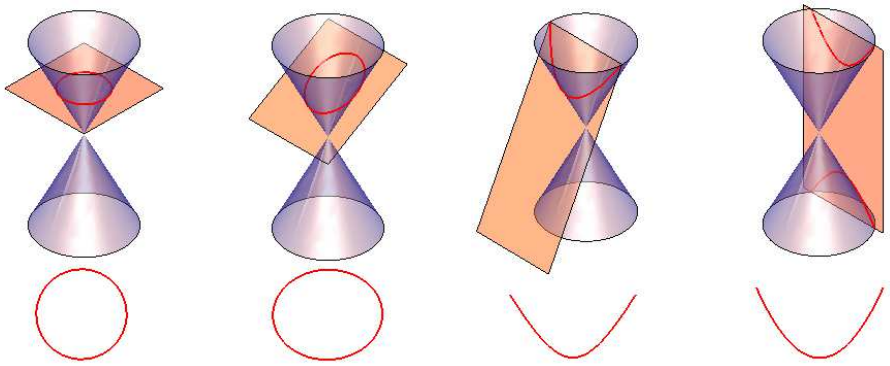
Kegelschnitte bezeichnen Kurven, die aus dem Schnitt eines Drehkegels mit einer Ebene ϵ hervor gehen, die nicht durch die Spitze des Kegels geht. Je nach Neigung der Ebene ϵ beschreibt die Schnittkurve einen *Kreis*, eine *Ellipse*, eine *Parabel* oder eine *Hyperbel*.

Ellipse

Bei der *Ellipse* ist der Winkel zwischen der Achse des Drehkegels und der Schnittebene größer als der halbe Öffnungswinkel des Kegels. Eine *Ellipse* mit den beiden Halbradien r_1 und r_2 lässt sich über die *implizite Darstellung*

$$f(x,y) = \left(\frac{x}{r_1}\right)^2 + \left(\frac{y}{r_2}\right)^2 - 1 = 0 \tag{2.4}$$

beschreiben. In der *Parameterform* lautet die Ellipsengleichung



(a) Kreis (b) Ellipse (c) Parabel (d) Hyperbel

Abbildung 2.1: Kegelschnitte

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} r_1 \cdot \cos(t) \\ r_2 \cdot \sin(t) \end{pmatrix} \quad \text{mit} \quad 0 \leq t \leq 2\pi. \tag{2.5a}$$

Kreis

Der *Kreis* ist ein Spezialfall der *Ellipse*, mit einem rechten Winkel zwischen der Achse des Drehkegels und der Schnittebene. Ein *Kreis* mit Mittelpunkt im Koordinatenursprung und Radius r besitzt die *implizite Darstellung*

$$f(x, y) = x^2 + y^2 - r^2 = 0 \tag{2.6}$$

In der *Parameterform* lautet die Kreisgleichung

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} r \cdot \cos(t) \\ r \cdot \sin(t) \end{pmatrix} \quad \text{mit} \quad 0 \leq t \leq 2\pi. \tag{2.7}$$

Parabel

Bei einer *Parabel* ist die Schnittebene ϵ parallel zu einer Mantellinie des Drehkegels, so dass der Winkel zwischen Achse und Ebene gleich dem halben Öffnungswinkel des Kegels entspricht. Eine *Parabel* mit Scheitelpunkt im Koordinatenursprung besitzt die *explizite Darstellung*

$$y = \lambda \cdot x^2 \tag{2.8}$$

In *impliziter Form* lautet die Parabelgleichung

$$f(x, y) = \lambda \cdot x^2 - y = 0. \tag{2.9}$$

Die *Parameterform* der *Parabel* lautet

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} t \\ \lambda \cdot t^2 \end{pmatrix} \quad \text{mit} \quad -\infty < t < \infty. \tag{2.10}$$

Hyperbel

Bei einer *Hyperbel* verläuft der Schnitt der Ebene ϵ parallel zu zwei Mantellinien des Drehkegels. Dies bedeutet, dass der Winkel zwischen Achse und Ebene kleiner als der halbe Öffnungswinkel des Kegels ist. Eine *Hyperbel* mit Mittelpunkt im Ursprung besitzt die *explizite Darstellung*

$$y = \frac{\lambda}{x} \tag{2.11}$$

In *impliziter Form* lautet die Hyperbelgleichung

$$f(x, y) = \left(\frac{x^2}{a^2}\right) - \left(\frac{y^2}{b^2}\right) - 1 = 0 \tag{2.12}$$

Die *Parameterform* der *Hyperbel* lautet

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{a}{\cos(t)} \\ \frac{b}{\tan(t)} \end{pmatrix} \quad \text{mit} \quad 0 \leq t \leq 2\pi. \tag{2.13}$$

2.1.3 Bézier-Kurven

Bézier-Kurven wurden Anfang der 1960er Jahre von Paul de Casteljau und Pierre Bézier unabhängig voneinander entwickelt und stellen ein wichtiges Werkzeug der geometrischen Kurvenbeschreibung dar. Eine *Bézier-Kurve* ist eine *Parameterdarstellung* eines Polynoms vom Grad p zu $p+1$ gegeben Kontroll- oder Bézier-Punkten, die mittels einer rekursiven Formel zu einer parametrischen Kurvengleichung verknüpft werden. *Bézier-Kurven* interpolieren den ersten und den letzten Kontrollpunkt und approximieren die inneren Kontrollpunkte. Dafür besitzen sie vorteilhafte mathematische wie geometrische Eigenschaften [51].

Bernsteinpolynome

Die *Bernsteinpolynome* wurden 1911 von Sergei Natanowitsch Bernstein formuliert und gehören zu den reellen Polynomen mit ganzzahligen Koeffizienten. *Bernsteinpolynome* bilden die Basis von *Bézier-Kurven* und sind ein nicht mehr weg zu denkender Grundstein des heutigen computergestützten Konstruierens. Die $p+1$ *Bernsteinpolynome* vom Grad p sind definiert als

$$B_{i,p}(t) = \binom{p}{i} t^i \cdot (1-t)^{p-i} \quad \text{mit} \quad 0 \leq t \leq 1 \tag{2.14}$$

für $i, p \in \mathbb{N}$ und $0 \leq i \leq p$, sowie $t \in [0,1]$. Alternativ lassen sich die *Bernsteinpolynome* über die einfache Rekursionsformel

$$B_{i,p}(t) = (i-t) \cdot B_{i,p-1}(t) + t \cdot B_{i-1,p-1}(t) \quad \text{mit} \quad 0 \leq t \leq 1 \tag{2.15}$$

erzeugen, mit $B_{i,p} = 0$ für $i < 0$ oder $i > p$, sowie $B_{0,0} := 1$. Abbildung 2.2 zeigt die fünf *Bernsteinpolynome* für $p = 4$.

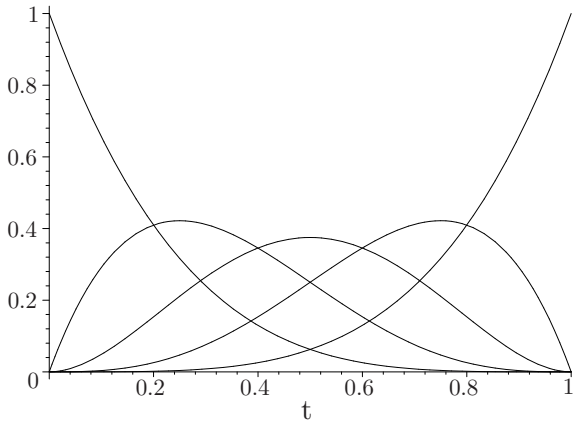


Abbildung 2.2: Bernsteinpolynome (p=4)

Nicht rationale Bézier-Kurven

Die *nicht rationale Bézier-Kurve* ist die einfachste Form der Darstellung einer Freiformkurve. Sie ergibt sich aus der Summe der Produkte der $p+1$ *Bernsteinpolynome* $B_{i,p}$ mit den $p+1$ Kontrollpunkten b_i nach

$$F(t) = \sum_{i=0}^{p} B_{i,p}(t) \cdot b_i \quad \text{mit} \quad 0 \leq t \leq 1 \tag{2.16}$$

Der De-Casteljau-Algorithmus ist ein numerisch stabiles, rekursives Berechnungsverfahren für die Werte einer *nicht rationalen Bézier-Kurve* zum Zeitpunkt t nach der Vorschrift

$$P_{i,0}(t) := b_i \tag{2.17}$$
$$P_{i,j}(t) := (1-t) \cdot P_{i,j-1}(t) + t \cdot P_{i+1,j-1}(t) \tag{2.18}$$

mit $j = 1, \ldots, p$ und $i = 0, \ldots, p-j$. Abbildung 2.3 zeigt eine *nicht rationale Bézier-Kurve* vom Grad $p = 4$ und das dazugehörige Kontrollpolygon.

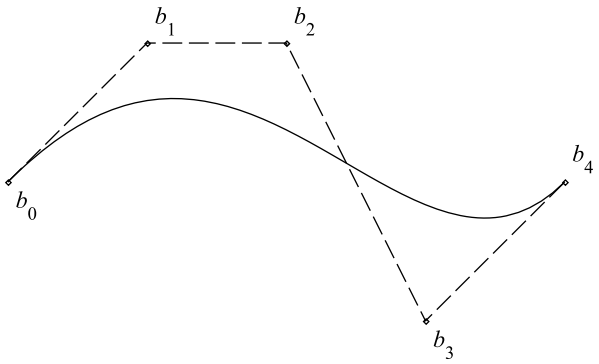


Abbildung 2.3: Nicht Rationale Bézier-Kurve (p=4)

Rationale Bézier-Kurven

Rationale Bézier-Kurven sind *Bézier-Kurven*, deren Kontrollpunkte b_i mit zusätzlichen Wichtungsfaktoren ω_i versehen werden, um damit die Kurvenverläufe zu beeinflussen. *Rationale Bézier-Kurven* sind definiert als

$$F(t) = \frac{\sum_{i=0}^{p} \omega_i \cdot B_{i,p}(t) \cdot b_i}{\sum_{i=0}^{p} \omega_i \cdot B_{i,p}(t)}. \tag{2.19}$$

Hohe Wichtungsfaktoren ziehen die Kurve näher an den entsprechenden Kontrollpunkt heran, während niedrige Wichtungsfaktoren die Kurve stärker vom Kontrollpunkt entkoppeln. Sind die Wichtungsfaktoren aller Kontrollpunkte identisch, so resultiert aus Gleichung (2.19) eine *nicht rationale Bézier-Kurve*. Abbildung 2.4 illustriert den Einfluss von verschiedenen Wichtungsfaktoren auf einer *rationalen Bézier-Kurve* vom Grad $p = 4$, anhand verschiedener Werte für Kontrollpunkt b_3.

2.1.4 BSpline-Kurven

Die in Kapitel 2.1.3 beschriebenen *Bézier-Kurven* sind zwar besonders einfach zu konstruieren, allerdings hängt der Polynomgrad der Kurve p direkt mit der An-

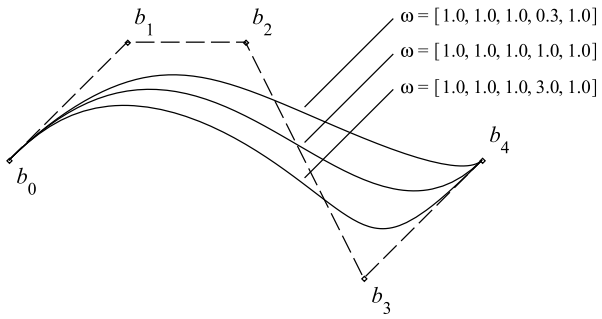


Abbildung 2.4: Rationale Bézier-Kurve (p=4)

zahl der Kontrollpunkte $(p+1)$ zusammen. Das hat den Nachteil, dass Kurven bei einer wachsenden Anzahl von Kontrollpunkten zum einen in komplexen und numerisch ineffizienten Termen resultieren und zum anderen lokale Verschiebungen von einzelnen Kontrollpunkten Einfluss auf den globalen Verlauf der gesamten Kurve haben.

BSpline-Kurven basieren auf der Idee, anstatt der global definierten *Bernsteinpolynome* abschnittsweise definierte Polynome als Basisfunktionen zu verwenden, um den Einfluss einzelner Kontrollpunktverschiebungen auf die lokale Umgebung zu beschränken und den Polynomgrad p der Kurve unabhängig von der Anzahl der Kontrollpunkte n klein halten zu können. BSpline-Kurven können aufgrund ihrer hohen Flexibilität nahezu jede Form mathematisch exakt beschreiben.

Knotenvektor

Der *Knotenvektor*

$$v = \{t_0, t_1, \ldots, t_n\} \quad \text{mit} \quad t_0 \leq t_1 \leq \ldots \leq t_n \tag{2.20}$$

spezifiziert die Definitionsbereiche der abschnittsweisen Polynome und bestimmt die Kontinuität an den Übergängen. Die Werte im *Knotenvektor* müssen aufsteigend sein, dürfen sich jedoch wiederholen. Die Anzahl der Wiederholungen eines Wertes im *Knotenvektor* nennt man *Vielfachheit*. Entspricht die *Vielfachheit* eines Wertes im Knotenvektor $p+1$, so wird der Kontrollpunkt an der entsprechenden Stelle interpoliert, bei geringerer Vielfachheit wird der Kontrollpunkt approximiert.

Knotenvektoren mit äquidistanten Abständen zwischen sich nicht wiederholenden Werten nennt man *uniforme Knotenvektoren*, bei nicht äquidistanten Abständen spricht man von *nicht-uniformen Knotenvektoren*.

BSpline-Basisfunktionen

Die *BSpline-Basisfunktionen* sind abschnittsweise definierte Polynome vom Grad p, die über das Intervall von $p+1$ benachbarten Werten im *Knotenvektor* definiert sind. Die *BSpline-Basisfunktionen* vom Grad p lassen sich anhand des *Knotenvektors* rekursiv ermitteln nach

$$N_{i,p}(t) = \frac{t - t_i}{t_{i+p} - t_i} \cdot N_{i,p-1}(t) + \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} \cdot N_{i+1,p-1}(t) \tag{2.21}$$

für $p \geq 1$ mit

$$N_{i,0}(t) = \begin{cases} 1 & t_i \leq t \leq t_{i+1} \\ 0 & \text{sonst} \end{cases} \tag{2.22}$$

Da der Nenner der Gleichung Null werden kann, definiert man $\frac{0}{0} := 0$. Abbildung 2.5 zeigt eine BSpline-Basisfunktion für einen *nicht-uniformen Knotenvektor* mit den Polynomgraden $p = 0, 1, 2, 3$.

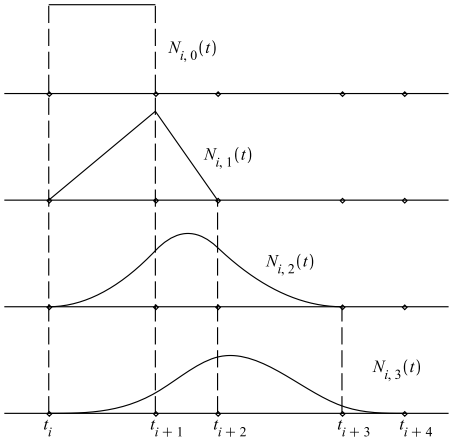


Abbildung 2.5: BSpline-Basisfunktion (p=0,1,2,3)

Nicht rationale BSpline-Kurven

Die *nicht rationale BSpline-Kurve* ergibt sich ähnlich wie die *Bézier-Kurve* aus der Summe der Produkte der Basisfunktionen $N_{i,p}$ (2.21) mit den Kontrollpunkten x_i nach

$$F(t) = \sum_{i=0}^{n} N_{i,p}(t) \cdot x_i \tag{2.23}$$

Bei *BSpline-Kurven* hängt die Anzahl der Kontrollpunkte x_i nicht direkt vom Polynomgrad p ab, sondern erfüllt die Bedingung

$$n_v = p + n + 1 \tag{2.24}$$

wobei n_v der Anzahl der Werte im *Knotenvektor* und n der Anzahl der Kontrollpunkte entspricht. *BSpline-Kurven*, deren erster und letzter Wert im *Knotenvektor* eine *Vielfachheit* von $p + 1$ besitzen, interpolieren den ersten und letzten Kontrollpunkt als Start- und Endkoordinate der Kurve. Abbildung 2.6 illustriert das Verhalten zweier *nicht rationaler BSpline-Kurven* vom Grad $p = 2$ mit identischem Kontrollpolygon aber unterschiedlichem *Knotenvektor* und unterschiedlicher *Vielfachheit*.

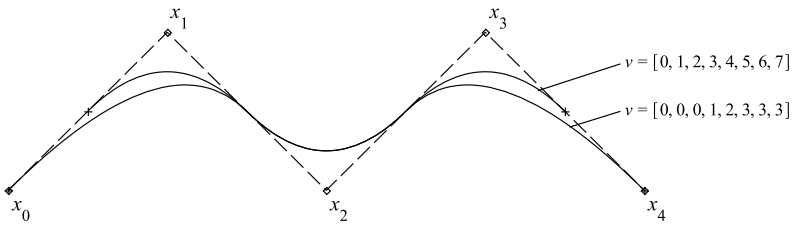


Abbildung 2.6: Nicht Rationale BSpline-Kurven (p=2)

Rationale BSpline-Kurven (NURBS)

Rationale BSpline-Kurven sind die allgemeinste Darstellung von Freiformkurven und werden im Fall von *nicht-uniformen Knotenvektoren* auch als *NURBS* (*Non-Uniform-Rational-B-Splines*) bezeichnet. *Rationale BSpline-Kurven* sind durch

$$F(t) = \frac{\sum_{i=0}^{n} \omega_i \cdot N_{i,p}(t) \cdot x_i}{\sum_{i=0}^{n} \omega_i \cdot N_{i,p}(t)} \tag{2.25}$$

definiert, wobei ω_i die zusätzlichen Wichtungsfaktoren in den jeweiligen Kontrollpunkten darstellen. Sind die Wichtungsfaktoren in allen Kontrollpunkten identisch, so resultiert daraus eine *nicht rationale BSpline-Kurve*. Durch geeignete Wahl der Wichtungsfaktoren lassen sich mit *rationalen BSpline-Kurven* viele Arten von Kurven exakt beschreiben. Für ein Kontrollpolygon $C = (P_0, P_1, P_2)$ mit dem Knotenvektor $v = [0, 0, 0, 1, 1, 1]$ und den Wichtungsfaktoren $\omega_0 = 1, \omega_1 = \eta, \omega_2 = 1$ folgt beispielsweise:

$\eta > 1$	hyperbolische Kurve
$\eta = 1$	parabolische Kurve
$\eta < 1$	elliptische Kurve
$\eta = \frac{1}{2}\sqrt{2}$	Kreiskurve
$\eta = 0$	Gerade

Abbildung 2.7 illustriert den Einfluss von unterschiedlichen Wichtungsfaktoren an zwei *rationalen BSpline-Kurven* mit identischem Kontrollpolygon und identischem Knotenvektor $v = [0, 0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 4]$. Eine Kurve beschreibt über die vier Kurvensegmente durchgehend eine Kreisbahn, während die zweite Kurve in den vier Abschnitten jeweils einen hyperbolischen, einen parabolischen, einen kreisförmigen und einen elliptischen Verlauf annimmt. Aufgrund der abschnittsweisen Beschreibung durch die *BSpline-Basisfunktionen* haben die variierenden Wichtungsfaktoren nur Einfluss auf den lokalen Verlauf der Kurve.

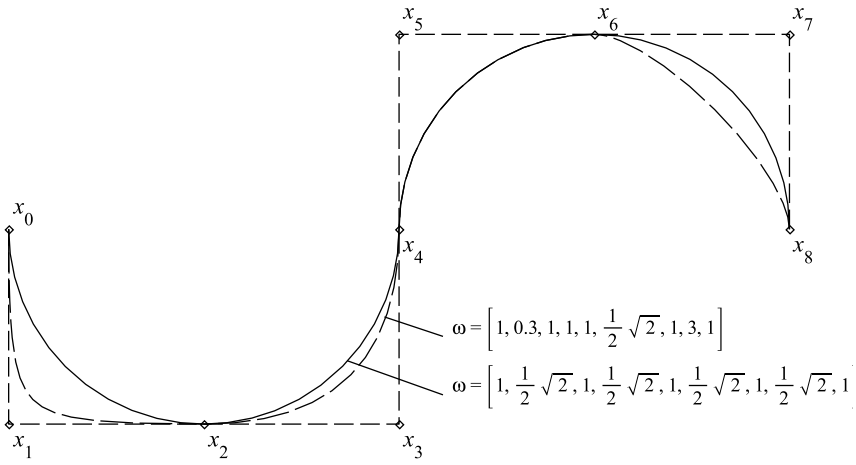


Abbildung 2.7: Rationale BSpline-Kurve (n=3)

2.2 Geometrische Beschreibung von Flächen

Geometrische Flächen sind zweidimensionale Objekte im euklidischen Raum $\mathbb{R}^3$. Flächen können sowohl eben als auch gekrümmt sein, wobei die Krümmung in jedem Punkt der Fläche von der Betrachtungsrichtung abhängt. Verschwindet die Krümmung auf einer Fläche in einer Betrachtungsrichtung, so spricht man von einfach gekrümmten Flächen. Existiert in jeder Betrachtungsrichtung eine Krümmung, so spricht man von zweifach gekrümmten Flächen. Ähnlich wie geometrische Kurven lassen sich geometrische Flächen mathematisch auf unterschiedliche Weise darstellen. Dieser Abschnitt behandelt die mathematische Beschreibung

von verschiedenen gekrümmten Flächentypen, beschränkt sich dabei jedoch auf die im Rahmen dieser Arbeit untersuchten Flächen. Diese bilden gleichzeitig das Fundament des computergestützten Konstruierens (CAD).

2.2.1 Darstellung von Flächen

Explizite Darstellung

Analog zur *expliziten Darstellung* von Kurven lassen sich Flächen als Funktion von zwei Variablen

$$z = f(x, y) \tag{2.26}$$

ansetzen. Die Fläche umfasst hierbei für zwei unabhängige Koordinaten (x, y) einen vorgegebenen Definitionsraum $B \subset \mathbb{R}^2$, die dritte Koordinate (z) bestimmt dabei für jeden Punkt des Definitionsraumes einen Punkt auf der Fläche.

Auch hier ist die Vielfalt an möglichen Formen durch die Zuordnung von genau einem Funktionswert pro Flächenkoordinate (x, y) stark eingeschränkt, daher spielen explizite Darstellungen von Flächen in der geometrischen Modellierung kaum eine Rolle.

Implizite Darstellung

Bei der *impliziten Darstellung* wird die Fläche nicht durch einen Term, sondern durch die Lösungsmenge einer Funktion

$$f(x, y, z) = 0 \tag{2.27}$$

beschrieben.

Mit der *impliziten Darstellung* lassen sich alle möglichen Flächenformen erstellen und es existieren einfache Kriterien dafür, ob ein Punkt innerhalb, außerhalb oder genau auf der zu beschreibenden Fläche liegt (siehe z.B. [30, 48]). Die *implizite Darstellung* von Flächen ist jedoch genau wie die *implizite Darstellung* von Kurven uneindeutig und somit nicht für alle geometrischen Problemstellungen geeignet.

Parameterdarstellung

Bei der *Parameterdarstellung* der Fläche werden die Koordinaten einzeln als Funktionen

$$x = f_x(u, v) \tag{2.28a}$$
$$y = f_y(u, v) \tag{2.28b}$$
$$z = f_z(u, v) \tag{2.28c}$$

von zwei Laufparametern u und v bestimmt. Die beiden Laufparameter durchlaufen einen vorgegebenen Definitionsraum $B \subset \mathbb{R}^2$ und die Koordinaten (x, y, z) werden in jedem Parameterpunkt (u, v) einzeln über reelle Funktionen bestimmt.

In der *Parameterdarstellung* können Flächen sehr leicht polygonisiert und damit näherungsweise grafisch dargestellt werden. Zudem lassen sich viele Flächentypen lediglich in der *Parameterdarstellung* beschreiben. Daher ist diese Darstellung wie bei den Kurven die geeignetste für die computergestützte Beschreibung und Verarbeitung von geometrischen Flächen.

Flächentypen und ihre Darstellung

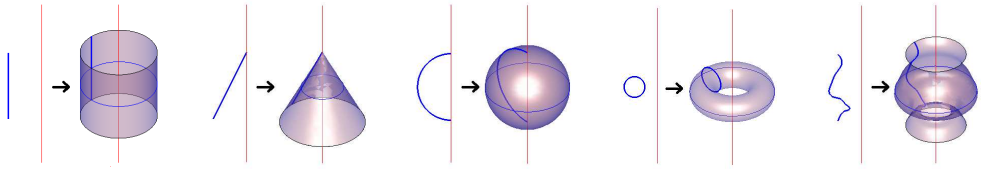
Wie bei den Kurven lässt sich die *explizite Darstellung* einer Fläche einfach in eine *implizite Darstellung* umwandeln, sowie die *implizite Darstellung* jederzeit in die *Parameterform* überführen. Der umgekehrte Weg ist jedoch auch hier nicht immer möglich bzw. eindeutig. Daher basieren die im Laufe dieser Arbeit entwickelten Algorithmen ausschließlich auf der *Parameterdarstellung*. Tabelle 2.2 zeigt die hier behandelten Flächentypen und die jeweiligen Darstellungsmöglichkeiten.

Fläche	Explizit	Implizit	Parametrisch
Zylinder	X	X	X
Kegel	X	X	X
Kugel	X	X	X
Torus	X	X	X
Rotationssymmetrische Fläche	-	-	X
Extrusions Fläche	-	-	X
Bezier-Fläche	-	-	X
BSpline-Fläche	-	-	X

Tabelle 2.2: Flächentypen und ihre Darstellungsarten

2.2.2 Rotationsflächen

Rotationsflächen bezeichnen Flächen, die durch Rotation einer Kurve um eine beliebige Achse erzeugt werden. Die Erzeugerkurve kann dabei gerade (*Zylinder*, *Kegel*), kreisförmig (*Kugel*, *Torus*) oder beliebig geformt (*Rotationssymmetrische Fläche*) sein (siehe Abbildung 2.8).



(a) Zylinder (b) Kegel (c) Kugel (d) Torus (e) Rotationsfl.

Abbildung 2.8: Rotationsflächen

Zylinder

Der *Zylinder* gehört zu den einfach gekrümmten Flächen und entsteht durch die Rotation einer Geraden um eine dazu parallele Rotationsachse. Ein *Zylinder* mit Mittelpunkt im Koordinatenursprung und Radius r lässt sich über die *implizite Darstellung*

$$f(x, y, z) = x^2 + y^2 - r^2 = 0 \tag{2.29}$$

beschreiben. Die *Parameterdarstellung* des *Zylinders* lautet

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} r \cdot \cos(u) \\ r \cdot \sin(u) \\ v \end{pmatrix} \quad \text{mit} \quad 0 \leq u \leq 2\pi, -\infty < v < \infty. \tag{2.30}$$

Kegel

Der *Kegel* beschreibt ebenfalls eine einfach gekrümmte Fläche und resultiert aus der Rotation einer Geraden um eine Rotationsachse, die weder parallel noch senkrecht zu ihr steht. Den Winkel zwischen Rotationsachse und Gerade nennt man Öffnungswinkel (γ), der Schnittpunkt zwischen Gerade und Rotationsachse heißt Scheitelpunkt. Ein *Kegel* mit Scheitelpunkt im Koordinatenursprung und Öffnungswinkel γ erfüllt die *implizite* Gleichung

$$f(x, y, z) = x^2 + y^2 - (\sin(\gamma) \cdot z)^2 = 0 \tag{2.31}$$

Die *Parameterdarstellung* des *Kegels* lautet

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} z \cdot \sin(\gamma) \cdot \cos(u) \\ z \cdot \sin(\gamma) \cdot \sin(u) \\ v \end{pmatrix} \quad \text{mit} \quad 0 \leq u \leq 2\pi, -\infty < v < \infty. \tag{2.32}$$

Kugel

Die *Kugel* gehört zu den doppelt gekrümmten Flächen und resultiert aus der Rotation eines Halbkreises mit Start- und Endpunkt auf der Rotationsachse. Eine *Kugel* mit Mittelpunkt im Koordinatenursprung und Radius r erfüllt die *implizite* Gleichung

$$f(x, y, z) = x^2 + y^2 + z^2 - r^2 = 0 \tag{2.33}$$

Die *Parameterdarstellung* der *Kugel* lautet

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} r \cdot \sin(u) \cdot \cos(v) \\ r \cdot \sin(u) \cdot \sin(v) \\ r \cdot \cos(u) \end{pmatrix} \quad \text{mit} \quad 0 \leq u \leq \pi, 0 \leq v \leq 2\pi. \tag{2.34}$$

Torus

Der *Torus* gehört ebenfalls zu den doppelt gekrümmten Flächen und resultiert aus der Rotation eines Vollkreises um eine Rotationsachse, die ihn nicht schneidet. Den Radius des Vollkreises nennt man den kleinen Radius (r_{min}), den Abstand des Kreismittelpunktes von der Rotationsachse den großen Radius (r_{max}). Ein *Torus* mit kleinem Radius r_{min} und großem Radius r_{max} erfüllt die *implizite* Gleichung

$$f(x, y, z) = (x^2 + y^2 + z^2 + {r_{max}}^2 - {r_{min}}^2)^2 - 4 \cdot {r_{max}}^2 \cdot (x^2 + y^2) = 0 \tag{2.35}$$

Die *Parameterdarstellung* des *Torus* lautet

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} (r_{max} + r_{min} \cdot \cos(u)) \cdot \cos(v) \\ (r_{max} + r_{min} \cdot \cos(u)) \cdot \sin(v) \\ r_{min} \cdot \sin(u) \end{pmatrix} \quad \text{mit} \quad 0 \leq u \leq 2\pi, 0 \leq v \leq 2\pi \tag{2.36}$$

Rotationssymmetrische Fläche

Eine *rotationssymmetische Fläche* resultiert aus der Rotation einer beliebigen Rotationskurve C um eine Rotationsachse z. Die Rotationskurve kann dabei durch jede mögliche organische Kurve definiert sein und resultiert in der *Parameterdarstellung*

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} f_x(u) \cdot \cos(v) \\ f_y(u) \cdot \sin(v) \\ f_z(u) \end{pmatrix} \quad \text{mit} \quad t_0 \leq u \leq t_1, 0 \leq v \leq 2\pi \tag{2.37}$$

mit $f_x(t)$, $f_y(t)$ und $f_z(t)$ als *Parameterfunktionen* der Rotationskurve.

2.2.3 Extrusionsflächen

Extrusionsflächen entstehen durch Parallelverschiebung einer beliebigen Kurve C (Extrusionskurve) im Raum entlang einer zweiten Kurve L (Leitkurve). *Extrusionsflächen* besitzen die *Parameterdarstellung*

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} f_{C,x}(u) + f_{L,x}(v) \\ f_{C,y}(u) + f_{L,y}(v) \\ f_{C,z}(u) + f_{L,z}(v) \end{pmatrix} \quad \text{mit} \quad t_{C,0} \leq u \leq t_{C,1}, t_{L,0} \leq v \leq t_{L,1} \tag{2.38}$$

mit $f_{C,x}(t)$, $f_{C,y}(t)$ und $f_{C,z}(t)$ als *Parameterdarstellung* der Extrusionskurve und $f_{L,x}(t)$, $f_{L,y}(t)$ sowie $f_{L,z}(t)$ als *Parameterdarstellung* der Leitkurve. Abbildung 2.9 zeigt eine Extrusionsfläche resultierend aus der Extrusion einer *BSpline-Kurve* entlang einer zweiten *BSpline-Kurve*.

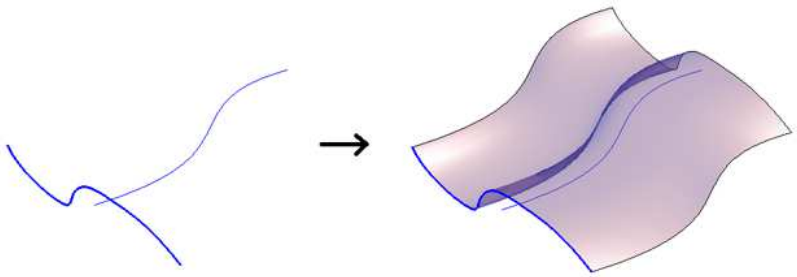


Abbildung 2.9: Extrusionsfläche

2.2.4 Bezier-Flächen

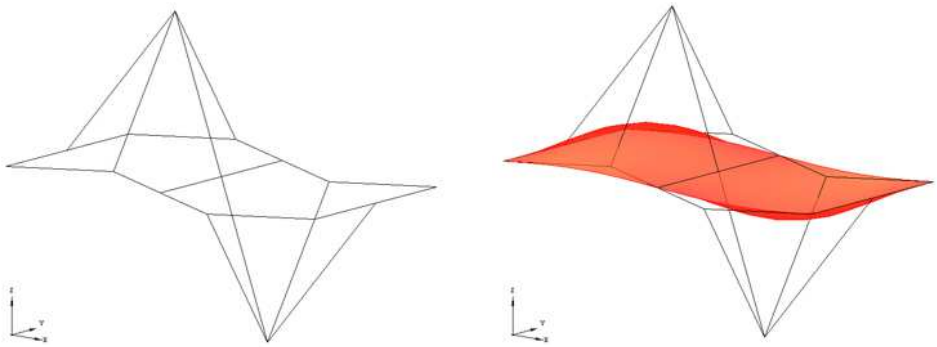
Die in Kapitel 2.1.3 eingeführten *Bézier-Kurven* können auch zur Definition von gekrümmten Flächen im Raum verwendet werden. Eine *Bézier-Fläche* ist eine Tensorprodukt-Fläche vom Grad p und q, die über ein Kontrollpunkt-Gitter von $(p+1) \times (q+1)$ Kontroll- oder Bézier-Punkten aufgespannt wird. Die Kontrollpunkte bilden dabei auf der Parameterebene $\mathbb{R}^2$ ein kartesisches Gitter. Im euklidischen Raum $\mathbb{R}^3$ werden die vier Eckpunkte des Gitters als Begrenzungspunkte der Fläche interpoliert und die inneren Kontrollpunkte approximiert.

Nicht Rationale Bézier-Fläche

Eine *nicht rationale Bézier-Fläche* ergibt sich aus der Summe der Tensorprodukte der $(p+1) \times (q+1)$ *Bernsteinpolynome* $B_{i,p} \cdot B_{j,q}$ zu gegebenen $(p+1) \times (q+1)$ Kontrollpunkten $b_{i,j}$ nach

$$\mathbf{F}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix} = \sum_{i=0}^{p} \sum_{j=0}^{q} B_{i,p}(u) \cdot B_{j,q}(v) \cdot b_{i,j}. \tag{2.39}$$

Abbildung 2.10 zeigt eine *nicht rationale Bézier-Fläche* vom Grad $p = 4$ und $q = 2$ zu einem gegebenen Kontrollpunkt-Gitter.



(a) Kontrollpunkt-Gitter (b) Bézier-Fläche

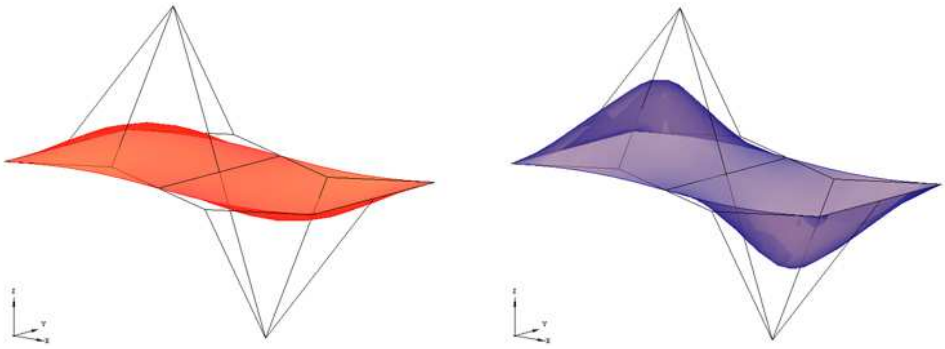
Abbildung 2.10: Nicht Rationale Bézier-Fläche (p=4, q=2)

Rationale Bézier-Fläche

Die *rationale Bézier-Fläche* besitzt zu den gegebenen $(p+1) \times (q+1)$ Kontrollpunkten $b_{i,j}$ zusätzliche Wichtungsfaktoren $\omega_{i,j}$ und ist definiert als

$$\mathbf{F}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix} = \frac{\sum_{i=0}^{p}\sum_{j=0}^{q}\omega_{i,j}\cdot B_{i,p}(u)\cdot B_{j,q}(v)\cdot b_{i,j}}{\sum_{i=0}^{p}\sum_{j=0}^{q}\omega_{i,j}\cdot B_{i,p}(u)\cdot B_{j,q}(v)}. \tag{2.40}$$

Sind die Wichtungsfaktoren aller Kontrollpunkte identisch, so resultiert aus Gleichung 2.40 eine *nicht rationale Bézier-Fläche*. Eine Änderung eines einzelnen Wichtungsfaktors hat wie bei den *Bezier-Kurven* Einfluss auf den Verlauf der gesamten Fläche: Ein größerer Wichtungsfaktor zieht die Fläche näher an den entsprechenden Punkt heran. Abbildung 2.11 illustriert diesen Effekt an zwei Flächen vom Grad $p = 4$ und $q = 2$ mit identischem Kontrollpunkt-Gitter aber unterschiedlichen Wichtungsfaktoren.



(a) Unifrome Gewichtung (b) Unterschiedliche Gewichtung

Abbildung 2.11: Rationale Bézier-Flächen (p=4, q=3)

2.2.5 BSpline-Flächen

BSpline-Flächen sind Tensorprodukt-Flächen zu den abschnittsweise definierten *BSpline-Basisfunktionen* aus Kapitel 2.1.4. *BSpline-Flächen* besitzen gegenüber *Bézier-Flächen* dieselben vorteilhaften Eigenschaften wie die entsprechenden Kurven: Sie entkoppeln die Polynomgrade p und q von der Anzahl der Kontrollpunkte $n \times m$ und sie begrenzen den Einfluss einzelner Kontrollpunkt-Verschiebungen auf deren unmittelbare lokale Umgebung.

Nicht Rationale BSpline-Flächen

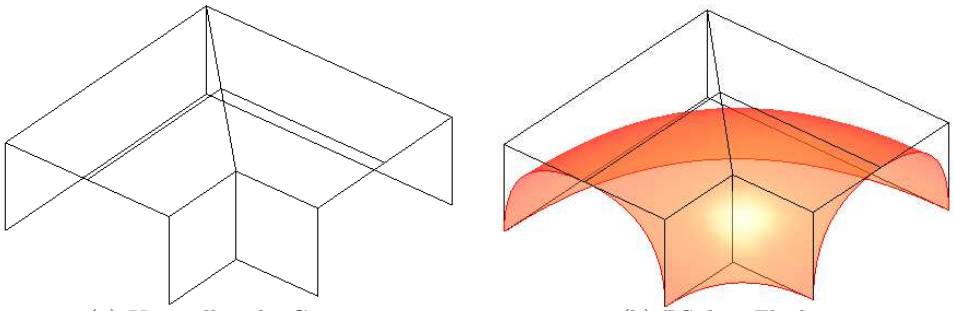
Die *nicht rationale BSpline-Fläche* vom Grad p und q zu $n \times m$ Kontrollpunkten ergibt sich aus der Summe der Tenorprodukte der Basisfunktionen $N_{i,p} \cdot N_{j,q}$ mit den Kontrollpunkten $x_{i,j}$ nach

$$\mathbf{F}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix} = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u) \cdot N_{j,q}(v) \cdot x_{i,j} \tag{2.41}$$

Die Anzahl der Kontrollpunkte $x_{i,j}$ erfüllt die Bedingung

$$n_v = p + n + 1 \tag{2.42}$$
$$n_w = q + m + 1 \tag{2.43}$$

mit n_v und n_w als Anzahl der Werte in den beiden *Knotenvektoren* und $n \times m$ als Anzahl der Kontrollpunkte. Analog zu den *BSpline-Kurven* bestimmt die *Vielfachheit* der Werte in den beiden *Knotenvektoren* die Kontinuität in den Flächen-Übergängen und das Interpolationsverhalten an den Randpunkten und im Inneren. Abbildung 2.12 zeigt eine *nicht rationale BSpline-Fläche* vom Grad $p = 2$ und $q = 2$.



(a) Kontrollpunkt-Gitter (b) BSpline-Fläche

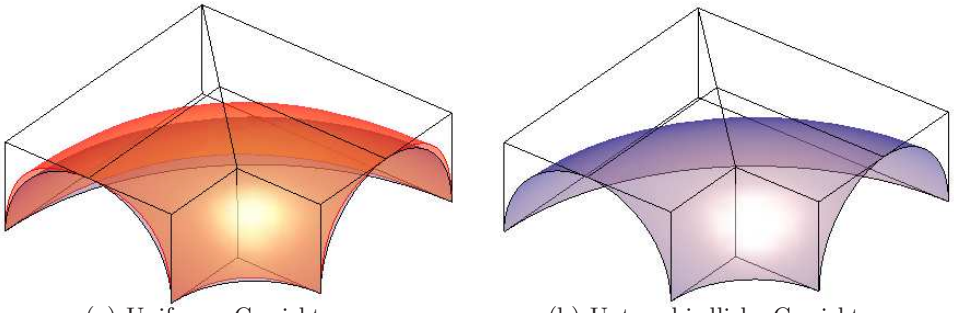
Abbildung 2.12: Nicht Rationale BSpline-Flächen (p=2,q=2)

Rationale BSpline-Flächen (NURBS)

Rationale BSpline-Flächen oder *NURBS-Flächen* besitzen zu jedem Kontrollpunkt einen Wichtungsfaktor $\omega_{i,j}$ und lassen sich definieren als

$$\mathbf{F}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix} = \frac{\sum_{i=0}^{n} \sum_{j=0}^{m} \omega_{i,j} \cdot N_{i,p}(u) \cdot N_{j,q}(v) \cdot x_{i,j}}{\sum_{i=0}^{n} \sum_{j=0}^{m} \omega_{i,j} \cdot N_{i,p}(u) \cdot N_{j,q}(v)}. \tag{2.44}$$

Über die Wahl von geeigneten Wichtungsfaktoren lassen sich analog zu den Kurven verschiedene Krümmungen definieren. Abbildung 2.13 zeigt den Einfluss von unterschiedlichen Wichtungsfaktoren in einzelnen Kontrollpunkten auf die erzeugte Fläche.



(a) Uniforme Gewichtung (b) Unterschiedliche Gewichtung

Abbildung 2.13: Rationale BSpline-Flächen (p=2,q=2)

2.3 Geometrische Volumenmodelle

Geometrische Volumenmodelle beschreiben dreidimensionale Körper im euklidischen Raum $\mathbb{R}^3$. Die zu beschreibenden Volumenkörper besitzen dabei eine geschlossene und orientierte Oberfläche und haben die gemeinsame Eigenschaft, dass für jeden beliebigen Punkt eindeutig die Aussage getroffen werden kann, ob sich der Punkt innerhalb, außerhalb oder auf der Oberfläche des Körpers befindet. Volumenkörper können auf verschiedene Weise modelliert werden, dieser Abschnitt beschreibt die im Rahmen dieser Arbeit wichtigsten Modellierungsmethoden.

2.3.1 Direkte Darstellungsschemata

Bei den direkten Darstellungsschemata werden komplexe Volumenkörper aus einfachen Grundkörpern zusammen gesetzt. Das hat den Vorteil, dass keine „unsinnigen“ Körper konstruiert werden können und eine mathematisch exakte Repräsentation der Körper existiert. Dieser Abschnitt erklärt die wichtigsten direkten Darstellungsschemata mit ihren Eigenschaften.

Konstruktive Festkörpergeometrie (CSG)

In der konstruktiven Festkörpergeometrie (CSG) werden komplexe Körper durch die Verknüpfung von einfachen Grundkörpern (z.B. Quader, Zylinder, Kegel, Kugel, Torus, . . .) mit booleschen Operatoren (Vereinigung $\cup$, Schnitt $\cap$, Differenz $\setminus$) erzeugt. Der Bauplan für ein CSG-Modell entspricht einer baumartigen Anweisungskette, die zur Laufzeit ausgewertet wird. Abbildung 2.14 zeigt ein Beispiel für ein CSG-Modell samt Konstruktionsbaum.

Durch geschickte Verknüpfung lassen sich mit CSG zwar sehr komplexe Körper modellieren, die Beschränkung auf relativ einfach zu beschreibende Grundformen geben der klassischen CSG jedoch geometrische Grenzen vor. Dafür bieten die meisten CSG-basierten Konstruktionsprogramme die Möglichkeit, einzelne Grundkörper oder einzelne boolsche Operatoren nachträglich zu verändern, um durch Versuch-und-Irrtum-Verfahren die finalen Modelle zu kreieren. Diese Ei-

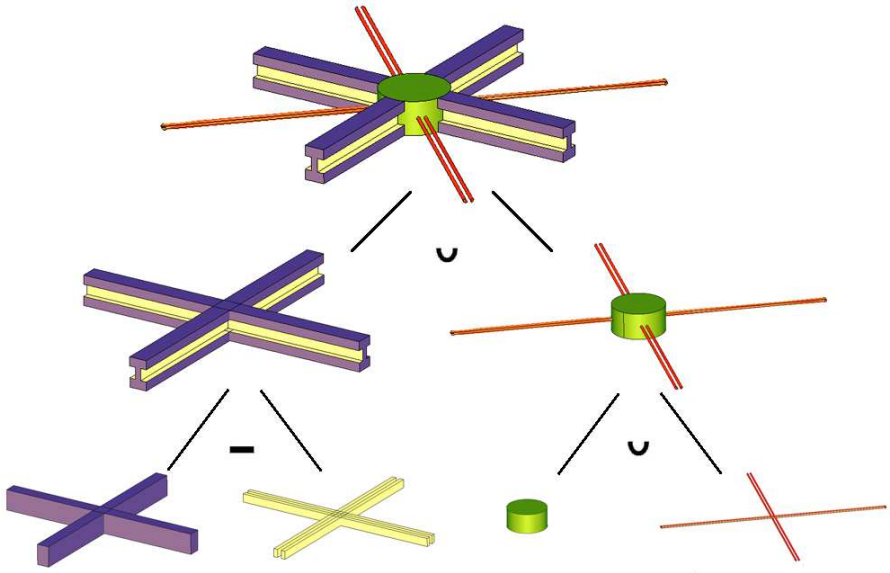


Abbildung 2.14: CSG-Modell mit Konstruktionsbaum

genschaft macht CSG-basierte Konstruktionssysteme bei Designern zu einem sehr beliebten Werkzeuge.

Normzellenschema

Das Normzellenschema unterteilt den Definitionsraum des zu beschreibenden Körpers in ein Gitter aus gleich großen Zellen („Normzellen“). Die einzelnen Zellen sind meist Würfel (Voxel), können aber auch andere Formen annehmen (z.B. Quader, Prismen, etc.).

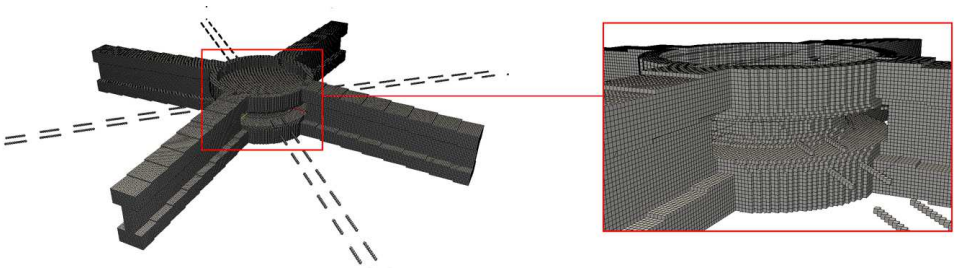


Abbildung 2.15: Normzellenschema

Normzellenschemata lassen sich als einfache Bit-Matrizen implementieren und sind dadurch unkompliziert zu benutzen. So lässt sich die Position einer Zelle im Speicher einfach in die Position im Raum umrechnen und umgekehrt. Das gleiche gilt für die Bestimmung der Nachbarzellen. Durch eine Zellverfeinerung lässt sich eine beliebig genaue Annäherung auch an die wahre Geometrie erreichen (vgl. Abbildungn 2.15), diese Annäherung ist bei gekrümmten Objekten in der Praxis

jedoch nie exakt. Da bei dieser Darstellungsvariante innerhalb des zu beschreibenden Volumens alle Zellen gleich groß sind, erhöht sich der Speicheraufwand bei steigenden Genauigkeitsanforderungen erheblich. Zudem geht die Information über die Oberflächenkrümmung innerhalb des Normzellenschemas vollständig verloren.

Oktalbäume

Oktalbäume sind baumartige Datenstrukturen, die ähnliche Eigenschaften wie die Normzellen besitzen. Der zu beschreibende Körper wird genau wie im Normzellenschema in ein Gitter aus Zellen aufgeteilt, allerdings haben diese Zellen keine einheitliche Größe, sondern werden so groß gewählt, wie es für eine ausreichende Approximation an der entsprechenden Stelle notwendig ist.

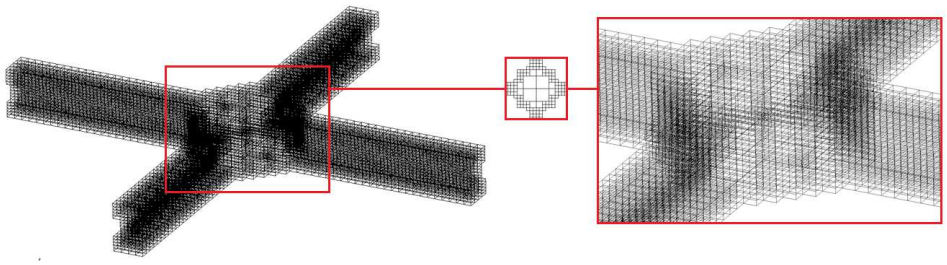


Abbildung 2.16: Oktalbaum

Oktalbäume lassen sich im Gegensatz zu Normzellen sehr speichereffizient beschreiben, allerdings ist eine einfache, allgemeingültige Bestimmung der Nachbarschaftsbeziehungen nicht mehr möglich. Daher sind Oktalbäume zwar nicht immer gut geeignet, um geometrische Modelle für numerische Simulationen direkt zu beschreiben, allerdings lassen sich diese einfach und unkompliziert in eine Vielzahl von geometrischen Algorithmen integrieren und sind somit als unterstützendes Modell ein nicht mehr weg zu denkender Bestandteil von vielen numerischen Simulationsalgorithmen.

2.3.2 Indirekte Darstellungsschemata

Bei den indirekten Darstellungsschemata werden Volumenkörper *indirekt* über die Oberfläche des Körpers modelliert. Geometrische Operationen werden nicht am Volumenkörper direkt ausgeführt, sondern auf den einzelnen Oberflächensegmenten. Auf diese Weise lassen sich beliebig geformte Körper geometrisch exakt beschreiben. Dieser Abschnitt erklärt die im Rahmen dieser Arbeit wichtigsten indirekten Darstellungsschemata.

Randdarstellungsschema (B-Rep)

Das Randdarstellungsschema (B-Rep) ist ein extrem leistungsfähiges und genaues Konzept zur effizienten Beschreibung von komplexen Körpern. In ihm wird die Begrenzung eines Körper über eine hierarchische Struktur von Punkten, Kanten

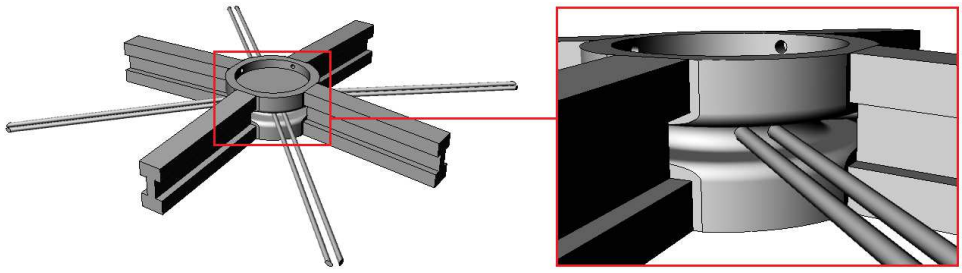


Abbildung 2.17: Knotenmodell

und Flächen definiert. Damit das Volumen geschlossen ist, teilen sich benachbarte Flächen gemeinsame Kanten und Punkte. Die Form der einzelnen Kanten und Flächen ist dabei beliebig und kann jeden geometrischen Kanten- und Flächentyp aus Abschnitt 2.1 und 2.2 annehmen. In der computergestützten Beschreibung von B-Rep Modellen werden die Nachbarschaftsbeziehungen der einzelnen Objekte (Topologie) separat von ihrer Form (Geometrie) beschrieben. Dadurch lassen sich die Modelle leicht und flexibel darstellen und weiter verarbeiten. Abbildung 2.18 zeigt die topologische und geometrische Hierarchie des Randdarstellungsschemas

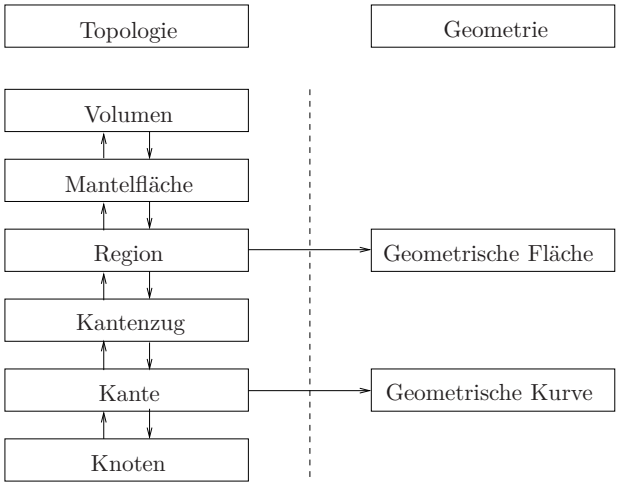


Abbildung 2.18: Topologische und geometrische Hierarchie des Randdarstellungsschemas

Jedes Volumen besitzt einen äußeren Rand, beschrieben durch einen geschlossenen Satz von Flächen. Weist ein Volumenkörper Hohlräume auf, so werden diese ebenfalls durch geschlossene Oberflächen beschrieben. Somit haftet dem Volumen in der Randdarstellung pro Hohlraum eine weitere innere Randbeschreibung an.

Die einzelnen Oberflächen besitzen jeweils einen äußeren Rand bestehend aus einem geschlossenen Kantenzug und können im Inneren Löcher aufweisen. Analog

zu den Hohlräumen bei Volumenkörpern wird jedes Loch durch eine weitere innere Randbeschreibung aus zusammenhängenden Kanten dargestellt.

Abbildung 2.19 illustriert den Aufbau eines Volumenmodels im Randdarstellungsschema nach Unterteilung in seine einzelnen Grundobjekte.

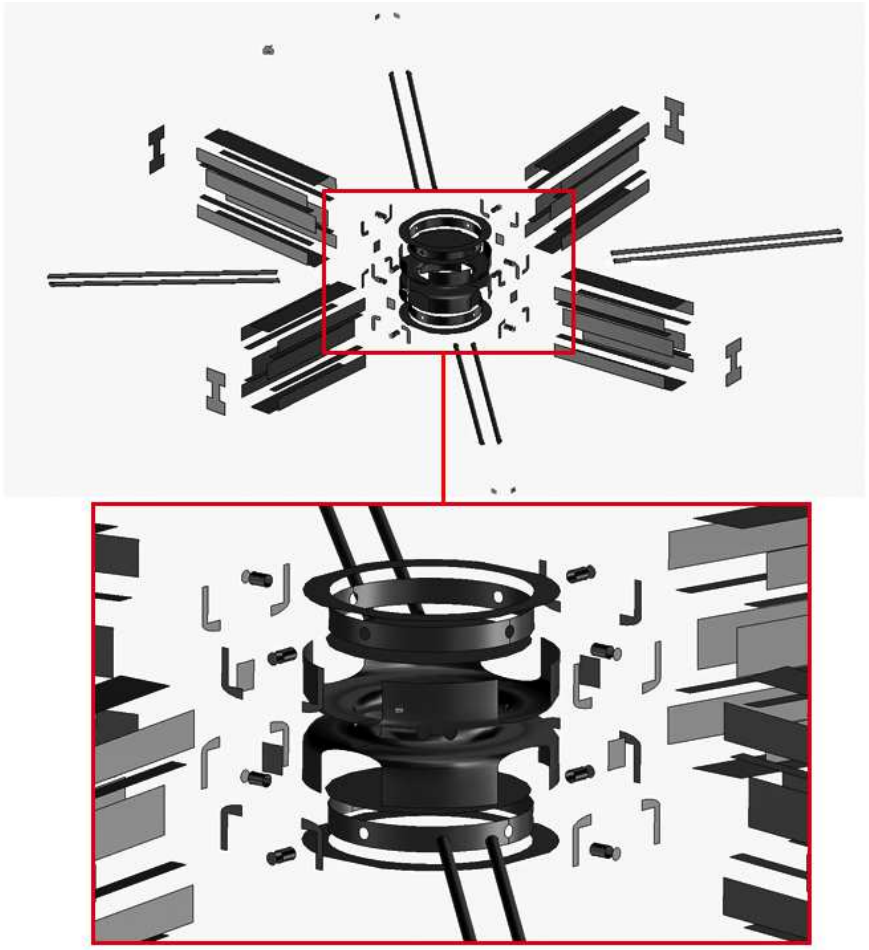


Abbildung 2.19: Randdarstellungsschema

Facettenmodell

Das Facettenmodell ist ein Spezialfall des Randdarstellungsschemas, bei dem die Oberfläche des Körpers aus Dreiecken und/oder Vierecken zusammen gesetzt ist. Bei gekrümmten Flächen kann das Facettenmodell den zu beschreibenden Körper geometrisch nicht exakt repräsentieren, durch Verfeinerung des Oberflächennetzes kann jedoch eine beliebige Genauigkeit erreicht werden. Facettenmodelle haben zwar einen hohen Speicherbedarf, allerdings sind geometrische Operationen auf den facettierten Flächen wesentlich leichter zu handhaben.

Abbildung 2.20 zeigt das Facettenmodell eines Volumenkörpers. Die Anzahl der Facetten ist dabei lokal an die Krümmung der Oberfläche angepasst.

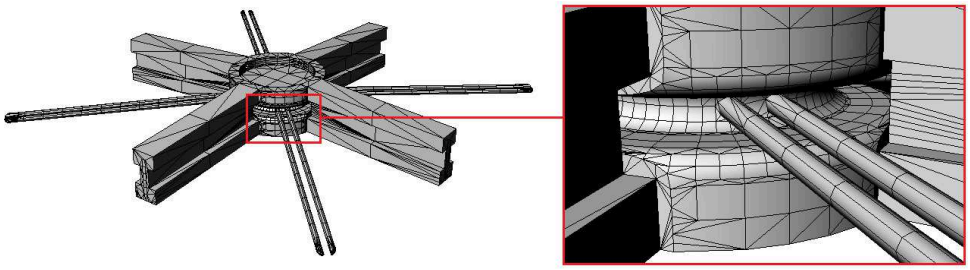


Abbildung 2.20: Facettenmodell

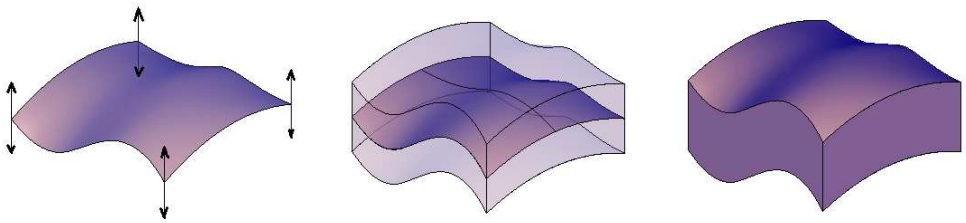
2.3.3 Extrusionsmodelle

Extrusionsmodelle beschreiben Volumenkörper über dimensionsreduzierte Mittelflächen, welche durch Verschiebung entlang einer Leitkurve ein Volumen erzeugen. Die so geformten Extrusionskörper besitzen dabei topologisch identische Ober- und Unterflächen. Somit sind einfache Extrusionsmodelle nur für eine kleine Gruppe von geometrischen Modellierungsaufgaben geeignet wie z.B. der Beschreibung von dünnwandigen Strukturen.

Im Rahmen dieser Arbeit wurden drei verschiedene Extrusionstypen eingeführt, die für eine anschließende Vernetzung auf unterschiedliche Art und Weise miteinander kombiniert werden können (vgl. Kapitel 6.3). Durch eine geeignete Kombination der Extrusionstypen konnten für eine Vielzahl von geometrischen Problemstellungen Lösungen gefunden werden. Dieser Abschnitt beschreibt die drei Extrusionstypen mit ihren grundlegenden Eigenschaften.

Extrusionstyp 1

Der erste Extrusionstyp bestimmt den Extrusionskörper über eine Mittelfläche und einem konstanten Vektor, der die Dicke und die Extrusionsrichtung definiert. Die Mittelfläche wird in jedem Punkt konstant entlang der vorgegebenen Extrusionsrichtung verschoben, so dass der auf diese Weise erzeugte Volumenkörper parallele Ober- und Unterflächen besitzt. Abbildung 2.21 zeigt ein Beispiel für ein Extrusionsmodell vom Extrusionstyp 1.

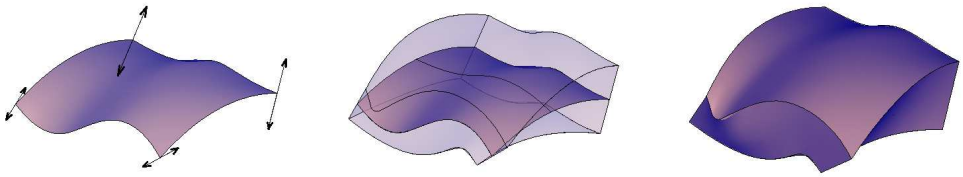


(a) Mittelfläche (b) Extrusionsmodell (c) Volumenkörper

Abbildung 2.21: Extrusionstyp 1

Extrusionstyp 2

Der zweite Extrusionstyp beschreibt den Extrusionskörper über eine Mittelfläche und einem Skalar als konstante Dicke. Die Extrusionsrichtung wird hierbei für jeden Punkt auf der Mittelfläche über den Flächennormalenvektor definiert. Der so erzeugte Extrusionskörper besitzt Seitenflächen, die senkrecht zum Rand der Mittelfläche stehen (vgl. Abbildung 2.22).

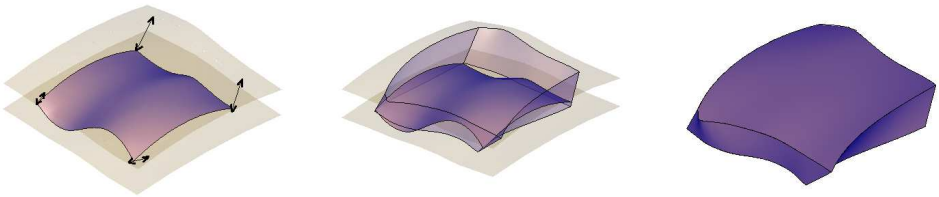


(a) Mittelfläche (b) Extrusionsmodell (c) Volumenkörper

Abbildung 2.22: Extrusionstyp 2

Extrusionstyp 3

Der dritte Extrusionstyp wurde entwickelt, um Extrusionskörper mit variierender Dicke modellieren zu können. Das finale Volumen wird bei dieser Variante über eine Mittelfläche, eine Ober- und eine Unterfläche definiert. Der Extrusionskörper entsteht durch Extrusion der Mittelfläche entlang der Flächennormalen wie bei Extrusionstyp 2, die Extrusionsdicke wird jedoch für jeden Punkt auf der Mittelfläche individuell durch eine Verschneidung des Normalenvektors mit der Ober- und Unterfläche festgelegt. Durch eine geeignete Modellierung der drei Referenzflächen lassen sich Extrusionskörper mit individuellen Ober- und Unterflächenformen einfach und effektiv beschreiben (vgl. Abbildung 2.23).



(a) Referenz-Flächen (b) Extrusionsmodell (c) Volumenkörper

Abbildung 2.23: Extrusionstyp 3

2.3.4 Hybridmodelle

Hybridmodelle werden eingesetzt, wenn keines der gezeigten Modelle für alle Anwendungen gleich gut geeignet ist. Sie beschreiben den zu modellierenden Körper mit mehreren unterschiedlichen Modellen gleichzeitig. Dabei muss auf die Konsistenz der Daten zwischen den verschiedenen Darstellungsarten geachtet werden. Um von einem Darstellungsschema in ein anderes zu wechseln sind entsprechende Konvertierungsalgorithmen notwendig [29, 64]. Jede Konvertierung ist in der Regel mit Genauigkeitsverlust verbunden und nicht alle Objekte können in jedes

Darstellungsschema überführt werden. Daher wird bei Hybridmodellen oft eine Primärdatenstruktur und eine Sekundärdatenstruktur verwendet.

Moderne Konstruktionsprogramme bieten in der Regel mehrere Varianten zur Modellierung von geometrischen Objekten an, wobei die beiden beliebtesten Konstruktionssysteme die Konstruktive Festkörpergeometrie (CSG) und das Randdarstellungsschema (B-Rep) sind. Da sich jedes CSG-Modell einfach in ein B-Rep Modell konvertieren lässt, jedoch nicht jedes Modell in umgekehrter Richtung konvertiert werden kann, bedient das Randdarstellungsschema das breiteste Spektrum an Anwendungen.

2.4 Einführung in die Differentialgeometrie

Die Differentialgeometrie stellt als Teilgebiet der Mathematik die Synthese von Analysis und Geometrie dar. Sie beschäftigt sich sowohl mit Kurven als auch mit zweidimensionalen gekrümmten Flächen im euklidischen Raum $\mathbb{R}^3$. Mit Hilfe der Differentialgeometrie ist es möglich, wichtige geometrische Eigenschaften wie die Krümmung oder die Abstände zwischen beliebigen Punkten auf einer gekrümmten Oberfläche auch quantitativ zu erfassen. Dieser Abschnitt beschreibt die für diese Arbeit wichtigsten differentialgeometrischen Eigenschaften von Kurven und Flächen, basierend auf den Ausführungen von [31, 50, 27].

2.4.1 Vektorrechnung

Für eine differentialgeometrische Betrachtung von geometrischen Objekten werden die zu untersuchenden Kurven und Flächen nach Abschnitt 2.1.1 und 2.2.1 in der *Parameterdarstellung* angesetzt. Dadurch können die modellierten Objekte als Vektorfunktionen formuliert werden. In diesem Abschnitt werden zunächst die wichtigsten Vektoroperationen eingeführt. Auf diesen aufbauend werden anschließend die wichtigsten differentialgeometrischen Eigenschaften der parametrischen Kurvenansätze ($\mathbb{R} \to \mathbb{R}^3$) und Flächenansätze ($\mathbb{R}^2 \to \mathbb{R}^3$) aufgezeigt. Dem euklidischen Vektorraum $\mathbb{R}^3$ liegt die kartesische Basis

$$\mathbf{e}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \mathbf{e}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \mathbf{e}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \tag{2.45}$$

zugrunde. Jeder Vektor $a \in \mathbb{R}^3$ lässt sich somit als Linearkombination der drei Basisvektoren darstellen.

$$\mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = a_1 \cdot \mathbf{e}_1 + a_2 \cdot \mathbf{e}_2 + a_3 \cdot \mathbf{e}_3 \tag{2.46}$$

Die Addition von zwei Vektoren und die Multiplikation mit einem Skalar ($\mathbb{R}^3 \rightarrow \mathbb{R}^3$) sind durch die folgenden Vorschriften definiert

$$\mathbf{a} + \mathbf{b} = (a_1 + b_1) \cdot \mathbf{e}_1 + (a_2 + b_2) \cdot \mathbf{e}_2 + (a_3 + b_3) \cdot \mathbf{e}_3 \tag{2.47}$$
$$\lambda \cdot \mathbf{a} = (\lambda \cdot a_1) \cdot \mathbf{e}_1 + (\lambda \cdot a_2) \cdot \mathbf{e}_2 + (\lambda \cdot a_3) \cdot \mathbf{e}_3 \tag{2.48}$$

Das Skalarprodukt von zwei Vektoren ($\mathbb{R}^3 \cdot \mathbb{R}^3 \rightarrow \mathbb{R}$) ist definiert als

$$\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + a_3 b_3 \tag{2.49}$$

Der Betrag eines Vektors ($\mathbb{R}^3 \rightarrow \mathbb{R}$) berechnet sich nach

$$|\mathbf{a}| = \sqrt{\mathbf{a} \cdot \mathbf{b}} = \sqrt{a_1 a_1 + a_2 a_2 + a_3 a_3} \tag{2.50}$$

Jede quadratische Matrix $A \in \mathbb{R}^{n \times n}$ besitzt eine Determinante $\det(A)$. Diese ist eine reelle Zahl, welche für eine Matrix im euklidischen Raum $A \in \mathbb{R}^{3 \times 3}$ nach folgender Vorschrift gebildet wird

$$\det \mathbf{A} = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = \begin{matrix} +a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} \\ -a_{13}a_{22}a_{31} - a_{11}a_{23}a_{32} - a_{12}a_{21}a_{33} \end{matrix} \tag{2.51}$$

Das Kreuzprodukt ($\mathbb{R}^3 \times \mathbb{R}^3 \rightarrow \mathbb{R}^3$) zweier Vektoren berechnet sich nach

$$\mathbf{a} \times \mathbf{b} = \begin{vmatrix} e_1 & e_2 & e_3 \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix} \tag{2.52}$$

Das Spatprodukt ($(\mathbb{R}^3 \times \mathbb{R}^3) \cdot \mathbb{R}^3 \to \mathbb{R}$) von drei Vektoren berechnet sich nach

$$(\mathbf{a} \times \mathbf{b}) \cdot \mathbf{c} = det(\mathbf{a}, \mathbf{b}, \mathbf{c}) = \begin{vmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{vmatrix} \tag{2.53}$$

Der Winkel α zwischen zwei Vektoren folgt aus

$$\alpha = \arcsin \frac{|\mathbf{a} \times \mathbf{b}|}{|\mathbf{a}| \cdot |\mathbf{b}|} \tag{2.54}$$

2.4.2 Differentialgeometrische Eigenschaften von Kurven

Grundbetrachtungen

In der *Parameterdarstellung* werden Kurven als Vektorfunktion von einem Parameter ($\mathbb{R} \to \mathbb{R}^3$) formuliert (vgl. Abschnitt 2.1.1).

$$\mathbf{y}(t) = \begin{pmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \end{pmatrix} = \sum_{i=1}^{3} y_i(t)\mathbf{e}_i \tag{2.55}$$

Besitzt jeder Punkt $\mathbf{x}$ auf der Kurve einen eindeutigen Parameterwert t so kann die Umkehrabbildung ($\mathbb{R}^3 \to \mathbb{R}$)

$$y^{-1}(\mathbf{x}) = t \tag{2.56}$$

analytisch oder iterativ ermittelt werden. Die Ableitung der Kurvenfunktion nach dem Parameter t ist definiert durch die Vorschrift

$$\dot{\mathbf{y}}(t) = \frac{d}{dt}\mathbf{y}(t) = \begin{pmatrix} \frac{d}{dt}y_1(t) \\ \frac{d}{dt}y_2(t) \\ \frac{d}{dt}y_3(t) \end{pmatrix} = \begin{pmatrix} \dot{y}_1(t) \\ \dot{y}_2(t) \\ \dot{y}_3(t) \end{pmatrix} \tag{2.57}$$

und kann als Parametergeschwindigkeit interpretiert werden. Verschwindet die Ableitung der Kurvenfunktion in einem Parameterwert $\dot{\mathbf{y}}(t_0) = \mathbf{0}$, so handelt es sich um einen singulären Punkt in dem die Kurve keine eindeutige Tangente hat (z.B. Mehrfachpunkt).

Bogenlänge

Die Länge einer Kurve wird auch als Bogenlänge bezeichnet. Ausgehend von einem festen Punkt $y(\mathbf{a})$ bis zu einem zweiten Punkt $y(\mathbf{b})$ bestimmt sich die Bogenlänge durch

$$s(t) = \int_a^b |\dot{\mathbf{y}}(t)| dt \tag{2.58}$$

Daraus resultierend lässt sich die Länge eines Bogenelementes definieren durch

$$ds = |\dot{\mathbf{y}}(t)| dt \tag{2.59}$$

Die Bogenlänge s einer Kurve wird auch als natürlicher Parameter bezeichnet. Dieser ordnet der aktuellen Länge des Kurvenstücks die aktuelle Lage im euklidischen Raum zu. Existiert die Bogenlänge $s(t)$ so spricht man von ratifizierbaren Kurven. Betrachtet man die Kurve über den Funktionsparameter t, so spricht man von freier Parametrisierung ($y(t)$), wird die Bogenlänge s als Funktionsparameter verwendet, so spricht man von natürlicher Parametrisierung ($\hat{y}(s)$).

Das begleitende Dreibein

Für eine Kurve in natürlicher Parametrisierung berechnet sich durch die folgende Vorschrift der Tangentenvektor bzw. Tangentialvektor

$$\mathbf{t}(s) = \hat{\mathbf{y}}'(s) = \frac{\dot{\mathbf{y}}(t)}{|\dot{\mathbf{y}}(t)|} \tag{2.60}$$

Betrachtet man die zweite Ableitung der Kurvenfunktion, so lässt sich der sogenannte Hauptnormalenvektor berechnen nach

$$\mathbf{n}(s) = \frac{\hat{\mathbf{y}}''(s)}{|\hat{\mathbf{y}}''(s)|} \tag{2.61}$$

Da die beiden Vektoren orthogonal zueinander sind, lässt sich leicht ein dritter hierzu orthogonaler Vektor angeben, der sogenannte Binormalenvektor

$$\mathbf{b}(s) = \mathbf{t}(s) \times \mathbf{n}(s) \tag{2.62}$$

Die drei oben beschriebenen Vektoren $(\mathbf{t}(s), \mathbf{n}(s), \mathbf{b}(s))$ bezeichnet man als begleitendes Dreibein.

Krümmung

Die Krümmung einer Kurve kann vereinfacht als Abweichung einer Kurve von einer Geraden betrachtet werden. Das Krümmungsmaß κ gibt für einen Punkt an, wie stark diese Abweichung ist. Die Krümmung einer Kurve in natürlicher Parametrisierung bestimmt sich durch

$$\kappa(s) = |\hat{\mathbf{y}}''(s)| \tag{2.63}$$

In freier Parametrisierung kann die Krümmung ermittelt werden nach

$$\kappa(t) = \frac{\dot{\mathbf{y}}(t) \times \ddot{\mathbf{y}}(t)}{|\dot{\mathbf{y}}(t)|^3} \tag{2.64}$$

2.4.3 Differentialgeometrische Eigenschaften von Flächen

Grundbetrachtungen

In der *Parameterdarstellung* werden Flächen als Vektorfunktion von zwei Parametern ($\mathbb{R}^2 \to \mathbb{R}^3$) formuliert (vgl. Abschnitt 2.2.1).

$$\mathbf{x}(u_1, u_2) = \begin{pmatrix} x_1(u_1, u_2) \\ x_2(u_1, u_2) \\ x_3(u_1, u_2) \end{pmatrix} = \sum_{i=1}^{3} x_i(u_1, u_2)\mathbf{e}_i \tag{2.65}$$

Besitzt jeder Punkt $\mathbf{x} \in \mathbb{R}^3$ auf der Fläche ein eindeutiges Parameterpaar $\mathbf{u} \in \mathbb{R}^2$ so kann die Umkehrabbildung ($\mathbb{R}^3 \to \mathbb{R}^2$)

$$\mathbf{x}^{-1}(\mathbf{x}) = \mathbf{u} \tag{2.66}$$

analytisch oder iterativ ermittelt werden. Die partiellen Ableitungen der Flächenfunktion nach den beiden Parametern u_1, u_2 sind definiert durch die Vorschrift

$$\frac{\partial}{\partial u_i}\mathbf{x}(u_1, u_2) = \begin{pmatrix} \frac{\partial}{\partial u_i} x_1(u_1, u_2) \\ \frac{\partial}{\partial u_i} x_2(u_1, u_2) \\ \frac{\partial}{\partial u_i} x_3(u_1, u_2) \end{pmatrix};\ i = 1, 2 \tag{2.67}$$

Verschwindet eine der beiden partiellen Ableitungen der Flächenfunktion in einem Parameterpaar ($\frac{\partial}{\partial u_i} x(u_1, u_2) = 0$), so handelt es sich um einen singulären Punkt in dem die Fläche keine eindeutige Tangente hat (z.B. Nord- und Südpol einer Kugeloberfläche). Für eine gegebene Flächenfunktion $x(u_1, u_2)$ wird die Matrix

$$\mathbf{J}_F(u_1, u_2) = \begin{bmatrix} \frac{\partial x_1(u_1,u_2)}{\partial u_1} & \frac{\partial x_1(u_1,u_2)}{\partial u_2} \\ \frac{\partial x_2(u_1,u_2)}{\partial u_1} & \frac{\partial x_2(u_1,u_2)}{\partial u_2} \\ \frac{\partial x_3(u_1,u_2)}{\partial u_1} & \frac{\partial x_3(u_1,u_2)}{\partial u_2} \end{bmatrix} \tag{2.68}$$

auch als Jakobimatrix bezeichnet.

Das begleitende Dreibein

Für eine Fläche in der Parameterdarstellung $x(u_1, u_2)$ berechnen sich die beiden Tangentialvektoren in einem beliebigen Punkt $u_{1,0}, u_{2,0}$ nach

$$\mathbf{t}_1(u_1, u_2) = \frac{\partial}{\partial u_1} x(u_1, u_{2,0}) \mid_{u_{1,0}} \tag{2.69a}$$

$$\mathbf{t}_2(u_1, u_2) = \frac{\partial}{\partial u_2} x(u_{1,0}, u_2) \mid_{u_{2,0}} \tag{2.69b}$$

Die Flächennormale ergibt sich aus dem Kreuzprodukt der beiden Tangentenvektoren nach

$$\mathbf{f}(u_1, u_2) = \frac{\mathbf{t}_1 \times \mathbf{t}_2}{|\mathbf{t}_1 \times \mathbf{t}_2|} = \frac{\mathbf{x}_{,1} \times \mathbf{x}_{,2}}{|\mathbf{x}_{,1} \times \mathbf{x}_{,2}|} \tag{2.70}$$

wobei $x_{,i}$ die partielle Ableitungen der Flächenfunktion nach u_i ist. Die beiden oben beschriebenen Vektoren $(t_1(u_1, u_2), t_2(u_1, u_2))$ sowie die Flächennormale $f(u_1, u_2)$ bezeichnet man als begleitendes Dreibein der Fläche.

1. metrische Fundamentalgröße

Die 1. metrische Fundamentalgröße G ist definiert als

$$\mathbf{G} = \mathbf{J}_F^{\mathrm{T}}(u_1, u_2)\, \mathbf{J}_F(u_1, u_2) = \begin{bmatrix} G_{11}(u_1, u_2) & G_{12}(u_1, u_2) \\ G_{21}(u_1, u_2) & G_{22}(u_1, u_2) \end{bmatrix} \tag{2.71}$$

Sie bestimmt alle Krümmungsmaße einer Fläche und wird verwendet, um die Bogenlänge von Kurven auf Flächen, die Schnittwinkel zwischen Flächenkurven und

den Flächeninhalt von Flächenstücken zu berechnen. Definiert man die Flächenparameter $u_i = u_i(t), i = 1, 2$ als Funktionen in Abhängigkeit des Parameters t, so liegt die Kurve

$$\mathbf{y}(t) = \mathbf{x}(u_1(t), u_2(t)) \tag{2.72}$$

in der Fläche F. Ausgehend von der Definition eines Bogenelementes einer Kurve (2.59) bestimmt sich ein Bogenelement einer Flächenkurve durch

$$ds = \sqrt{\dot{\mathbf{y}}^T \mathbf{G} \dot{\mathbf{y}}} \tag{2.73}$$

Die Bogenlänge einer Flächenkurve ist definiert durch die Vorschrift

$$s = \int_0^t \sqrt{\dot{\mathbf{y}}^T \mathbf{G} \dot{\mathbf{y}}} dt \tag{2.74}$$

Betrachtet man den Schnittwinkel zwischen zwei Flächenkurven $\mathbf{y}_1(t)$ und $\mathbf{y}_2(t)$, so ergibt sich die Gleichung

$$\cos \Phi = \frac{\dot{y}_2^T G \dot{y}_1}{\sqrt{\dot{y}_1^T G \dot{y}_1} \sqrt{\dot{y}_2^T G \dot{y}_2}} \tag{2.75}$$

Der Flächeninhalt des Flächenstücks $(dy_1 x dy_2)$ berechnet sich zu

$$A_F = \int_0^{t_1} \int_0^{t_2} \sqrt{\det \mathbf{G}} dy_1 dy_2 \tag{2.76}$$

2. metrische Fundamentalgröße

Die 2. metrische Fundamentalgröße $\mathbf{L}$ ist definiert als

$$\mathbf{L} = \mathbf{x}_{,ij}(u_1, u_2)\, \mathbf{f}(u_1, u_2) = \begin{bmatrix} L_{11}(u_1, u_2) & L_{12}(u_1, u_2) \\ L_{21}(u_1, u_2) & L_{22}(u_1, u_2) \end{bmatrix} \tag{2.77}$$

mit $\mathbf{x}_{,ij}(u_1, u_2)$ als den 2. partiellen Ableitungen der Parameterdarstellung der Fläche und $\mathbf{f}(u_1, u_2)$ als dem Flächennormalenvektor. Sie liefert ein direktes Maß für die Krümmung von Flächenkurven. Im Rahmen dieser Arbeit spielt die 2. metrische Fundamentalgröße jedoch keine Rolle.

2.5 Topologische Aspekte der geometrischen Modellierung

Bei der Betrachtung von Volumenmodellen indirekter Darstellungsweise (vgl. Kapitel 2.3.2) lassen sich mit Hilfe der Graphentheorie und einfachen Matrizenoperationen Aussagen über die Topologie und die Orientierung des geometrischen Modells treffen. Dieser Abschnitt behandelt die für diese Arbeit wichtigsten topologischen Aspekte der graphenbasierten geometrischen Modellierung.

2.5.1 Graphenbasierte Datenbasis

Ein einfacher Graph $G = (V, E)$ besteht aus Knoten V und Kanten E, die jeweils zwei Knoten verbinden. Wird bei Kanten zwischen Startknoten v_a und Endknoten v_e unterschieden, so spricht man von einem gerichteten Graphen. Ist die Reihenfolge unwesentlich, so ist der Graph ungerichtet. Abbildung 2.24 zeigt ein Beispiel für einen gerichteten Graphen mit 6 Knoten und 9 Kanten.

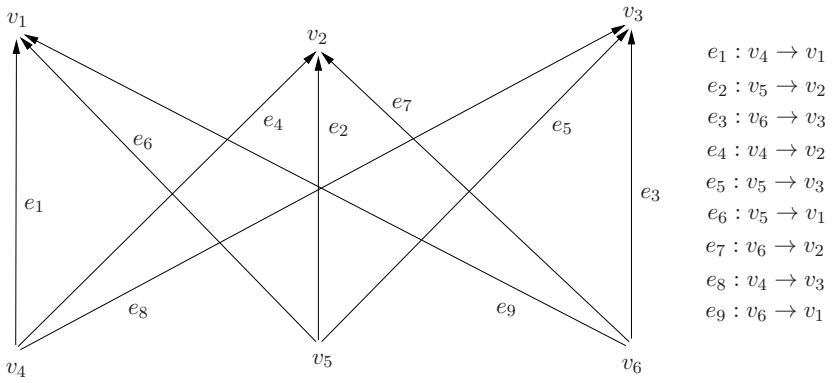


Abbildung 2.24: Gerichteter Graph

Alternativ lässt sich der Graph auch als Matrix M darstellen, bei der die Zeilen die einzelnen Kanten E und die Spalten die Knoten V repräsentieren. Ist ein Knoten v_i der Startknoten einer Kante e_j, so steht in der Matrix an der entsprechenden Stelle (i, j) eine -1, ist der Knoten v_i der Endknoten einer Kante e_j, so steht an entsprechender Stelle eine $+1$. Besteht keine Verbindung zwischen Knoten v_i und Kante e_j, so steht in der Matrix an der entsprechenden Stelle eine 0 (vgl. Abbildung 2.25)

Der Nullraum einer Matrix $\mathbf{M}$ bezeichnet die Menge aller von Null verschiedenen Vektoren $\mathbf{x}$, für die gilt:

$$\mathbf{M} \cdot \mathbf{x} = 0 \tag{2.78}$$

Die Lösung des linearen Gleichungssystems liefert den Nullraum der Matrix.

$$\begin{bmatrix}
-1 & 0 & 0 & 1 & 0 & 0 \\
0 & -1 & 0 & 0 & 1 & 0 \\
0 & 0 & -1 & 0 & 0 & 1 \\
0 & -1 & 0 & 1 & 0 & 0 \\
0 & 0 & -1 & 0 & 1 & 0 \\
-1 & 0 & 0 & 0 & 1 & 0 \\
0 & -1 & 0 & 0 & 0 & 1 \\
0 & 0 & -1 & 1 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}$$

Abbildung 2.25: Matrix des gerichteten Graphen

2.5.2 Identifizierung von geschlossenen Volumina

Die Identifizierung von geschlossenen Flächen und Volumina und deren lokaler Orientierung spielt bei der Weiterverarbeitung mit numerischen Algorithmen eine entscheidende Rolle. Mit Hilfe der Graphentheorie lassen sich wichtige Aussagen zu den betrachteten Modellen treffen.

Knoten-Kanten Matrix

In der Randdarstellung werden alle geometrischen Flächen topologisch über ihre Randkanten (äußerer Rand, innerer Rand) definiert. Dabei besitzt jede Randkante eine eindeutige Orientierung mit einem Start- und einem Endknoten. Beschreibt man das geometrische Modell über einen gerichteten Graphen, bei dem die Kanten des Graphen den Randkanten entsprechen und die Knoten des Graphen den geometrischen Knoten, so lässt sich die Topologie des Modells über eine Knoten-Kanten Matrix darstellen (vgl. Abbildung 2.26).

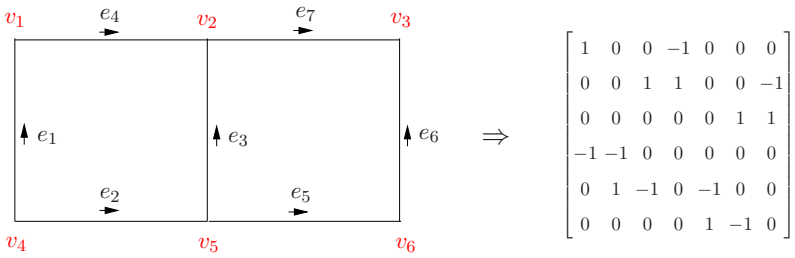


Abbildung 2.26: Knoten-Kanten Matrix

Berechnet man den Nullraum dieser Matrix wie oben beschrieben, so entspricht jeder Lösungsvektor einer topologisch geschlossenen Fläche. Die Koeffizienten im Lösungsvektor markieren dabei die entsprechenden Kanten nach folgendem Muster:

- *0:* Kante gehört nicht zu Flächenbeschreibung

- *1:* Kante gehört zur Flächenbeschreibung, Kantenorientierung entspricht Flächenumlaufsinn.
- *-1:* Kante gehört zur Flächenbeschreibung, Kantenorientierung entgegen Flächenumlaufsinn.

Der Nullraum der Knoten-Kanten Matrix aus Abbildung 2.26 enthält die zwei folgenden Lösungsvektoren:

$$\begin{bmatrix} -1 & 1 & 1 & -1 & 0 & 0 & 0 \end{bmatrix} \tag{2.79a}$$

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 1 & 1 & -1 \end{bmatrix} \tag{2.79b}$$

Die beiden dazugehörigen topologischen Flächen bestehen demzufolge aus den Kanten (e_1, e_2, e_3, e_4) sowie (e_3, e_5, e_6, e_7) mit den entgegen dem Umlaufsinn orientierten Kanten e_1 und e_4 für die erste Fläche und e_3 und e_7 für die zweite Fläche.

Kanten-Flächen Matrix

Analog zum oben gezeigten Vorgehen lassen sich bei indirekten Volumenmodellen Aussagen über die einzelnen Volumina treffen. Hierfür werden alle topologischen Flächen und ihre Randkanten betrachtet. Die Orientierung der Flächen ist hierbei durch die Flächennormalenvektoren definiert, die immer im Sinne der *Rechten-Faust Regel* zum Flächenumlauf orientiert sind: Weist der Daumen der rechten Hand in Richtung der Flächennormalenvektoren, so zeigen die gekrümmten Finger in die Umlaufrichtung des Randes dieser Fläche. Zunächst wird das geometrische Modell über eine Kanten-Flächen Matrix beschrieben, bei der die einzelnen Zeilen den topologischen Kanten entsprechen und die Spalten der Matrix den topologischen Flächen. Jede Kante, die zur Randbeschreibung einer Fläche gehört, wird in der Matrix an entsprechender Stelle mit einer 1 (Kantenorientierung entspricht Flächenumlauf) oder einer -1 (Kantenorientierung entgegen Flächenumlauf) markiert. Gehört die Kante nicht zum Flächenrand, so steht in der Matrix an entsprechender Stelle eine 0 (siehe Abbildung 2.27).

Berechnet man den Nullraum dieser Matrix, so entspricht jeder Lösungsvektor einem topologisch geschlossenen Volumen. Die Koeffizienten im Lösungsvektor markieren jetzt die entsprechenden Flächen nach folgendem Muster:

- *0:* Fläche gehört nicht zur Volumenbeschreibung
- *1:* Fläche gehört zur Volumenbeschreibung, Flächennormalenvektor zeigt nach außen.
- *-1:* Fläche gehört zur Volumenbeschreibung, Flächennormalenvektor zeigt nach innen.

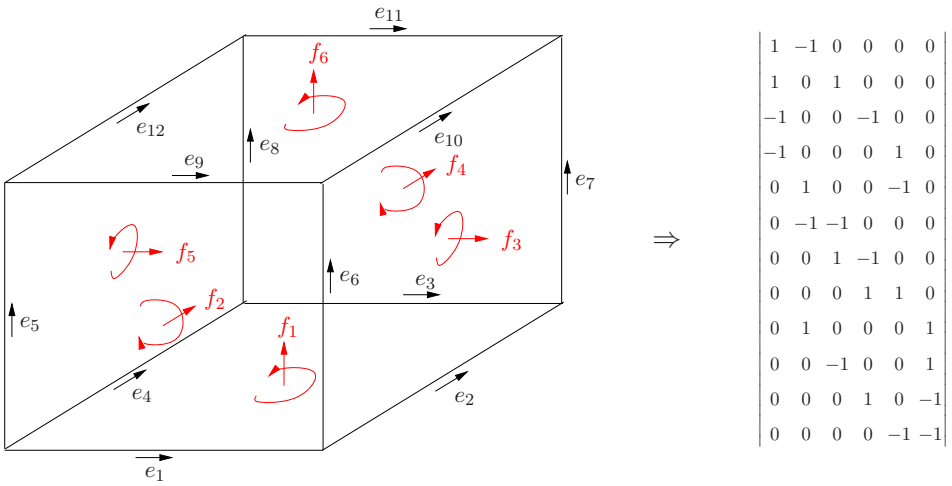


Abbildung 2.27: Kanten-Flächen Matrix

Der Nullraum der Kanten-Flächen Matrix aus Abbildung 2.27 enthält folgenden Lösungsvektor:

$$\begin{bmatrix} -1 & -1 & 1 & 1 & -1 & 1 \end{bmatrix} \tag{2.80}$$

Das innerhalb des Lösungsvektors beschriebene Volumen besteht aus den sechs Flächen ($f_1, f_2, f_3, f_4, f_5, f_6$), wobei die Flächennormalenvektoren der Flächen f_1, f_2 und f_5 ins Volumeninnere zeigen und die Flächennormalenvektoren der Flächen f_3, f_4 und f_6 nach außen.

Kapitel 3

Methode der Finiten Elemente

Nachdem in dem vorangegangenen Kapitel die Grundlagen der geometrischen Modellierung vermittelt worden sind, folgt nun eine Beschreibung der Finite-Elemente-Methode, welche eines der am häufigsten eingesetzten Verfahren zur Berechnung von geometriebasierten physikalischen Phänomenen in der Fluid- und Strukturmechanik ist. Zunächst wird der klassische Finite-Elemente-Ansatz für ein dreidimensionales lineares Elastizitätsproblem eingeführt. Anschließend wird auf die p-Version der Finite-Elemente-Methode eingegangen. Das Hauptaugenmerk dieser Ausführung liegt in der Diskretisierung des geometrischen Modells für die partitionierte Integration. Detaillierte Beschreibungen zu dem gesamten Themengebiet sind in der Literatur zu finden [15, 19, 40, 42, 81, 100, 117].

3.1 Grundlagen der linearen Elastizitätstheorie

3.1.1 Gleichgewicht

Die Betrachtung des Gleichgewichts für ein dreidimensionales Kontinuum mit dem Gebiet Ω und dem Rand $\partial\Omega = \Gamma$ erfolgt an einem infinitesimal heraus geschnittenen Elementarteilchen (vgl. Abbildung 3.1). Die Gleichgewichtsbedingungen für das dreidimensionale Elementarteilchen lassen sich in Form eines Differentialgleichungssystems beschreiben als

$$\begin{aligned} \frac{\partial \sigma_x^{(u)}}{\partial x} + \frac{\partial \tau_{xy}^{(u)}}{\partial y} + \frac{\partial \tau_{xz}^{(u)}}{\partial z} + f_x = 0 \\ \frac{\partial \tau_{yx}^{(u)}}{\partial x} + \frac{\partial \sigma_y^{(u)}}{\partial y} + \frac{\partial \tau_{yz}^{(u)}}{\partial z} + f_y = 0 \\ \frac{\partial \tau_{zx}^{(u)}}{\partial x} + \frac{\partial \tau_{zy}^{(u)}}{\partial y} + \frac{\partial \sigma_z^{(u)}}{\partial z} + f_z = 0 \end{aligned} \tag{3.1}$$

wobei f_x, f_y, f_z die Volumenkräfte darstellen.

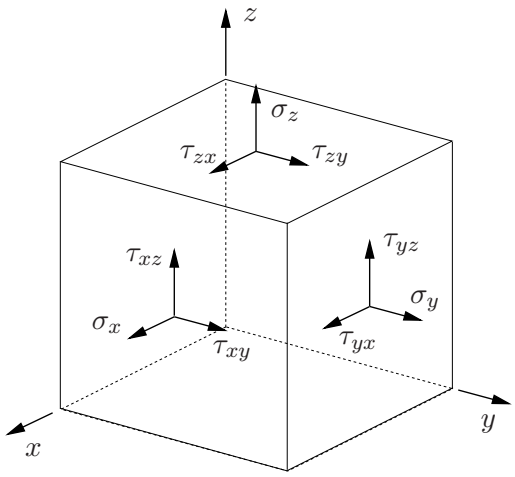


Abbildung 3.1: Spannungszustand am Elementarteilchen

3.1.2 Kinematik

Die Verschiebungen **u** des Körpers aus dem unverformten Zustand werden durch den Verschiebungsvektor

$$\mathbf{u} = \begin{bmatrix} u_x(x,y,z) & u_y(x,y,z) & u_z(x,y,z) \end{bmatrix}^T \tag{3.2}$$

erfasst, der als Funktion in Abhängigkeit von euklidischen Koordinaten definiert ist. Das Verhältnis von Verzerrungen zu Verschiebungen wird durch einen linearen Zusammenhang zu den Verschiebungsableitungen dargestellt und resultiert in dem Verzerrungsvektor

$$\epsilon^{(u)} = \begin{bmatrix} \epsilon_x^{(u)} & \epsilon_y^{(u)} & \epsilon_z^{(u)} & \gamma_{xy}^{(u)} & \gamma_{yz}^{(u)} & \gamma_{xz}^{(u)} \end{bmatrix}^T = \mathbf{Du} \tag{3.3}$$

wobei **D** der Differentialoperatormatrix

$$\mathbf{D} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \end{bmatrix} \tag{3.4}$$

entspricht.

3.1.3 Stoffgesetz

Für ein isotropes, linear elastisches Material ergeben sich die Spannungen

$$\sigma^{(u)} = \begin{bmatrix} \sigma_x^{(u)} & \sigma_y^{(u)} & \sigma_z^{(u)} & \tau_{xy}^{(u)} & \tau_{yz}^{(u)} & \tau_{xz}^{(u)} \end{bmatrix}^T \tag{3.5}$$

durch einen linearen Zusammenhang zwischen Spannungen und Verzerrungen mit der Beziehung

$$\sigma^{(u)} = \mathbf{C}(\epsilon^{(u)} - \epsilon^{(0)}) + \sigma^{(\mathbf{0})} \tag{3.6}$$

Dabei werden in $\epsilon^{(\mathbf{0})}$ die vorgegebenen Anfangsverzerrungen und in $\sigma^{(\mathbf{0})}$ die aufgebrachten Anfangsspannungen definiert. Die Elastizitätsmatrix $\mathbf{C}$ ergibt sich mithilfe des Hookeschen Gesetzes zu

$$\mathbf{C} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} (1-\nu) & \nu & \nu & 0 & 0 & 0 \\ \nu & (1-\nu) & \nu & 0 & 0 & 0 \\ \nu & \nu & (1-\nu) & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \tag{3.7}$$

wobei E den Elastizitätsmodul und ν die Poissonzahl bezeichnet.

3.1.4 Randbedingungen

Die Randbedingungen werden als Neumann-Randbedingungen Γ_T (Kraftrandbedingungen) und als Dirichlet-Randbedingungen Γ_u (Verschiebungsrandbedingungen) auf dem Rand $\partial\Omega$ des Berechnungsgebietes Ω definiert. Für eine vektorielle Erfassung der Randbedingungen werden diese mithilfe des Normalenvektors $\mathbf{n}$ und den Tangentenvektoren $\mathbf{t_1}$ und $\mathbf{t_2}$ in Komponenten normal und tangential zum Rand angegeben. Aufgrund des Kräftegleichgewichts am Rand ergibt sich

$$\begin{bmatrix} T_n^{(u)} \\ T_{t1}^{(u)} \\ T_{t2}^{(u)} \end{bmatrix} = \begin{bmatrix} n_x & n_y & n_z \\ t_{1x} & t_{1y} & t_{1z} \\ t_{2x} & t_{2y} & t_{2z} \end{bmatrix} \begin{bmatrix} \sigma_x^{(u)} & \tau_{xy}^{(u)} & \tau_{xz}^{(u)} \\ \tau_{xy}^{(u)} & \sigma_y^{(u)} & \tau_{yz}^{(u)} \\ \tau_{xz}^{(u)} & \tau_{yz}^{(u)} & \sigma_z^{(u)} \end{bmatrix} \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} \tag{3.8}$$

$$\begin{bmatrix} u_n^{(u)} \\ u_{t1}^{(u)} \\ u_{t2}^{(u)} \end{bmatrix} = \begin{bmatrix} n_x & n_y & n_z \\ t_{1x} & t_{1y} & t_{1z} \\ t_{2x} & t_{2y} & t_{2z} \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} \tag{3.9}$$

An jedem Randpunkt muss für jede Richtungskomponente entweder eine Verschiebungs- oder eine Kraftrandbedingung gegeben sein.

$$\begin{aligned} \Gamma &= \Gamma_T^n \cup \Gamma_u^n, & \Gamma_T^n \cap \Gamma_u^n &= 0 \\ \Gamma &= \Gamma_T^{t1} \cup \Gamma_u^{t1}, & \Gamma_T^{t1} \cap \Gamma_u^{t1} &= 0 \\ \Gamma &= \Gamma_T^{t2} \cup \Gamma_u^{t2}, & \Gamma_T^{t2} \cap \Gamma_u^{t2} &= 0 \end{aligned} \tag{3.10}$$

3.2 Prinzip der virtuellen Arbeit

Das Prinzip der virtuellen Arbeit stellt als klassisches Variationsprinzip die Grundlage der Methode der finiten Elemente dar. Durch Multiplikation des Differentialgleichungssystems (3.1) mit einer Testfunktion

$$\mathbf{v} = \begin{bmatrix} v_x(x,y,z) & v_y(x,y,z) & v_z(x,y,z) \end{bmatrix}^T \tag{3.11}$$

und anschließender Integration über das Berechnungsgebiet lässt sich die schwache Form des Gleichgewichts

$$\iiint_\Omega \epsilon^{(\mathbf{v})T} \mathbf{C} \epsilon^{(\mathbf{u})} d\Omega = \iiint_\Omega \mathbf{v}^T \mathbf{f}_\Omega d\Omega + \iint_{\partial_N \Omega} \mathbf{v}^T \bar{\mathbf{t}} d\partial_N \Omega \tag{3.12}$$

herleiten. Die Testfunktion $\mathbf{v}$ wird dabei oft als virtuelle Verschiebung bezeichnet. Befindet sich ein System im Gleichgewicht, so entspricht für jede beliebige, virtuelle Verschiebungsfunktion die innere virtuelle Arbeit der äußeren virtuellen Arbeit. Aus (3.3) und (3.6) ergibt sich

$$\epsilon^{(v)T} \sigma^{(u)} = (\mathbf{Dv})^T \mathbf{C}\, \mathbf{Du} - (\mathbf{Dv})^T \mathbf{C}\, \epsilon^{(\mathbf{0})} - (\mathbf{Dv})^T \sigma^{(0)} \tag{3.13}$$

Aus Gleichung (3.12) folgt nach partieller Integration

$$\begin{aligned}\iiint_{\Omega}(\mathbf{Dv})^T\mathbf{C}\,\mathbf{Du}\;dx\;dy\;dz = \;& \iiint_{\Omega}(f_x v_x + f_y v_y + f_z v_z)\;dx\;dy\;dz \\ & + \iiint_{\Omega}(\mathbf{Dv})^T(\mathbf{C}\epsilon^{(\mathbf{0})} + \sigma^{(\mathbf{0})})\;dx\;dy\;dz \\ & + \iint_{\Gamma}(T_n^{(u)} v_n + T_{t1}^{(u)} v_{t1} + T_{t2}^{(u)} v_{t2})\;d\Gamma\end{aligned} \tag{3.14}$$

wobei $\mathbf{v}$ die virtuellen Verschiebungen, $\epsilon^{(\mathbf{v})}$ die virtuellen Verzerrungen und $\sigma^{(\mathbf{v})}$ die virtuellen Spannungen bezeichnen. Die linke Seite von Gleichung (3.14) wird als Energieskalarprodukt

$$\mathcal{B}(u,v) := \iiint_{\Omega}(\mathbf{Dv})^T\mathbf{C}\,\mathbf{Du}\;dx\;dy\;dz \tag{3.15}$$

bezeichnet und ist eine symmetrische Bilinearform $\mathcal{B}(u,v) = \mathcal{B}(v,u)$. Die rechte Seite entspricht dem Lastfunktional und stellt eine Linearform $\mathcal{F}$ dar

$$\begin{aligned}\mathcal{F}(v) := \;& \iiint_{\Omega}(f_x v_x + f_y v_y + f_z v_z)\;dx\;dy\;dz \\ & + \iiint_{\Omega}(\mathbf{Dv})^T(\mathbf{C}\epsilon^{(\mathbf{0})} + \sigma^{(\mathbf{0})})\;dx\;dy\;dz \\ & + \iint_{\Gamma}(T_n^{(u)} v_n + T_{t1}^{(u)} v_{t1} + T_{t2}^{(u)} v_{t2})\;d\Gamma\end{aligned} \tag{3.16}$$

3.3 Finite-Elemente-Diskretisierung

In der Finite-Elemente-Methode wird das Berechnungsgebiet Ω zunächst in eine endliche Anzahl von Teilgebieten Ω_e, die sogenannten finiten Elemente, unterteilt (vgl. Abbildung 3.2).

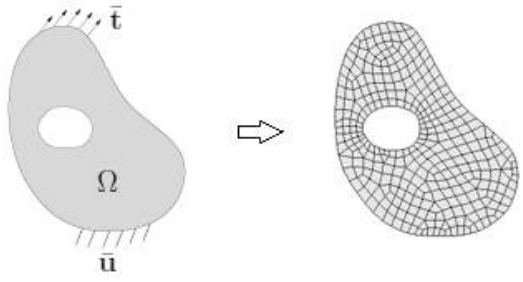
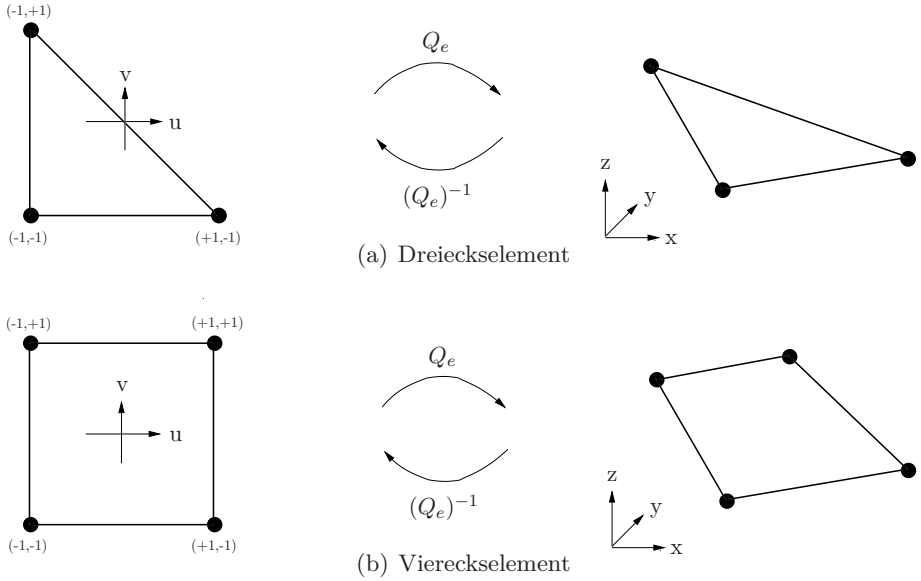


Abbildung 3.2: Diskretisierung durch finite Elementen

Auf diesen Elementen sind Ansatzfunktionen N_i formuliert, welche durch freie Koeffizienten U_i (Freiheitsgrade) miteinander verknüpft werden und so den Lösungsraum für eine Näherungslösung definieren. Die Ansatzfunktionen sind auf einem Standardelement definiert, welches einen einfachen Parameterraum beschreibt. Die Verknüpfung des Parameterraums mit der wahren Geometrie der Elemente ist durch die Abbildungsfunktion Q_e gegeben. Zu dieser muss für eine Berechnung mit der Finite-Elemente-Methode eine eindeutige Umkehrabbildung $(Q_e)^{-1}$ existieren. Abbildung 3.3 zeigt die in dieser Arbeit verwendeten Standardelemente für Dreiecke und Vierecke und deren Abbildung in den euklidischen Raum.



(a) Dreieckselement

(b) Viereckselement

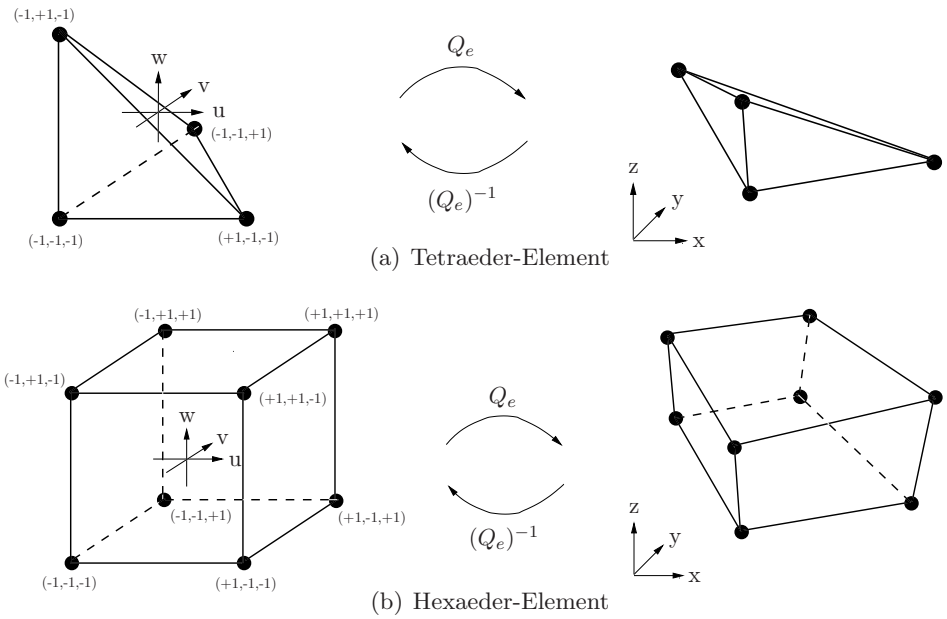
Abbildung 3.3: Abbildung und Umkehrabbildung auf Flächenelementen

Volumenartige Körper wie Tetraeder und Hexaeder werden ebenfalls auf Standardelementen definiert. Diese beschreiben dann einen dreidimensionalen Parameterraum. Abbildung 3.3 zeigt die in dieser Arbeit angesetzten Standardelemente für Tetraeder und Hexaeder und die dazugehörige Abbildung in den euklidischen Raum.

3.3.1 Gleichungssystem

Auf dem diskretisierten Berechnungsgebiet lässt sich die gesuchte Funktion $\mathbf{u}$ der partiellen Differentialgleichung jetzt aus der Summe der Produkte der n Ansatzfunktionen N_i mit den Freiheitsgraden U_i näherungsweise formulieren als

$$\mathbf{u}_{FE} = \sum_{i=1}^{n} \mathbf{U}_i \cdot N_i \tag{3.17}$$



(a) Tetraeder-Element

(b) Hexaeder-Element

Abbildung 3.4: Abbildung und Umkehrabbildung auf Volumenelementen

Setzt man die Näherungsfunktion $\mathbf{u}_{FE}$ in die *schwache Formulierung* der Differentialgleichung (3.13) ein und die Ansatzfunktionen N_j gleichzeitig als Testfunktionen an, so entsteht das Näherungssystem

$$\mathcal{B}(\sum_{i=1}^{n} U_i N_i, N_j) = \mathcal{F}(N_j) \tag{3.18}$$

welches umgeformt werden kann zu

$$\sum_{i=1}^{n} U_i \mathcal{B}(N_i, N_j) = \mathcal{F}(N_j) \tag{3.19}$$

und schließlich in dem zu lösenden Gleichungssystem resultiert:

$$\mathbf{KU} = \mathbf{F} \tag{3.20}$$

wobei $\mathbf{K}$ die Gesamtsteifigkeitsmatrix bezeichnet und $\mathbf{F}$ den Gesamtlastvektor. Die Dimension der Gesamtsteifigkeitsmatrix ergibt sich aus der Anzahl der Ansatzfunktionen multipliziert mit den dem physikalischen Modell zugrunde liegenden Freiheitsgraden (z.B. Verschiebungen in x-Richtung, y-Richtung, Verdrehungen um diverse Achsen).

3.3.2 Integrationsproblem

Durch die Diskretisierung des Berechnungsgebietes verteilt sich das Integrationsproblem der Gesamtsteifigkeitsmatrix $\mathbf{K}$ und des Gesamtlastvektors $\mathbf{F}$ aus Gleichung (3.20) auf Integrale über die einzelnen Elemente. Das Integrationsproblem der Elementsteifigkeitsmatrix $\mathbf{K}^e$ eines dreidimensionalen Elementes lässt sich formulieren als

$$\mathbf{K}^e = \int_{\Omega_E} (.)\, d\Omega_E = \iiint_{\Omega_E} (.)\, dx\, dy\, dz \tag{3.21}$$

Das Integrationsproblem eines dreidimenionalen Elementlastvektors $\mathbf{F}^e$ lässt sich analog darstellen. Innerhalb der Formulierung der Elementsteifigkeitsmatrix $\mathbf{K}^e$ und des Elementlastvektors $\mathbf{F}^e$ werden die Ansatzfunktionen N_i angesetzt, die auf den lokalen Parametern des Standardelementes definiert sind. Um das Integrationsproblem über globale Koordinaten aus Gleichung (3.21) in ein Integrationsproblem auf lokalen Parametern zu überführen, wird die *Jacobi-Matrix* $\mathbf{J}$ der Abbildungsfunktionen Q_e eingeführt:

$$\begin{bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \\ \frac{\partial N_i}{\partial \zeta} \end{bmatrix} = \mathbf{J} \begin{bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} \end{bmatrix} \qquad \text{mit} \quad \mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix} \tag{3.22}$$

$$\begin{bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \\ \frac{\partial N_i}{\partial \zeta} \end{bmatrix} \qquad \text{mit} \quad \mathbf{J}^{-1} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} & \frac{\partial \zeta}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} & \frac{\partial \zeta}{\partial y} \\ \frac{\partial \xi}{\partial z} & \frac{\partial \eta}{\partial z} & \frac{\partial \zeta}{\partial z} \end{bmatrix} \tag{3.23}$$

Die Determinante der *Jacobi-Matrix* beschreibt die Transformation der euklidischen Koordinaten in die Parameterebene

$$dx\, dy\, dz = \det \mathbf{J} \cdot d\xi\, d\eta\, d\zeta \tag{3.24}$$

Mit Hilfe dieser Beziehung kann die Integration auf das Standardelement Ω_{std} transformiert werden. Für die Elementsteifigkeitsmatrix $\mathbf{K}^e$ ergibt sich daraus:

$$\mathbf{K}^e = \int_{-1}^{1} \int_{-1}^{1} \int_{-1}^{1} (.) \det \mathbf{J}\, d\xi\, d\eta\, d\zeta \tag{3.25}$$

Durch numerische Integration lassen sich die einzelnen Elementsteifigkeitsmatritzen $\mathbf{K}^e$ und Elementlastvektoren $\mathbf{F}^e$ einfach und effektiv berechnen und zu der Gesamtsteifigkeitsmatrix $\mathbf{K}$ und dem Gesamtlastvektors $\mathbf{F}$ assemblieren.

$$\mathbf{K} = \bigwedge_{e=1}^{n_{el}} \mathbf{K}^e \tag{3.26}$$

$$\mathbf{F} = \bigwedge_{e=1}^{n_{el}} \mathbf{F}^e \tag{3.27}$$

Diese bilden das Fundament des zu lösenden Gleichungssystems. Im Rahmen unterschiedlicher Finite-Elemente-Formulierungen unterscheiden sich sowohl die Ansatzfunktionen N_i als auch die Abbildungsfunktionen Q_e. Im Folgenden werden die Ansätze der klassischen h-Version und der p-Version der Finite-Elemente-Methode vorgestellt, und miteinander verglichen.

3.3.3 h-Version

Die klassische h-Version der Finite-Elemente-Methode verwendet in der Regel knotenbasierte Ansatzfunktionen, die jeweils in einem Knoten des Standardelementes den Wert 1 und in den anderen Knoten den Wert 0 annehmen. Die Anzahl der Knoten pro Element kann dabei erhöht werden, um polynomiale Ansatzfunktionen höherer Ordnung zu erlauben. Die Verknüpfung benachbarter Elemente während der Assemblierung ist durch gemeinsame Knoten bestimmt, in denen die Freiheitsgrade des Systems definiert sind. Abbildung 3.5 zeigt die eindimensionalen Ansatzfunktionen der klassischen h-Version basierend auf den Lagrange-Polynomen für $p = 1, 2, 3$.

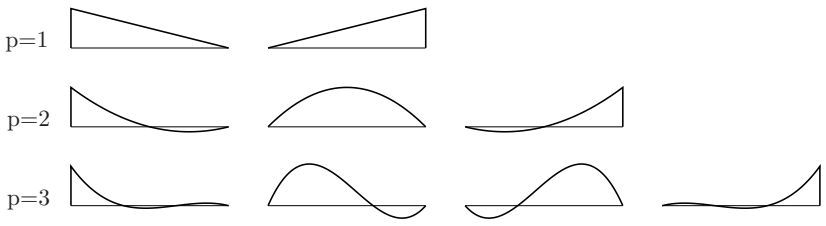


Abbildung 3.5: Lagrange-Polynome

In der h-Version der Finite-Elemente-Methode hängt die Genauigkeit der Lösung direkt von der Anzahl der Knoten ab. Da die Anzahl an Knoten auf allen Elementen vom selben Elementtyp gleich ist, erhöht sich die Genauigkeit einer Berechnung nur durch eine Verfeinerung des Finite-Elemente-Netzes. In der h-Version erfolgt dies oft problemangepasst (adaptiv) an den Stellen, an denen hohe Gradienten z.B. der Spannungen erwartet werden (vgl.Abbildung 3.6).

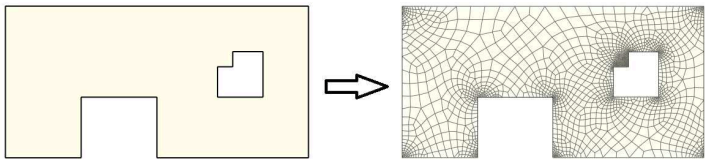


Abbildung 3.6: Adaptive Netzverfeinerung

3.3.4 *p*-Version

Bei der *p*-Version der Finite-Elemente-Methode werden hierarchische Ansatzfunktionen angesetzt und in jedem Element einen polynomialer Lösungsraum hoher Ordnung definiert. Abbildung 3.7 zeigt die eindimensionalen hierarchischen Ansatzfunktionen der Finite-Elemente-Methode hoher Ordnung für $p = 1, 2, 3$.

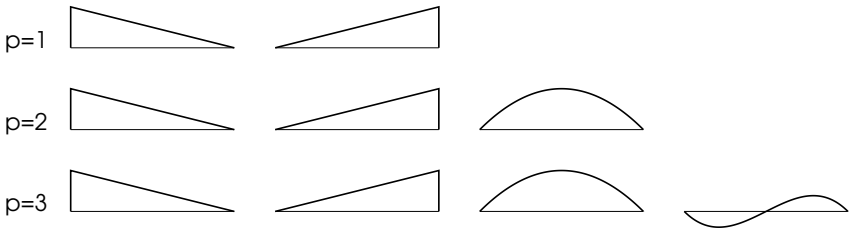
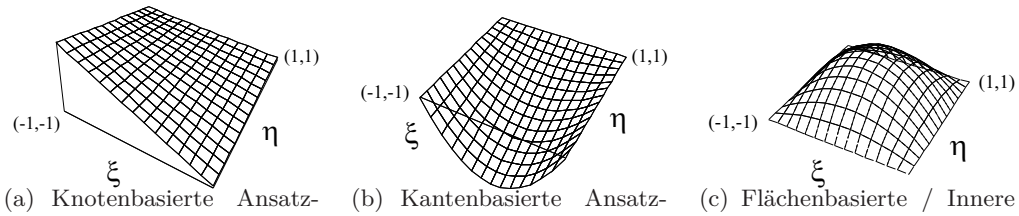


Abbildung 3.7: Hierarchische Ansatzfunktionen

Diese basieren im Gegensatz zum klassischen Ansatz auf den integrierten Legendre-Polynomen [6, 15]. Die hierarchischen Ansatzfunktionen werden unterschieden in

- knotenbasierte Ansatzfunktionen, die jeweils in einem der Knoten den Wert 1, in den anderen Knoten den Wert Null annehmen (vgl. Abbildung 3.8(a)),
- kantenbasierte Ansatzfunktionen, die jeweils auf einer der Kanten eine polynomiale Form annehmen und auf den anderen Kanten verschwinden (vgl. Abbildung 3.8(b)),
- flächenbasierte Ansatzfunktionen, die bei dreidimensionalen Elementen auf einer der Oberflächen leben (vgl. Abbildung 3.8(c)) und in
- innere Ansatzfunktionen, die im Inneren des Elementes definiert sind und auf dem Elementrand vollständig verschwinden.

In der *p*-Version ist auf jedem Element eine hohe Zahl von Ansatzfunktionen definiert. Die Anzahl von Ansatzfunktionen auf einem Element hängt dabei direkt vom Polynomgrad p ab. Dieser kann auf jedem Element und für jede lokale Parameterrichtung individuell gesetzt werden. Eine Erhöhung der Genauigkeit der Lösung erfolgt daher durch eine Erhöhung des Polynomgrads p, während die



(a) Knotenbasierte Ansatzfunktion (b) Kantenbasierte Ansatzfunktion (c) Flächenbasierte / Innere Ansatzfunktion

Abbildung 3.8: Typen von hierarchischen 2D Ansatzfunktionen

Anzahl der Elemente konstant bleibt [40, 101]. Damit die Größe des resultierenden Gleichungssystems nicht unkontrollierbar groß wird, werden in der p-Version Netze mit nur sehr wenigen groben Elementen gefordert. Diese dürfen im Gegensatz zum klassischen Ansatz jedoch stark verzerrt sein und können den Rand des Berechnungsgebietes exakt abbilden (vgl. Abbildung 3.9).

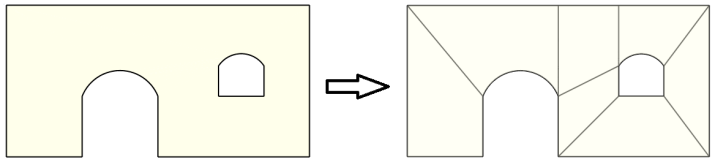


Abbildung 3.9: Finite-Elemente-Netz hoher Ordnung

Da die Elemente in der p-Version eine gekrümmte Berandung annehmen können, bedarf es spezieller Abbildungsfunktionen, die eine Beschreibung beliebiger Elementberandungen erlauben. Eine beliebte Methode zur Abbildung gekrümmter Elementränder ist die Blending-Funktionen-Methode [49, 55], welche die parametrische Beschreibung der geometrischen Randkurve oder Randfläche über die lokalen Parameter des Standardelementes ein- bzw. ausblendet. Abbildung 3.10 zeigt die Abbildung eines Viereckselementes, bei dem die gekrümmte Randkante des Berechnungsgebietes mit Hilfe der Blending-Funktionen-Methode auf das Viereckselement im euklidischen Raum abgebildet wird.

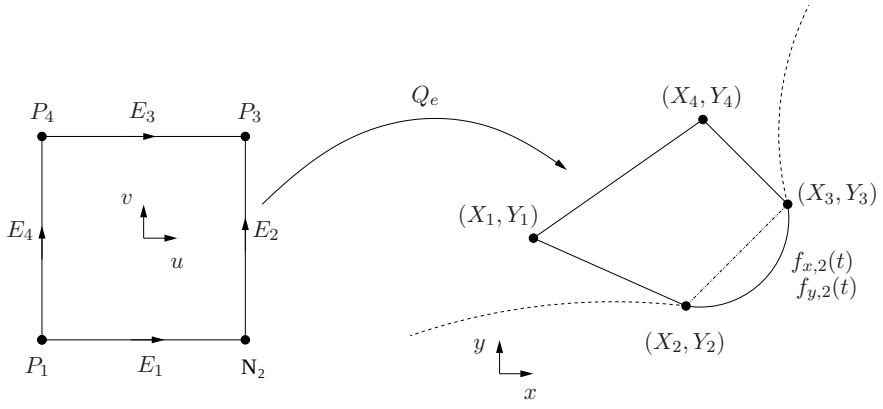


Abbildung 3.10: Abbildung auf Elementen hoher Ordnung

3.4 Fehlerquellen

Die Beschreibung physikalischer Systeme oder Prozesse mit Hilfe numerischer Berechnungsverfahren liefert nur eine Näherung an die reale Welt. Für Probleme der Struktur- oder Strömungsmechanik können Fehler folgendermaßen klassifiziert werden (vgl. Abbildung 3.11):

- *Idealisierungsfehler* entstehen aus der Diskrepanz zwischen der Realität und den idealisierten konstitutiven Gesetzen, den kinematischen Beziehungen und den Randbedingungen.
- *Modellierungsfehler* entstehen aus der Diskrepanz zwischen mathematischer Formulierung und physikalischem Modell (z.B. Formulierung von dimensionsreduzierten Ansätzen für räumliche Probleme).
- *Diskretisierungsfehler* entstehen aus der Diskrepanz zwischen der kontinuierlichen Beschreibung und der diskreten Beschreibung des Modells
- *Lösungsfehler* entstehen bei der Lösung des Gleichungssystems durch den Einsatz von iterativen Näherungsverfahren und durch Rundungsfehler, die im Allgemeinen vernachlässigbar sind.

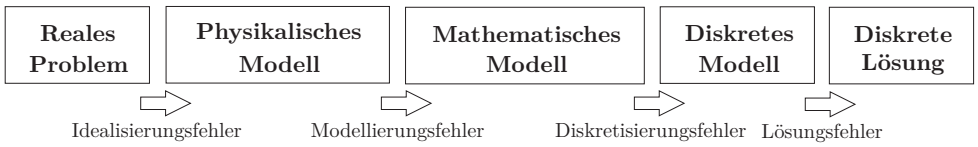


Abbildung 3.11: Fehlerquellen der numerischen Simulation

Idealisierungsfehler entstehen aufgrund vereinfachender Annahmen bei der Definition der physikalischen Gesetze, Modellierungsfehler aufgrund vereinfachender Annahmen bei der Formulierung des mathematischen Modells. Bei komplexen Aufgabenstellungen des konstruktiven Ingenieurwesens werden die berechneten Systemantworten daher immer durch praktische Versuche validiert.

Diskretisierungsfehler entstehen durch die näherungsweise Beschreibung der Geometrie und des Verschiebungsansatzes auf Basis von finiten Elementen. Die Generierung von geeigneten Finite-Elemente Netzen hat daher einen entscheidenden Einfluss auf die Qualität der berechneten Lösung. Im Folgenden beschäftigt sich diese Arbeit mit Verfahren zur Generierung von qualitativ hochwertigen Netzen auf zwei- und dreidimensionalen Berechnungsgebieten, für eine Berechnung mit der h- und der p-Version der Finite-Elemente-Methode.

Kapitel 4

TUM.GeoFrame Rahmen-Software

Die numerische Berechnung von Problemen der Strukturmechanik (siehe Kapitel 3) erfordert die Diskretisierung des geometrischen Modells (vgl. Kapitel 2), oftmals gegeben als computergestütztes Konstruktionsmodell (CAD). Die Implementierung der entwickelten Diskretisierungstechniken wurde innerhalb der eigens entwickelten Rahmen-Software *TUM.GeoFrame* umgesetzt. Die Rahmen-Software umfasst in der aktuellen Version sieben Programmmodule mit etwa 120.000 Zeilen Quellcode und etwa 45.000 Kommentarzeilen. Es beinhaltet Funktionen zur Beschreibung, Diskretisierung, Simulation und Visualisierung von CAD-Modellen. Dieses Kapitel beschreibt den Aufbau von *TUM.GeoFrame*, die Zusammensetzung der einzelnen Programmmodule und die implementierten Schnittstellen.

4.1 Programmaufbau

Die Kernaufgabe des zu entwickelnden Programmfundamentes *TUM.GeoFrame* liegt in der Beschreibung, Bearbeitung und Visualisierung von geometrischen Daten. Die Beschreibung der geometrischen Rohdaten wurde in dem Geometriemodul *TUM.GeometryEngine* als zentraler Projekt-Datenbasis umgesetzt. Die Visualisierung erfolgt im Visualisierungskern (*TUM.VisualEngine*). An den Visualisierungskern angeschlossen ist ein Werkzeugsystem (*TUM.ToolsEngine*), bestehend aus einem kompilierten Hilfemodul (*Help Engine*), sowie einem Lizenzierungssystem für die Verwaltung von Benutzerrechten (*Licence Engine*).

Drei Module mit Algorithmen zur Reparatur (*TUM.HealingEngine*), Diskretisierung (*TUM.MeshingEngine*) und Simulation (*TUM.AnalysisEngine*) der Rohdaten sind über eine gemeinsame Schnittstelle (*TUM.InterfaceEngine*) an die Projekt-Datenbasis angebunden. Diese Schnittstelle bietet darüber hinaus Funktionen zum Anbinden externer Programmbibliotheken. Abbildung 4.1 zeigt den Aufbau der Rahmen-Software sowie die Zusammensetzung der einzelnen Module.

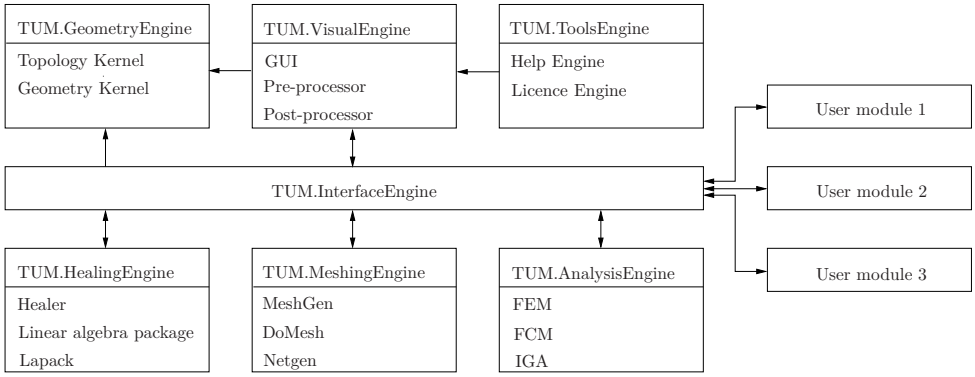


Abbildung 4.1: TUM.GeoFrame - Grafische Benutzeroberfläche

Eine direkte Anbindung der drei algorithmischen Programmmodule an die Projekt-Datenbasis ohne grafische Benutzeroberfläche (GUI) erfolgt über das Verlinken einer Konsolenanwendung (*Console application*). Diese Variante ist durch das Umgehen des Visualisierungssystems schneller und eignet sich besonders für umfangreiche Parameterstudien. Abbildung 4.2 zeigt den Aufbau der Konsolenanwendung.

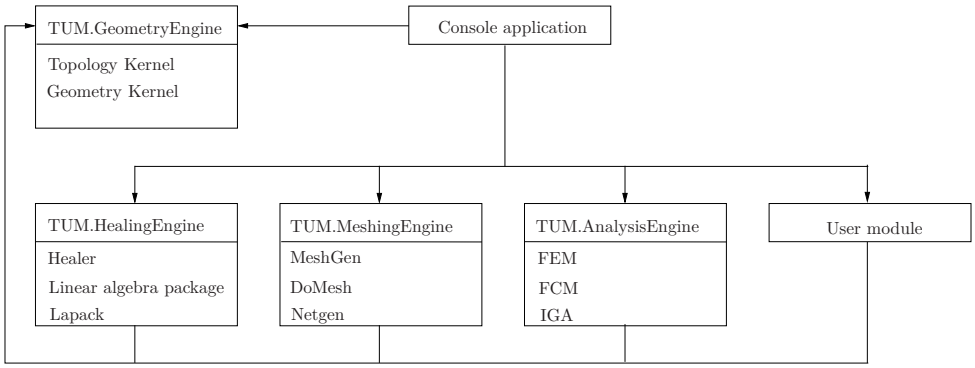


Abbildung 4.2: TUM.GeoFrame - Konsolenanwendung

4.1.1 *TUM.GeometryEngine*

Die Beschreibung von physikalischen Problemstellungen, basierend auf einem geometrischen Modell, den Randbedingungen und den jeweiligen Materialeigenschaften, erfordert eine leistungsfähige und flexible Datenbasis mit optimalen Zugriffszeiten unter vertretbarem Speicheraufwand. Die *TUM.GeometryEngine*, das Herzstück von *TUM.GeoFrame*, ist eine solche Datenbasis. Sie dient als allein stehende Programmbibliothek mit integriertem Geometriekern der Beschreibung von physikalischen Problemstellungen.

Die Beschreibung einer einzelnen Problemformulierung ist in der Basisklasse *Project* implementiert. Diese besitzt zur Geometriebeschreibung vier verschiedene Modelle vom Typ *Topo_Shape*:

- Das *geometrische Hauptmodell* beschreibt die aktuelle Geometrie und ist mit der Visualisierungsschnittstelle gekoppelt.
- Das *geometrische Nebenmodell* ermöglicht die Sicherung der Originalgeometrie und unterstützt die Reparatur-Algorithmen.
- Das *diskretisierte Hauptmodell* beschreibt das aktuelle Finite-Elemente Netz und stellt die Rohdaten für die Visualisierung.
- Das *diskretisierte Nebenmodell* ermöglicht die Definition eines Hintergrundnetzes und unterstützt die Vernetzungsalgorithmen.

Je nach verwendetem Verfahren kann auf eines oder mehrere dieser Modelle gleichzeitig zugegriffen werden und verschiedene geometriebasierte Techniken beliebig kombiniert werden. Darüber hinaus bietet die Basisklasse Methoden zur Kommunikation (*Messages*) und Interaktion (*Interactions*) mit der grafischen Benutzeroberfläche (vgl. Abschnitt 4.1.2). Abbildung 4.3 zeigt den schematischen Aufbau der Projekt-Datenbasis mit den wichtigsten Basisklassen.

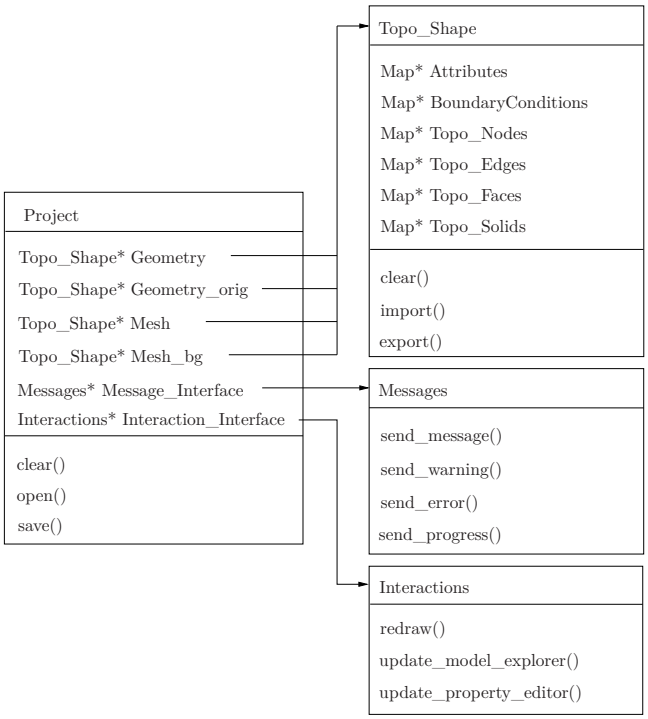


Abbildung 4.3: Projekt-Datenbasis der *TUM.GeometryEngine*

Die Basisklasse *Topo_Shape* dient der Beschreibung der vier parallelen Modelle und besitzt Abbildungslisten (*Map*) für die Verwaltung der unterschiedlichen

Entitätsarten. Jede Entitätsart ist dabei noch mal in topologische und geometrische Entitätsklassen untergliedert. Die topologischen Entitätsklassen besitzen Informationen über die Nachbarschaftsbeziehungen zu den höher und niedriger gestellten topologischen Entitätsklassen und verweisen auf ein Objekt der jeweiligen geometrischen Entitätsklasse. Die geometrischen Entitätsklassen besitzen Informationen über die Form der jeweiligen Objekte und beinhalten eine Vielzahl an geometrischen Funktionalitäten (z.B. Verschneidung, Abbildung/Umkehrabbildung, Metrik, ...). Der Geometriekern basiert auf den Klassen des Geometriemodellierers *Open CASCADE* [71]. Für einen Vollzugriff auf die komplette Funktionalität des Geometriemodellierers kann auf die Objekte der entsprechenden *Open CASCADE*-Klassen zugegriffen werden. Abbildung 4.4 zeigt den Aufbau des Entitätsmodells (*Topo_Shape*) und die Hierarchie der topologischen und geometrischen Entitätsklassen.

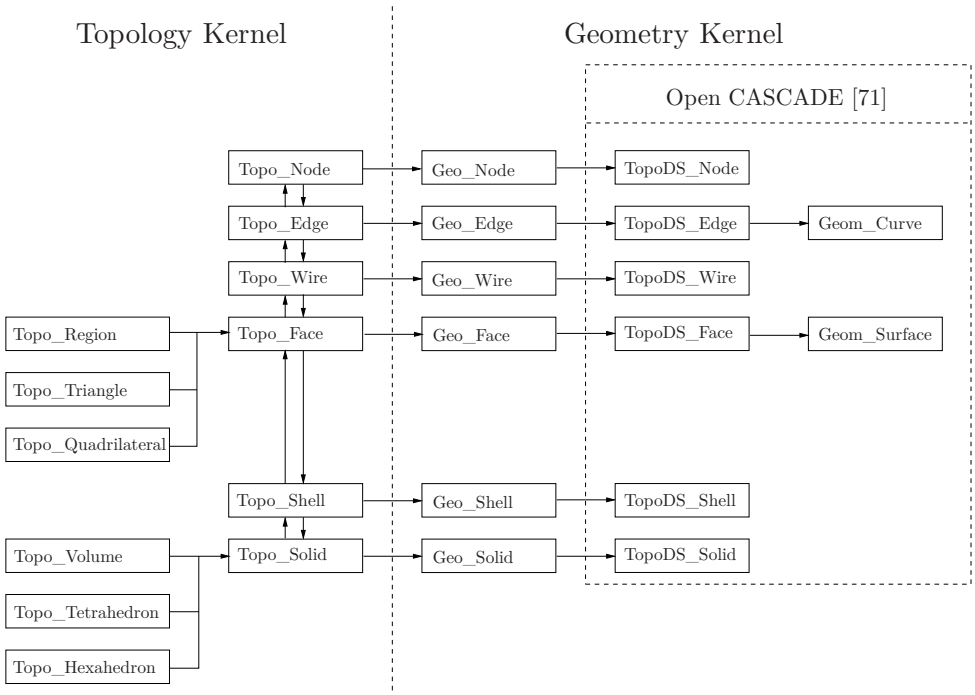


Abbildung 4.4: Topologische und geometrische Hierarchie der Entitätsklassen

4.1.2 *TUM.VisualEngine*

Eine der wichtigsten Grundlagen in der heutigen computergestützten Berechnung von physikalischen Problemstellungen ist die Visualisierung des geometrischen Modells und der berechneten physikalischen Lösungen. Darüber hinaus ist eine möglichst benutzerfreundliche und intuitive Interaktion mit dem Modell samt Randbedingungen eines der wichtigsten Merkmale von erfolgreichen Softwarekonzepten. Die *TUM.VisualEngine* beinhaltet eine benutzerfreundliche grafische Benutzeroberfläche (GUI) und dient der Visualisierung der geometrischen Modelle über zwei verschiedene Visualisierungsschnittstellen, einer Prä-Prozessor

Schnittstelle zur Interaktion mit den geometrischen Entitäten und einer Post-Prozessor Schnittstelle zur grafischen Ausgabe der Simulationsergebnisse (vgl. Abbildung 4.5).

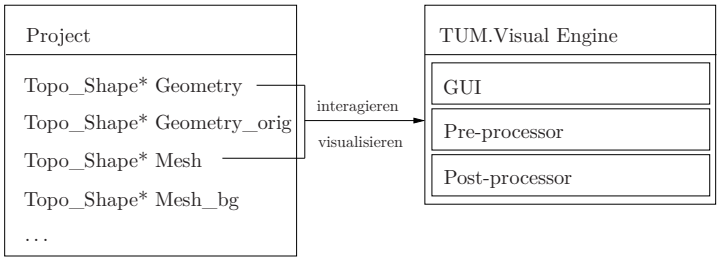
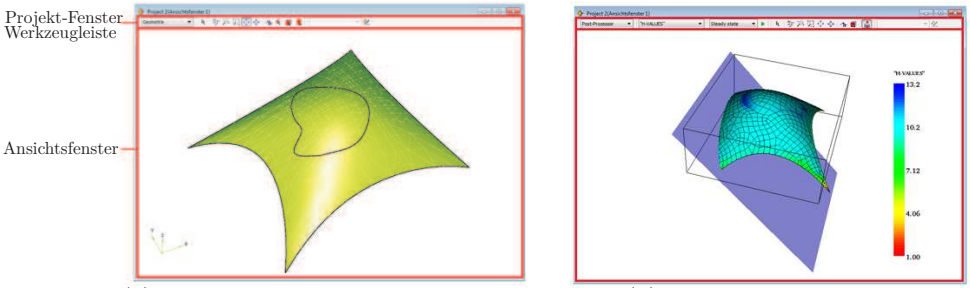


Abbildung 4.5: TUM.Visual Engine

Die Prä-Prozessor Schnittstelle basiert auf den Visualisierungsklassen von *Open CASCADE* [71] und kann zur Visualisierung des *geometrischen Hauptmodells* sowie des *diskretisierten Hauptmodells* in der Projekt-Datenbasis eingesetzt werden. Die Post-Prozessor Schnittstelle beruht auf den Bibliotheken von *VTK* [109] und visualisiert lediglich das *diskretisierte Hauptmodell*, allerdings mit entsprechender Farbcodierung und zahlreichen weiteren Visualisierungstechniken.

Die grafischen Prä- und Post-Prozessor Schnittstellen sind in der mit *Qt* [79] implementierten GUI in einem sogenannten Projekt-Fenster eingebettet, bei dem zwischen folgenden Ansichtsfenstern gewechselt werden kann:

- *Geometrie*: Prä-Prozessor Schnittstelle auf das *geometrische Hauptmodell* (vgl. Abbildung 4.6(a))
- *Netz*: Prä-Prozessor Schnittstelle auf das *diskretisierte Hauptmodell* (vgl. Abbildung 4.6(a))
- *Post-Prozessor*: Post-Prozessor Schnittstelle auf das *diskretisierte Hauptmodell* (vgl. Abbildung 4.6(b))



(a) Prä-Prozessor Fenster (b) PostProzessor Fenster

Abbildung 4.6: GUI - Projekt-Fenster

Innerhalb der GUI können mehrere Projekt-Fenster zu einem Projekt geöffnet werden. Zusätzlich können auch mehrere Projekte gleichzeitig geladen und angezeigt werden (siehe Abbildung 4.7).

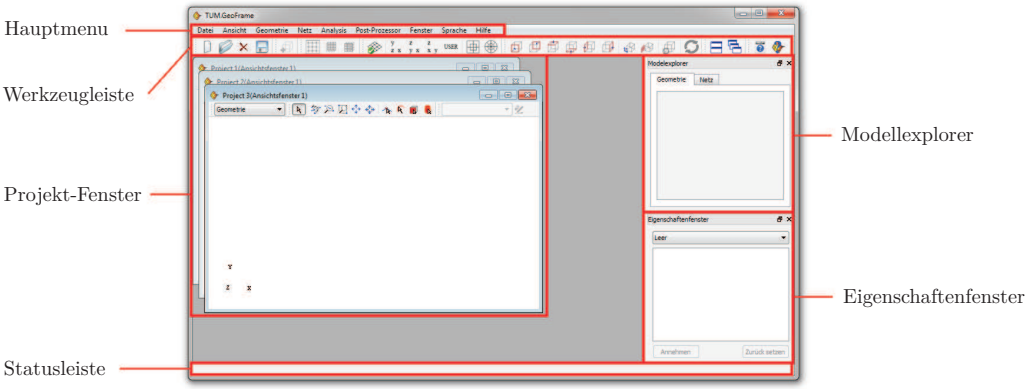
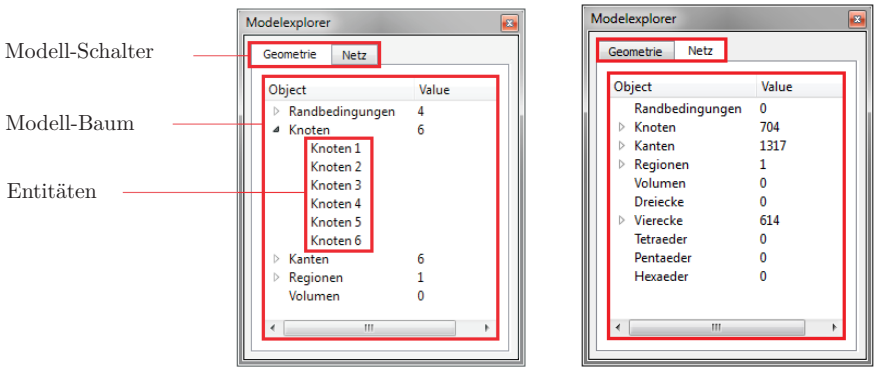


Abbildung 4.7: GUI - Anwendungsfenster

Die Interaktion mit den Entitäten im Prä-Prozessor Modus erfolgt über den *Modellexplorer* und das *Eigenschaftenfenster*, welche die ausgewählten grafischen Objekte mit der Benutzeroberfläche verbinden. Der *Modellexplorer* zeigt das *geometrische Hauptmodell* (Abbildung 4.8(a)) und das *diskretisierte Hauptmodell* (Abbildung 4.8(b)) als Baumstruktur an und dient zum Auswählen und Markieren einzelner Entitäten.

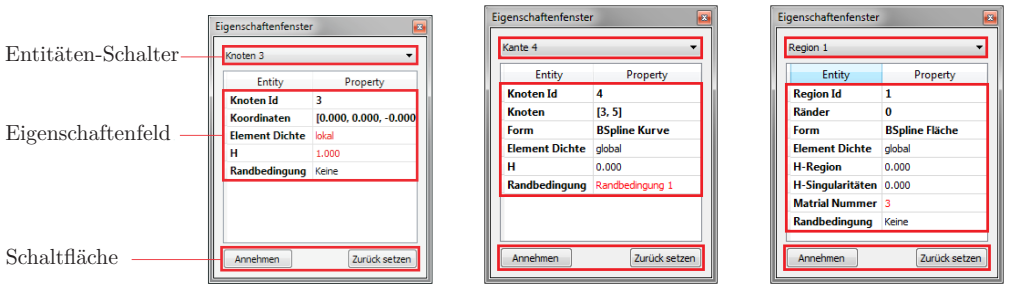


(a) Modellexplorer - Geometrie (b) Modellexplorer - Netz

Abbildung 4.8: GUI - Modellexplorer

Das Festlegen und Verändern der konstruktiven Eigenschaften einzelner Entitäten, wie z.B. der Randbedingungen und Materialwerte, erfolgt im *Eigenschaftenfenster* (Abbildung 4.9). Dort können die entweder im Projekt-Fenster oder im *Modellexplorer* ausgewählten Entitäten einzeln oder in Gruppen bearbeitet werden.

Darüber hinaus beinhaltet die *TUM.VisualEngine* zahlreiche weitere Dialogfenster und Einstellungsmöglichkeiten zur Steuerung des geometrischen Modells und



(a) Eigenschaftenfenster - Knoten (b) Eigenschaftenfenster - Kante (c) Eigenschaftenfenster - Region

Abbildung 4.9: GUI - Eigenschaftenfenster

einzelner Entitäten. Auf diese zusätzlichen Funktionalitäten wird in dieser Arbeit jedoch nicht näher eingegangen wird.

4.1.3 *TUM.ToolsEngine*

Die Benutzung einer umfangreichen konstruktiven Ingenieursanwendung durch einen ungeübten Anwender bedarf einer unterstützenden Hilfefunktionalität. Das Hilfesystem kann auf verschiedene Arten in die Anwendung integriert werden. Ein Überblick über verschiedene Implementierungskonzepte von grafischen Hilfesystemen steht in [108].

Die *TUM.HelpEngine* ist das kompilierte und integrierte Hilfesystem von *TUM.GeoFrame* und bietet interaktive Hilfe zum aktuell bearbeiteten Thema durch den Austausch von Schlüsselworten. Die *TUM.HelpEngine* ist beliebig durch eigene kompilierte Hilfeprojekte erweiterbar und beinhaltet standardmäßig Hilfeprojekte zu allen Kernmodulen. Abbildung 4.10 zeigt den Aufbau des Hilfesystems *TUM.HelpEngine*.

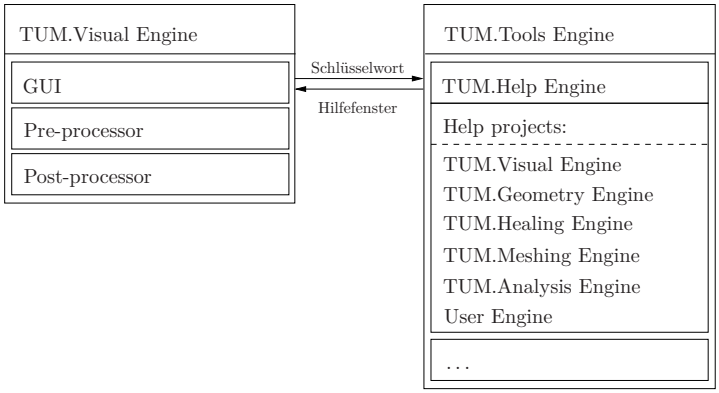


Abbildung 4.10: TUM.Help Engine

TUM.GeoFrame besitzt zum Schutz vor unkontrollierter Verbreitung und Benutzung ein Lizenzierungssystem (*TUM.LicenceEngine*), welches an die

TUM.VisualEngine angeschlossen ist. In der *TUM.LicenceEngine* wird aus der lokalen Festplattennummer mittels Primzahlenmultiplikation eine Seriennummer generiert, die über ein grafisches Eingabefenster registriert wird. Die GUI prüft über einen einfachen Funktionsaufruf das Vorhandensein der korrekten Seriennummer und deaktiviert im negativen Fall alle Exportfunktionen. Abbildung 4.11 zeigt den Aufbau des Lizenzierungssystems *TUM.LicenceEngine*.

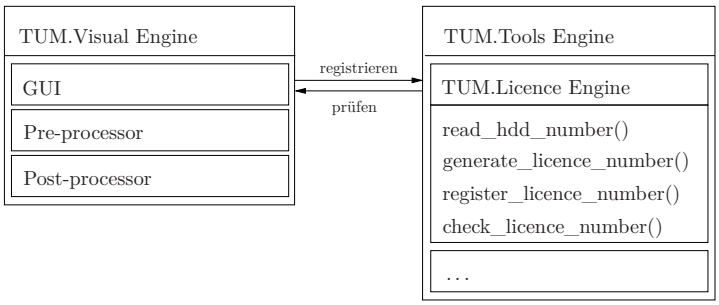


Abbildung 4.11: TUM.Licence Engine

4.1.4 *TUM.HealingEngine*

Der Austausch von CAD-Daten zwischen unterschiedlichen Konstruktionssystemen über standardisierte Dateiformate resultiert häufig in fehlerhaften oder „unsauberen“ geometrischen Modellen. Solche fehlerhaften Modelle besitzen dann beispielsweise Klaffungen, Überlappungen, doppelte Kanten und Flächen, topologisch nicht verbundene Regionen. In der Praxis werden solche Fehler oft durch mühevolle Handarbeit beseitigt.

TUM.GeoFrame besitzt einen integrierten Heilungskern, der fehlerhafte Entitäten nach eigens definierbaren Suchkriterien identifiziert und die fehlerhaften Stellen automatisch beseitigt. Die innerhalb der Rahmen-Software *TUM.GeoFrame* implementierten Heilungsprozeduren sind nicht Teil dieser Arbeit. Im folgenden wird von geometrisch und topologisch “sauberen” Modellen ausgegangen. Ein Überblick über die vollautomatischen Heilungs-Algorithmen in *TUM.GeoFrame* ist in [45] zu finden.

4.1.5 *TUM.MeshingEngine*

Die Diskretisierung von geometrischen Modellen gehört zum Forschungsschwerpunkt dieser Arbeit und ist in den nachfolgenden beiden Kapiteln im Detail beschrieben. Die Implementierung der entwickelten Diskretisierungsalgorithmen erfolgt innerhalb der *TUM.MeshingEngine*, die verschiedene interne wie externe Vernetzungstechniken zu einem leistungsfähigen Vernetzungskern kombiniert. Die Anbindung der *TUM.MeshingEngine* an die Projekt-Datenbasis *Project* ist in Abbildung 4.12 skizziert. Für die Vernetzung des *geometrischen Hauptmodells* kann neben den Geometrieinformationen ein Hintergrundnetz importiert werden. Die Ausgangsinformationen werden innerhalb des Kernmoduls *MeshGen*

in einer eigenen Datenbasis verarbeitet. Nach Abschluss des Vernetzungsvorgangs werden die numerischen Daten des Finite-Elemente Netzes aus dem Kernmodul *MeshGen* in die Projekt-Datenbasis *Project* zurück geschrieben.

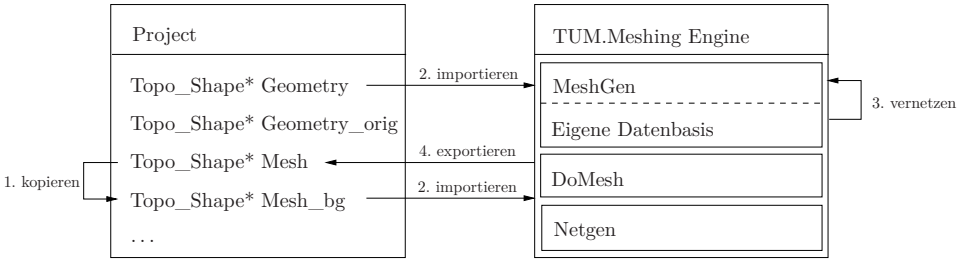


Abbildung 4.12: TUM.Meshing Engine

4.1.6 *TUM.AnalysisEngine*

TUM.GeoFrame dient als Rahmen-Software für verschiedene numerische Simulationsmethoden. Hierbei kann die Projekt-Datenbasis *Project* für die Formulierungen von verschiedenen mechanischen Aufgabenstellungen angesetzt werden. Der Forschungsschwerpunkt liegt in der Unterstützung der klassischen Finite-Elemente-Methode sowie der Finite-Elemente-Methode hoher Ordnung durch die Entwicklung geeigneter Vernetzungsprozeduren. Daneben unterstützt die *TUM.AnalysisEngine* jedoch auch geometriebasierte Simulationsmethoden (z.B. Iso-Geometric-Analysis [52]) durch Vollzugriff auf die geometrischen Rohdaten der zugrunde liegenden BSpline- und NURBS-Objekte, sowie Embedded-Domain-Methoden (z.B. Finite-Cell-Methode [76]) durch die Bereitstellung von leistungsfähigen Innen-Außen-Testfunktionen und Abstandsfunktionen zum Rand. Weiterführende Details zu den bereit gestellten Prozeduren sind in [96] zu finden.

4.1.7 *TUM.InterfaceEngine*

Die Anbindung der Programmmodule *TUM.HealingEngine*, *TUM.MeshingEngine* und *TUM.AnalysisEngine*, sowie beliebiger externer Programmmodule erfolgt über die Programmschnittstelle *TUM.InterfaceEngine*. Alle Programmmodule werden dabei von der Klasse *Meta_Project* abgeleitet und als dynamische Bibliotheken in die GUI geladen. Die Anbindung der implementierten Algorithmen erfolgt über Einträge in der Menü- bzw. Werkzeugleiste. Dazu können beliebige Schaltflächen definiert werden, die anschließend in der Menü- bzw. Werkzeugleiste eingetragen werden. Der Austausch der Programmmodule mit der Problemformulierung in der Projekt-Datenbasis erfolgt über Zugriffsfunktionen auf die *TUM.GeometryEngine*. Abbildung 4.13 zeigt den Aufbau eines Benutzerprojektes (*User Module*) und seiner Superklasse *Meta_Project*. Durch die Einbindung des Benutzerprojektes in die grafische Benutzeroberfläche werden die Benutzerprozeduren (*Slots*) mit den neu definierten Schaltflächen verbunden. Die Interaktion der Benutzerprozedur mit der Projekt-

Datenbasis ist über eine vererbte Zugriffsfunktion (*get_current_project*()) möglich.

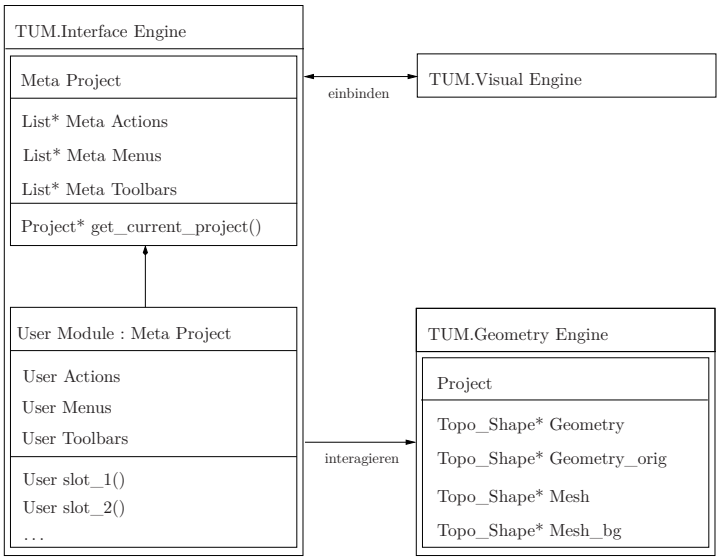


Abbildung 4.13: TUM.Interface Engine

4.2 Schnittstellen

Die Interaktion der physikalischen Problemformulierung mit externen Programmen erfordert den Austausch von geometrischen und konstruktiven Daten. Als Schnittstellen für den Austausch sind verschiedene Dateiformate implementiert, sowie eine Programmschnittstelle für die direkte Kommunikation mit der Projekt-Datenbasis. Dieser Abschnitt beschreibt die entwickelten Schnittstellen für *TUM.GeoFrame*

4.2.1 Geometrie-Schnittstelle

Wie in Abschnitt (4.1.1) beschrieben, untergliedern sich die Entitäten der Projekt-Datenbasis in topologisch und geometrische Entitätsklassen. Handelt es sich bei der zu untersuchenden Entität um eine Kante oder eine Fläche, so besitzt diese Entität innerhalb des Geometriekerns detaillierte geometrische Eigenschaften. Um die geometrischen Informationen einer Entität vollständig im Topologiekern vorzuhalten, werden den topologischen Kanten und Flächen geometrische Attribute zugewiesen. Diese Attribute können aus den Objekten des Geometriekerns ausgelesen werden. Umgekehrt können geometrische Objekte aus dem topologischen Objekt mit dem dazu gehörigen geometrischen Attribut erstellt werden. Abbildung 4.14 zeigt das Zusammenspiel der topologischen und geometrischen Klassen von Kanten.

Analog zu den Kanten werden geometrische Flächen durch topologische Flächen mit geometrischem Attribut beschrieben und umgekehrt (siehe Abbildung 4.15).

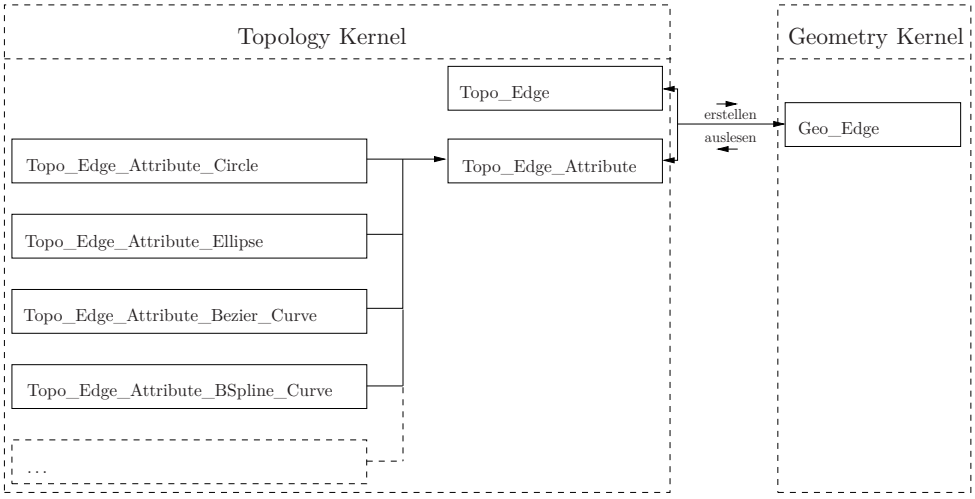


Abbildung 4.14: Geometrie Pipeline - Kanten

Dabei spielt es für den Erstellungs- bzw. Auslesealgorithmus keine Rolle, ob es sich bei der topologischen Fläche um ein Dreieckselement, ein Viereckselement oder eine topologische Region handelt.

4.2.2 Dateischnittstellen

Die Kommunikation von verschiedenen Konstruktionsprogrammen erfolgt in der Praxis oft über standardisierte Dateiformate. Dabei ist es mit Hilfe dieser standardisierten Formate nicht immer möglich, die programminterne Datenbasis vollständig und effektiv abzubilden. Daher besitzen alle gängigen Konstruktions- und Berechnungssysteme in der Regel zusätzliche interne Dateiformate. In *TUM.GeoFrame* können nachfolgend aufgelistete standardisierte Dateiformate direkt über die *Open CASCADE* internen Parser in die entsprechenden Objekte des Geometriekerns geladen werden:

- Iges [2]
- Step [3]
- BRep [1]
- CSFDB [71]
- STL [4]
- VRML [5]

Aus den Geometriekern-Objekten wird wie oben beschrieben das topologische Modell im Topologiekern generiert. Der Austausch zwischen internen Dateiformaten erfolgt direkt im Topologiekern. Hierbei werden die topologischen Objekte

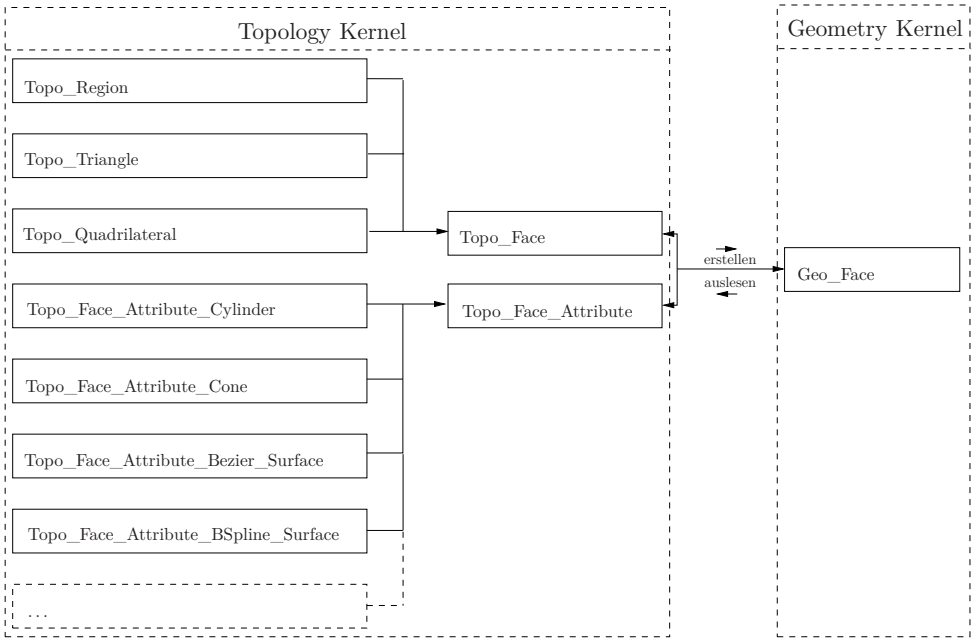


Abbildung 4.15: Geometrie Pipeline - Flächen

samt Attributen aus den externen Dateien gelesen und in die Objekte des Geometriekerns umgewandelt. Abbildung 4.16 zeigt das Zusammenwirken der unterschiedlichen Dateischnittstellen.

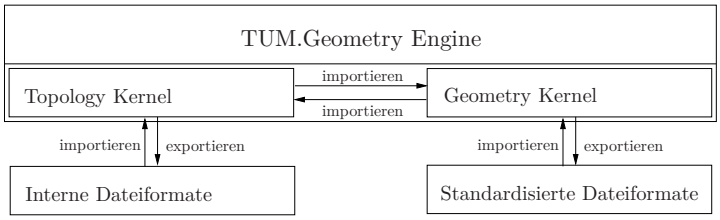


Abbildung 4.16: Dateischnittstellen

4.2.3 Programmschnittstellen

Die Anbindung von externen Programmmodulen erfordert

- den Austausch von geometrischen Daten mit der Projekt-Datenbasis,
- die Möglichkeit zur Darstellung von Rückmeldungen sowie
- den Zugriff auf verschiedene Anweisungsfunktion zum Aktualisieren der grafischen Benutzeroberfläche.

Der Zugriff auf die Projekt-Datenbasis ist wie in Abschnitt (4.1.1) beschrieben implementiert. Die Darstellung von Rückmeldungen (vgl. Abbildung 4.17) ist über die Basisklasse *Messages* möglich, auf die über den globalen Projekt-Zeiger (vgl. Abbildung 4.3) zugegriffen werden kann.

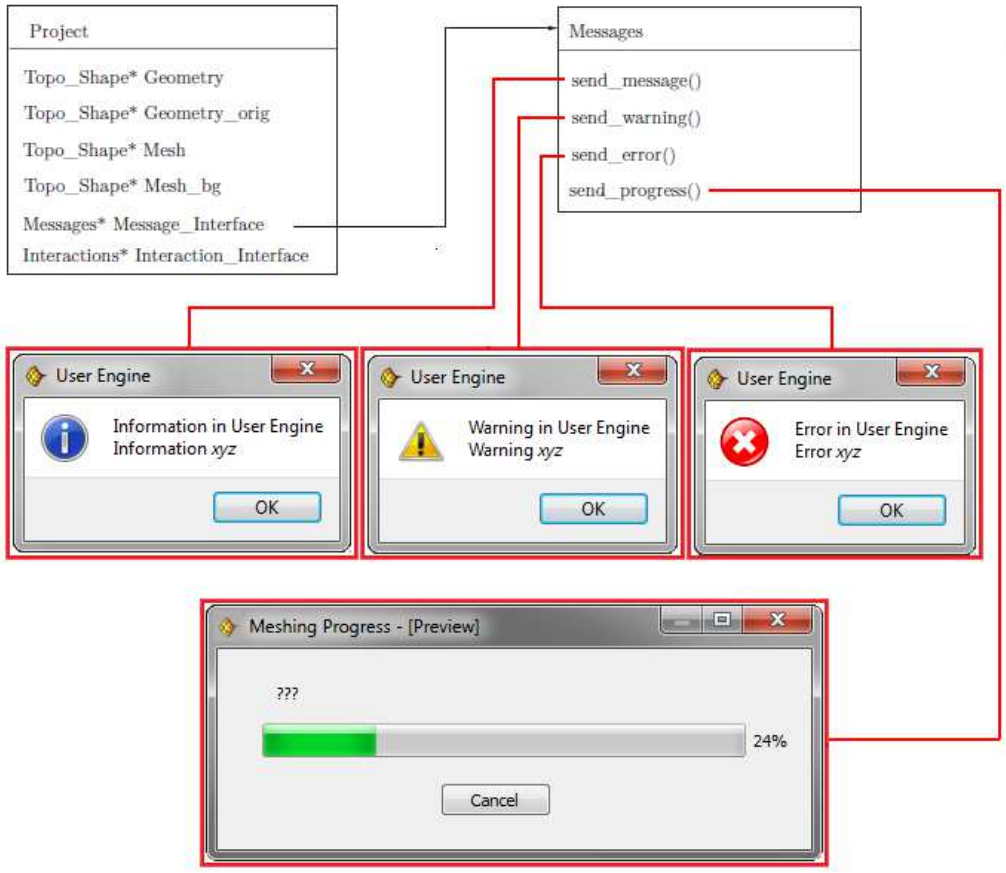


Abbildung 4.17: Programmschnittstelle - Messages

Die Aktualisierung der GUI ist über die Basisklasse *Interactions* gewährleistet. Hier sind Funktionsaufrufe zur Aktualisierung des Projekt-Fensters, des *Modellexplorers* und des *Eigenschaftenfensters* implementiert.

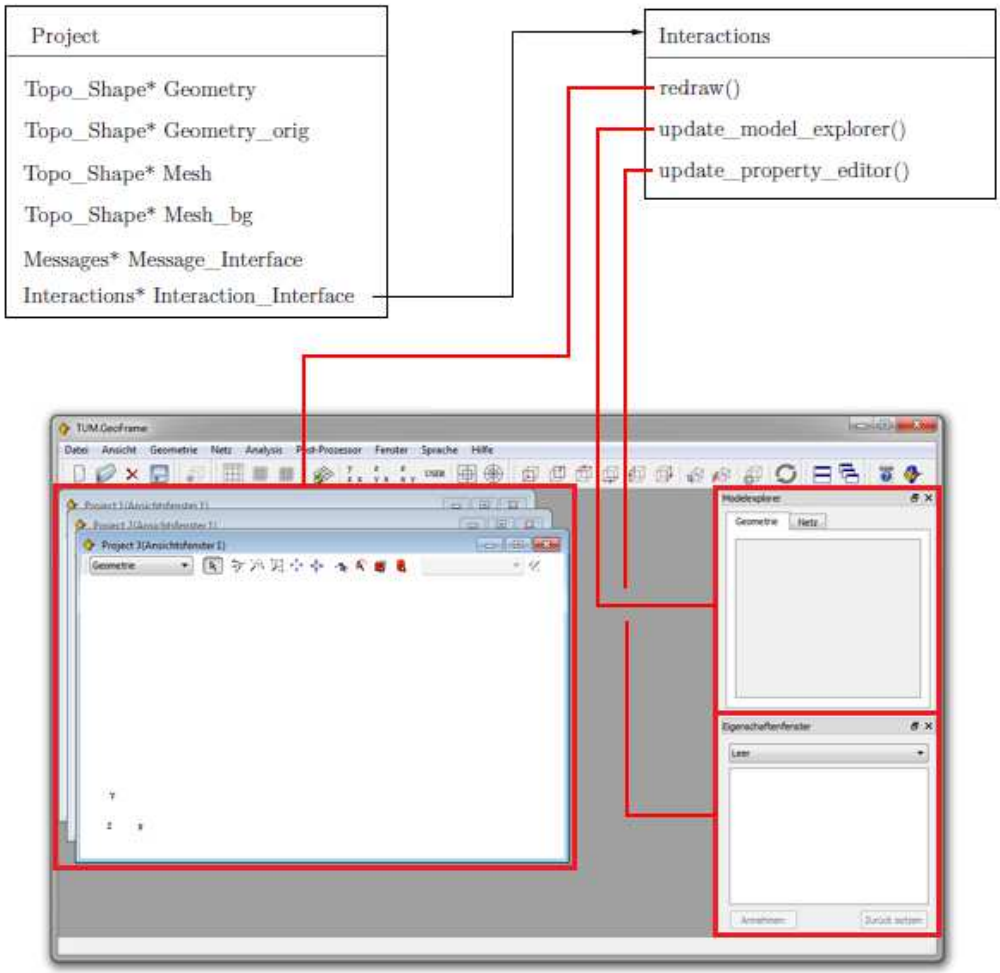


Abbildung 4.18: Programmschnittstelle - Interactions

Kapitel 5

Generierung von Oberflächennetzen

Nachdem in den vorangegangenen Kapiteln die für das Verständnis dieses Kapitels benötigten Grundlagen dargelegt wurden, soll nun der in das Rahmenwerk *TUM.GeoFrame* (vgl. Kapitel 4) implementierte Algorithmus zur Vernetzung von beliebig gekrümmten geometrischen Flächen im euklidischen Raum $\mathbb{R}^3$ erläutert werden. Der im Rahmen dieser Arbeit entwickelte Algorithmus basiert dabei im Wesentlichen auf den Ideen und Ausführungen von [18, 82, 91].

5.1 Einführung

Die Generierung von Oberflächennetzen gliedert sich prinzipiell in Techniken zur Generierung von strukturierten Netzen, auch (konforme) Gitter-Generierung genannt, und Techniken zur Generierung von unstrukturierten Netzen, auch Freivernetzung genannt.

Bei strukturierten Vernetzungstechniken [105, 106] haben alle inneren Knoten die gleiche Anzahl von angrenzenden Elementen, allerdings lassen sich für viele komplexe Strukturen keine konformen strukturierten Gitter erzeugen. Daher sind strukturierte Vernetzungstechniken als alleinige Netzgenerierungstechnik für allgemeine geometrische Problemstellungen nicht ausreichend.

Unstrukturierte Vernetzer haben keine einschränkenden Bedingungen an die Konnektivität in den inneren Knoten und lassen sich grundsätzlich für jede beliebige geometrische Problemstellung anwenden. Dafür besitzen die Netze in der Regel einen höheren Anteil an verzerrten Elementen, was sich nachteilig auf die numerischen Berechnungen auswirken kann. Die genaue Platzierung der inneren Knoten spielt daher die zentrale Rolle bei der Generierung von geeigneten Finite-Elemente-Netzen.

Die Generierung von unstrukturierten Dreiecksnetzen basiert in der Regel auf einer oder mehreren Grundtechniken, wie der

- Delauney-Triangulierung [59, 110], bei der die inneren Knoten zu Beginn nach einem bestimmten Schema positioniert werden und anschließend über ein Umkreiskriterium [37] zu Dreiecken verknüpft werden, der

- Advancing-Front-Methode [63, 61], bei der neue Knoten und Elemente vom aktuellen Gebietsrand ausgehend generiert werden, oder der

- Gebietsteilungstechnik [18, 82, 91], welche die Grundlage des im Rahmen dieser Arbeit entwickelten Algorithmus ist.

Das bei weitem beliebteste Kriterium bei der Generierung von Dreiecksnetzen ist das Delauney-Kriterium [37]. Dieses Kriterium besagt, dass kein Knoten innerhalb des Umkreises sein darf, der durch die drei Knoten eines nicht angrenzenden Elementes verläuft. Das Delauney-Kriterium selbst ist noch kein Vernetzungsalgorithmus, sondern wird auf eine zuvor generierte Menge von Knoten angewendet. Die Platzierung der inneren Knoten kann auf unterschiedliche Weise erfolgen, beispielsweise als reguläres Gitter, aus dem Element-Umkreis [35, 85, 95], aus dem Voronoi-Diagramm [84, 58], oder einer Reihe weiterer Platzierungstechniken. Obwohl das Delauney-Kriterium einfach zu implementieren und anzuwenden ist, resultiert das Generieren der inneren Knoten bei komplexen geometrischen Strukturen speziell in Randnähe oft in stark verzerrten Elementen.

Eine weitere beliebte Familie der Netzgenerierungsalgorithmen ist die Advancing-Front-Methode [63, 61, 78]. In dieser Methode werden die Dreiecke stets vom aktuellen Gebietsrand in Richtung Gebietinneres abgespalten, so dass sich der Gebietsrand mit fortlaufendem Vernetzungsfortschritt nach innen bewegt. Diese Methode hat im Vergleich zu anderen Techniken ihre Stärke vor allem in der Qualität der Elemente am Gebietsrand. Bei Gebieten mit mehreren Rändern (z.B. Region mit Löchern) ist die Advancing-Front-Methode aufgrund der vielen benötigten Verschneidungsoperationen jedoch teuer und erzeugt im Fall von unterschiedlichen Netzdichten (adaptive Vernetzung) speziell in den Übergangsbereichen stark verzerrte Elemente.

Die Gebietsteilungstechnik [18, 82, 91] generiert die inneren Knoten entlang neuer Teilungskanten, die durch eine rekursive Teilung des Vernetzungsgebietes entstehen. Das Dreiecksnetz entsteht durch eine endliche Wiederholung dieser rekursiven Prozedur, bis das Vernetzungsgebiet nur noch aus Dreiecken besteht. Diese Vorgehensweise hat im Vergleich zu den anderen beiden Grundtechniken ihre Stärken vor allem bei adaptiver Vernetzung und bei komplexen, beliebig gekrümmten Strukturen mit vielen Strukturrändern, wie in vielen realen Problemstellungen der Fall.

Die Generierung von unstrukturierten Viereckснetzen gliedert sich in direkte und indirekte Techniken. Bei den indirekten Techniken wird zunächst ein Dreiecksnetz generiert, welches anschließend in ein Viereckснetz konvertiert wird. Direkte Techniken platzieren die Viereckselemente direkt auf dem Vernetzungsgebiet, ohne vorher ein Dreiecksnetz generieren zu müssen.

Die bekanntesten direkten Viereckvernetzungstechniken sind die Dekompositionstechniken, bei denen das Gebiet zunächst in Unterstrukturen unterteilt wird, z.B. durch Quadtrees [16], Polygone [102, 33, 70] oder Mittelachsen [103], auf denen mit Hilfe von verschiedenen Vorlagen Viereckselemente platziert werden.

Eine weit verbreitete Alternative der direkten Vierecks-Vernetzung ist, wie schon bei der Dreiecks-Netzgenerierung, die Advancing-Front-Methode. Diese projiziert die Viereckselemente entweder durch Verschiebung des Gebietsrandes in Richtung Gebietinneres [116], oder durch das Einfügen von kompletten Elementreihen in das Berechnungsgebiet [26, 32, 112].

Obwohl die direkten Vierecks-Netzgenerierungstechniken oft gute Elmente am Gebietsrand generieren, sind die beteiligten geometrischen Operationen bei komplexen Vernetzungsgebieten mit vielen Regionsrändern in der Regel sehr teuer und Netze mit unterschiedlicher Elementdichte nur begrenzt möglich. Deutlich schneller und flexibler sind diesbezüglich die indirekten Konvertierungsschemata, da alle geometrischen Operationen nur lokal statt finden und die teuren Verschneidungsoperationen wegfallen.

Bei den indirekten Konvertierungsschemata wird prinzipiell unterschieden zwischen

- Methoden, bei denen die Knoten des Dreiecksnetzes topologisch erhalten bleiben und Vierecke durch Zusammenfassen [54] oder Unterteilen [83, 60] der Dreiecke generiert werden und
- freien Konvertierungsmethoden, bei denen das Viereckснetz durch eine Manipulation der Topologie des Dreiecksnetzes, beispielsweise durch Einfügen von zusätzlichen Knoten oder Teilen bzw. Zusammenfassen von Randkanten [75], generiert wird.

Ersteres Prinzip ist zwar leichter zu implementieren, dafür lassen sich mit dem zweiten Prinzip qualitativ hochwertigere Netze speziell in Randnähe generieren.

Bei der Beschreibung von Oberflächennetzen für die Finite-Elemente-Methode hoher Ordnung (vgl. Abschnitt 3.3.4) muss für jedes Element die exakte Geometrie aus dem geometrischen Modell bekannt sein. Dies erfolgt entweder durch eine Kopplung des Finite-Elemente-Netzes mit dem zugrunde liegenden Geometrie-Modellierer, oder durch eine eigene Diskretisierung der Elementoberflächen [12].

5.2 Gebietsteilungstechnik auf Freiformflächen

Im Folgenden wird der im Rahmen dieser Arbeit entwickelte Algorithmus zur Vernetzung von Freiformflächen für die h und die p-Version der Finite-Elemente-Methode beschrieben (vgl. Kapitel 3). Der Algorithmus ist eine Erweiterung des Netzgenerators *DoMesh*, bei dem durch Einführung von Längen- und Winkelberechnungen auf Basis der lokalen Metrik (vgl. Abschnitt 2.4.3) adaptive Netze

mit hoher Elementqualität auf beliebigen polygonal berandeten Freiformflächen generiert werden.

5.2.1 Topologisches Modell

Das topologische Model der zu vernetzenden Oberflächenstruktur ist eine Erweiterung der Randdarstellung (vgl. Abbildung 5.1).

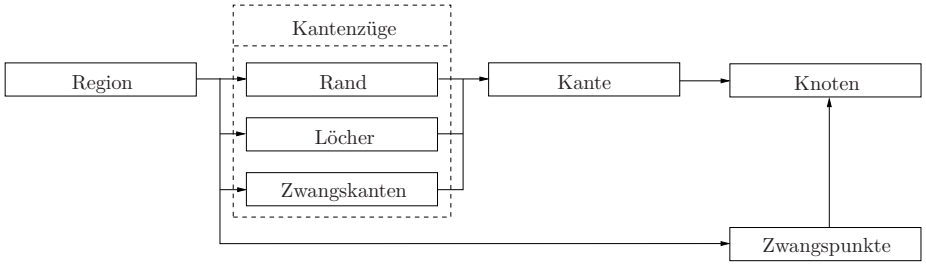


Abbildung 5.1: Topologisches Oberflächenmodell

Modelle mit mehreren Regionen besitzen Konnektivität und resultieren in konformen Netzen, sobald sich benachbarte Regionen gemeinsame topologische Entitäten teilen. Diese Entitäten können gemeinsame Regionsränder in Form von äußeren oder inneren Randkanten sein (vgl. Abbildung 5.2), oder im Fall von geschnittenen Regionen topologische Zwangskanten sein, die zur Regionsbeschreibung hinzu gefügt werden (vgl. Abbildung 5.3).

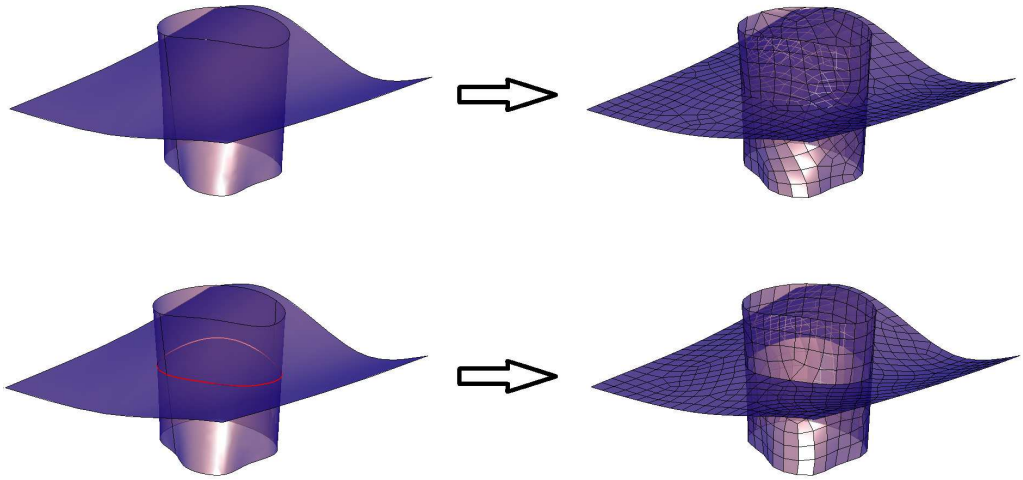


Abbildung 5.2: Konnektivität mit und ohne innere Randkante

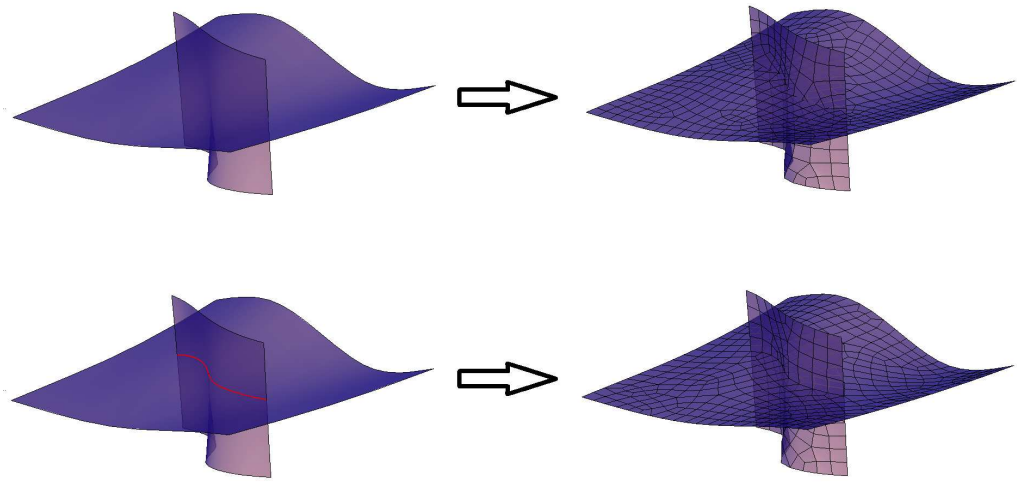


Abbildung 5.3: Konnektivität mit und ohne Zwangskante

Die Konnektivität zwischen Regionen und geschnittenen Kurven wird über Zwangspunkte definiert, die unverändert als Elementknoten in die Vernetzung übernommen werden (vgl. Abbildung 5.4).

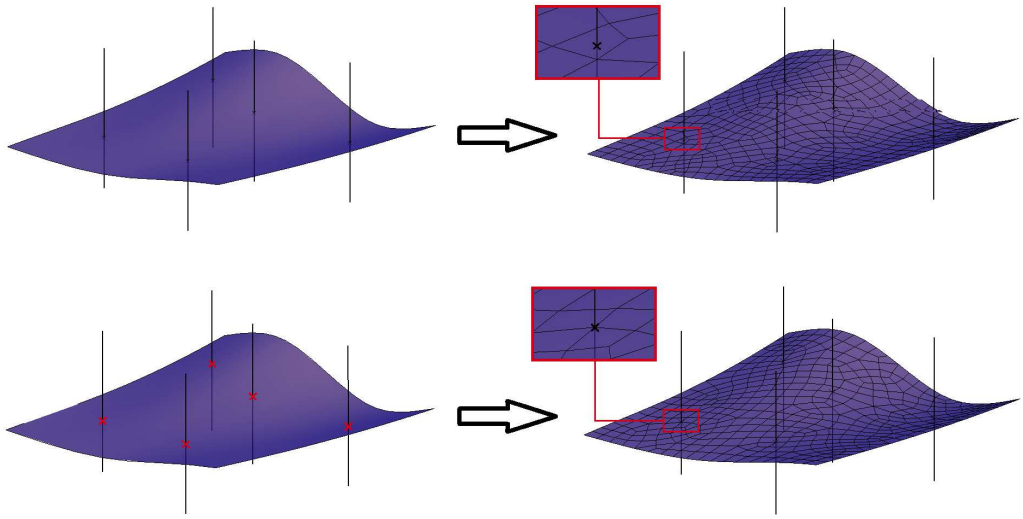


Abbildung 5.4: Konnektivität mit und ohne Zwangspunkte

5.2.2 Elementdichte-Verteilung

h-Wert

Die Verteilung der Elementdichte des generierten Netzes wird durch die *h*-Wert Funktionen, geometrische Besonderheiten des Modells und durch ein adaptives Vernetzungsschema definiert. Der *h*-Wert beschreibt dabei die maximale Länge ei-

ner Elementkante und ist umgekehrt proportional zur Elementdichte. Der h-Wert wird global definiert, kann aber auf einzelnen Entitäten lokal verändert werden, um lokale Netzverfeinerungen oder Netzvergröberungen zu erreichen. Sind für eine Entität mehrere h-Werte gleichzeitig definiert so gilt folgende Priorität:

1. Knoten-h-Wert
2. Kanten-h-Wert
3. Regions-h-Wert
4. Globaler h-Wert

Einfacher Lauf und adaptiver Lauf

Die h-Werte werden vor dem eigentlichen Vernetzungslauf über eine Verteilungsfunktion auf das gesamte Vernetzungsgebiet abgebildet und geglättet. Die Verteilungsfunktion wird dabei entweder über die Strukturknoten des geometrischen Modells, bestehend aus den Knoten der Rand- und Zwangskanten, sowie allen Zwangspunkten (vgl. Abschnitt 5.2.1) beschrieben (einfacher Lauf), oder in einem zusätzlichen Hintergrundnetz diskretisiert (adaptiver Lauf).

In einem einfachen Lauf werden die lokalen h-Werte der Entitäten sowie der globale h-Wert nach oben aufgeführter Priorität auf die Strukturknoten verteilt. Um den Einfluss von kurzen Kanten (vgl. Abbildung 5.5) auf die Elementqualität zu reduzieren, werden anschließend alle h-Werte in den Strukturknoten so weit herab gesetzt, dass sie das λ-fache der Länge der längsten angrenzenden Strukturkante nicht überschreiten. Ein Wert von $\lambda = 1.4$ hat sich empirisch bewährt und sorgt dafür, dass einerseits stark verzerrte Elemente in Bereichen mit kurzen Kanten vermieden werden und andererseits der Einfluss von kurzen Kanten auf den globalen h abgefedert wird. Da eine Auflösung der h-Wert Verteilung über die Strukturknoten nur eine sehr grobe Diskretisierung der h-Wert Funktion zulässt, haben kleine h-Werte in einzelnen Bereichen Einfluss auf den Elementverlauf in der gesamten Struktur (vgl. Abbildung 5.6).

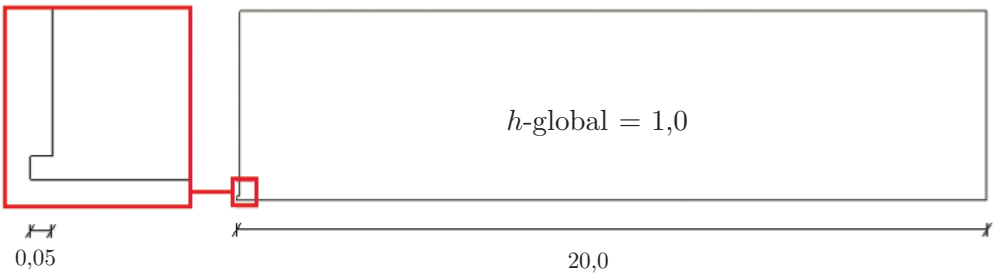


Abbildung 5.5: Geometrie mit kurzer Kante

Bei einem adaptiven Lauf werden die h-Werte in den Knoten eines Hintergrundnetzes diskretisiert. Dadurch ist eine deutlich feinere Auflösung der Verteilungsfunktion möglich und Schwankungen in der h-Wert Verteilung können

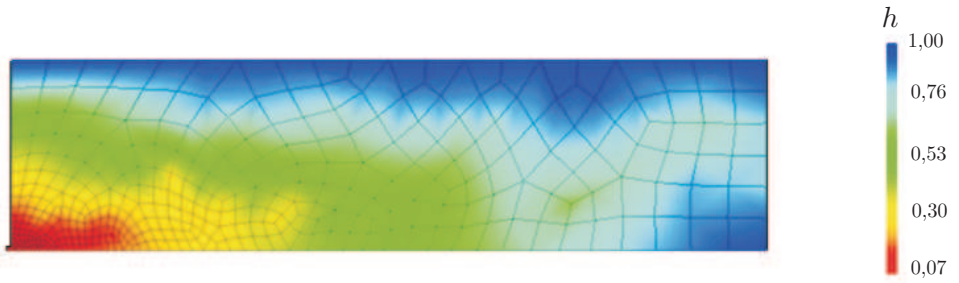


Abbildung 5.6: h-Wert Verteilung - einfacher Lauf

lokal geglättet werden. Die Glättungsfunktion wurde mit dem Ziel entwickelt, sowohl Elemente mit guten Innenwinkel-Verhältnissen für die h-Version der Finite-Elemente-Methode zu generieren, als auch schnelle Übergänge von groben auf feine Elemente für die p-Version der Finite-Elemente-Methode zuzulassen. Die Propagation der h-Werte basiert auf den folgenden beiden Kontrollparametern:

- *Propagationsfaktor:* Definiert den maximalen Gradienten der h-Wert Verteilungsfunktion im Inneren des Vernetzungsgebietes und kontrolliert die Elementqualität in den Übergangsbereichen von groben auf feine Elemente.
- *Randkantenfaktor:* Kontrolliert den Einfluss von kurzen Kanten auf die Elementqualität am Rand durch Skalierung der oberen h-Wert Schranke in den Randknoten des Hintergrundnetzes. Die h-Wert Schranke ist dabei definiert durch die Länge der kürzesten angrenzenden Strukturkante.

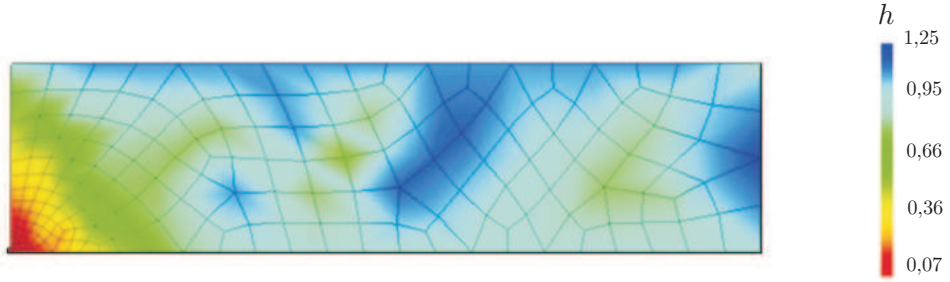
Abbildung 5.7 zeigt den Einfluss der beiden Kontrollparameter auf die h-Wert Verteilung bei einem adaptiven Lauf mit Hintergrundnetz.

5.2.3 Randanbindung

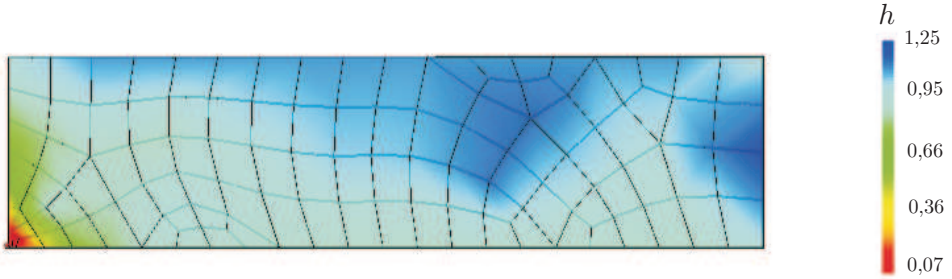
Die Ausgangsgeometrie für den Algorithmus zur Triangulierung der Oberflächenstruktur beruht auf geschlossenen Polygonen. Um das in Abschnitt 5.2.1 eingeführte topologische Modell verarbeiten zu können, werden zunächst alle Löcher, Zwangskanten und Zwangspunkte mit Hilfe von neuen Anbindungskanten an den äußeren Rand der Struktur angebunden. In einem zweiten Schritt werden aus der Originalgeometrie und den Anbindungskanten neue geschlossene Polygonzüge definiert. Diese sind so orientiert, dass der Rand des Polygons im Gegenuhrzeigersinn durchlaufen wird und das Strukturinnere linksseitig des Polygonrandes liegt (vgl. Abbildung 5.8).

5.2.4 Randunterteilung

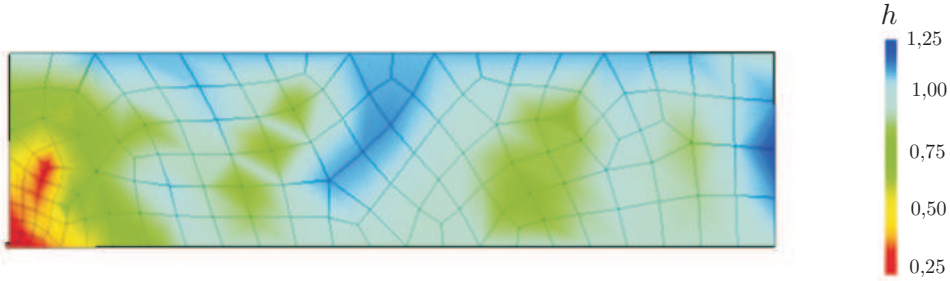
Nach Anbindung der inneren Strukturen an den Regionsrand werden die Strukturkanten der neu definierten Polygonzüge entsprechend der geglätteten h-Werte



(a) Propagationsfaktor = 1,4; Randkantenfaktor = 1,4



(b) Propagationsfaktor = 2,4; Randkantenfaktor = 1,4



(c) Propagationsfaktor = 1,4; Randkantenfaktor = 5,0

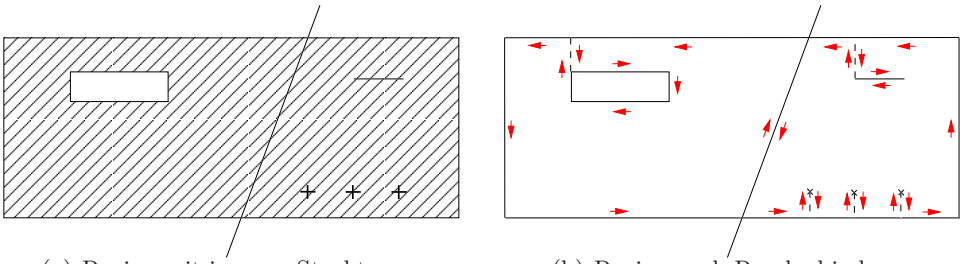
Abbildung 5.7: h-Wert Verteilung - adaptiver Lauf

(vgl. Abschnitt 5.2.2) unterteilt. Die Platzierung der neuen Teilungsknoten (vgl. Abbildung 5.9) erfolgt analog zur Platzierung der inneren Knoten während der Triangulierung und wird in Abschnitt 5.2.5 im Detail erläutert.

Die Darstellung der zugrunde liegenden geometrischen Kurven für die Randunterteilung beruht auf einer natürlichen Parametrisierung der Parameterdarstellung der einzelnen Kurvenobjekte im Bogenlängenmaß (vgl. Abschnitt 2.4.2). Dadurch werden für die Abbildung und Umkehrabbildung die tatsächlichen Längenparameter angesetzt und Schwankungen der Parametergeschwindigkeit in der Parameterdarstellung bei stark gekrümmten Freiformkurven abgefangen.

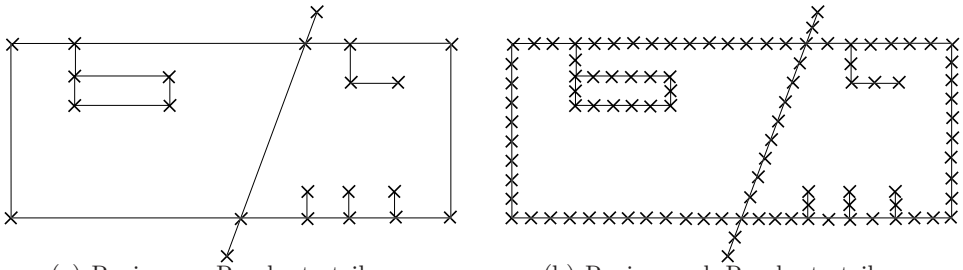
Referenzierte Teilung

Alternativ zur freien Kantenteilungstechnik nach der *h*-Wert Verteilung kann die Unterteilung einer sogenannten *Slave-Kante* über eine Referenz auf eine übergeordnete *Master-Kante* vorgegeben werden. Die *Slave-Kante* wird dadurch mit



(a) Region mit inneren Strukturen (b) Region nach Randanbindung

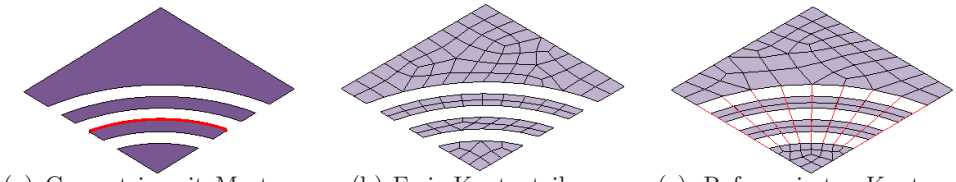
Abbildung 5.8: Randanbindung



(a) Region vor Randunterteilung (b) Region nach Randunterteilung

Abbildung 5.9: Randunterteilung

der selben Anzahl an Unterteilungen und äquivalenten Längenverhältnissen unterteilt, wie die entsprechende *Master-Kante*. Eine *Master-Kante* kann dabei von mehreren *Slave-Kanten* gleichzeitig referenziert werden. Abbildung 5.10 illustriert den Einfluss von referenzierten Kanten auf einem rechteckigen Gebiet mit mehreren kreisförmigen Aussparungen von unterschiedlichem Radius und unterschiedlicher Bogenlänge.



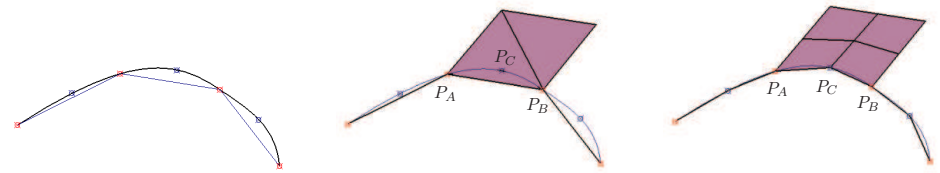
(a) Geometrie mit Masterkante (rot) (b) Freie Kantenteilung (c) Referenzierte Kantenteilung

Abbildung 5.10: Referenzierte Teilung bei Viereckselnetz Generierung

Vorgegebene Unterteilung

Neben der freien und referenzierten Teilung der Strukturkanten kann der Elementverlauf am Gebietsrand durch eine vorgegebene Unterteilung der Randkanten bestimmt werden. Anschließend wird die weitere Teilung der betroffenen Kanten innerhalb des Vernetzungslaufs unterdrückt. Während der Konvertierung in Viereckselemente wird jede Dreieckskante von Knoten P_A nach Knoten P_B jedoch genau einmal unterteilt. Bei der Generierung von Viereckselnetzen werden

die Kanten bei vorgegebener Unterteilung daher mit einem inneren Knoten P_C angereichert, der auf der geometrischen Kurve innerhalb des Intervalls zwischen Knoten P_A und Knoten P_B liegt. Die Teilung der betroffenen Kanten wird dadurch während der Dreiecksnetz-Generierung unterdrückt. Bei der Konvertierung in Viereckselemente wird jede Kante jedoch einmal in zwei Kanten von Knoten P_A nach Knoten P_C und von Knoten P_C nach Knoten P_B unterteilt (vgl. Abbildung 5.11).



(a) Vorgegebene Teilung mit inneren Knoten (blau)

(b) Dreiecksnetz

(c) Viereckesnetz

Abbildung 5.11: Vorgegebene Teilung bei Viereckesnetz Generierung

5.2.5 Triangulierung

Rekursive Gebietsteilungstechnik

Die rekursive Gebietsteilungstechnik ist das Herzstück der Dreiecksvernetzung und basiert auf zwei grundlegenden Prozeduren, zum einen für die Abspaltung von einzelnen Dreiecken (*chop*) und zum anderen für die Teilung des aktuellen Vernetzungsgebietes (*divide*). Beide Prozeduren werden durch ein rekursives Schema miteinander verknüpft (vgl. Abbildung 5.12). Durch eine endliche Wiederholung des Schemas bis zum Erreichen des Abbruchkriteriums wird das Gebiet vollständig in Dreieckselemente unterteilt.

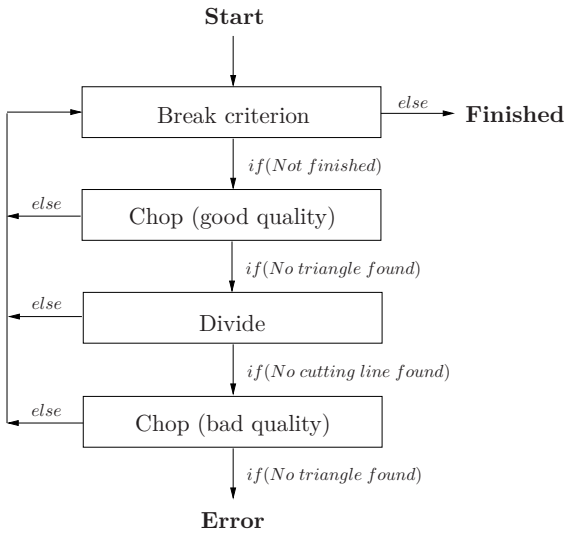
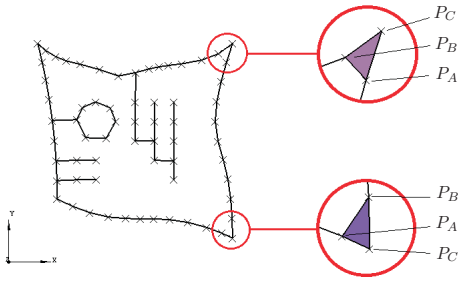


Abbildung 5.12: Grundschema der rekursiven Gebietsteilungstechnik

Die *chop*-Prozedur verbindet drei aufeinender folgende Randknoten und spaltet die so erstellten Dreiecke vom aktuellen Gebietsrand ab (vgl. Abbildung 5.13(a)).



(a)

$$a_1 = \frac{\min\left(h_{P_A}, \|P_A P_B\|\right)}{\max\left(h_{P_A}, \|P_A P_B\|\right)}$$

$$a_2 = \frac{\min\left(h_{P_B}, \|P_A P_B\|\right)}{\max\left(h_{P_B}, \|P_A P_B\|\right)}$$

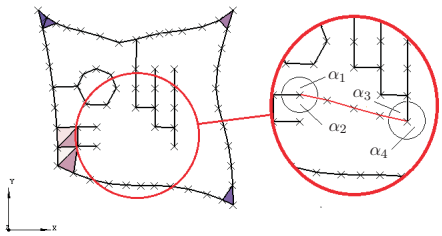
$$a = \min\left(a_1, a_2\right) > \begin{cases} 0.6 & \text{good quality} \\ 1.0E-6 & \text{bad quality} \end{cases}$$

(b)

Abbildung 5.13: Dreiecks-Abspaltung (*chop*)

Die *chop*-Prozedur kann dabei mit zwei unterschiedlichen Gütestufen aufgerufen werden. Bei einem ersten Aufruf werden nur Dreiecke abgespalten, bei denen das Verhältnis zwischen der neuen Dreieckskante und den h-Werten in den beiden dazu gehörigen Knoten (h_{P_A}, h_{P_b}) ein vorgeschriebenes empirisch ermitteltes Minimalverhältnis nicht unterschreitet (vgl. Abbildung 5.13(b)). Der zweite Versuch spaltet das bestmögliche gültige Dreiecke vom aktuellen Gebietsrand ab.

Die *divide*-Prozedur sucht nach einer Teilungskante durch Verbinden zweier nicht benachbarter Randknoten, die das aktuelle Vernetzungsgebiet in zwei Untergebiete unterteilt (vgl. Abbildung 5.14(a)). Die Teilungskante darf dabei den aktuellen Gebietsrand nicht schneiden oder berühren und muss mit den angrenzenden Gebietskanten ein Mindestkriterium für die Winkel zwischen Teilungskante und den benachbarter Randkante erfüllen (vgl. Abbildung 5.14(b)). Entlang dieser Teilungskante werden wie im nachfolgenden Abschnitt erläutert neue Knoten platziert.



(a)

$$\alpha_1 > \frac{\pi}{4}$$

$$\alpha_2 > \frac{\pi}{4}$$

$$\alpha_3 > \frac{\pi}{4}$$

$$\alpha_4 > \frac{\pi}{4}$$

(b)

Abbildung 5.14: Gebietsteilung (*divide*)

Durch eine endliche Wiederholung dieser beiden Prozeduren nach dem rekursiven Gebietsteilungsschema (vgl. Abbildung 5.12) wird das Vernetzungsgebiet in ein reines Dreiecksnetz unterteilt (vgl. Abbildung 5.15).

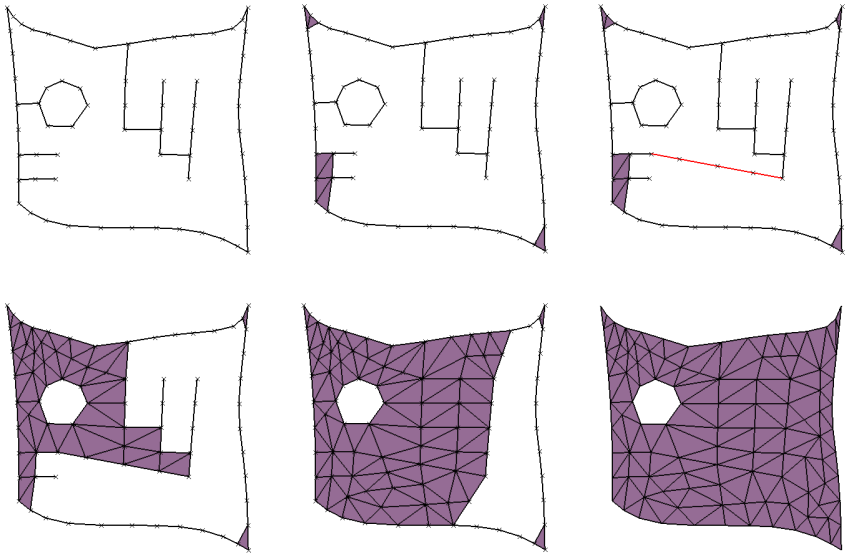


Abbildung 5.15: Rekursive Gebietsteilung einer Strukturfläche

Längen- und Winkelberechnung von Flächenkurven

Die an der rekursiven Gebietsteilung beteiligten geometrischen Operationen basieren auf ebenen Vernetzungsgebieten. Eine Vernetzung von beliebigen Raumflächen erfolgt daher durch Ansetzen der Parameterdarstellung der zugrunde liegenden geometrischen Flächen. Bei gekrümmten Raumflächen gehen durch eine Transformation der Flächenparameter (u, v) in euklidische Koordinaten (x, y, z) jedoch die tatsächlichen Längen- und Winkelverhältnisse verloren. Dieses Problem tritt bereits in 2D bei Parameterfunktionen mit stark variierenden Ableitungen auf. Der Klarheit der Darstellung wegen wurde zur Veranschaulichung der Methoden zur Längen- und Winkelberechnung ein 2D-Beispiel mit einer stark variierenden Metrik verwendet (vgl. Abbildung 5.16).

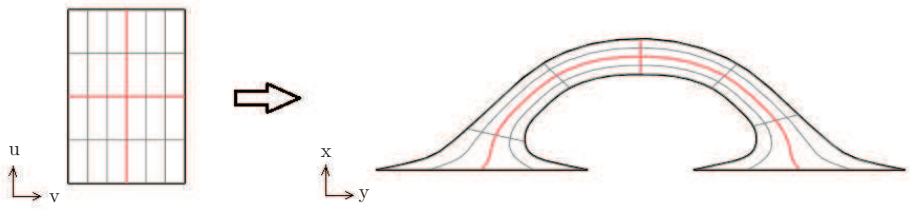
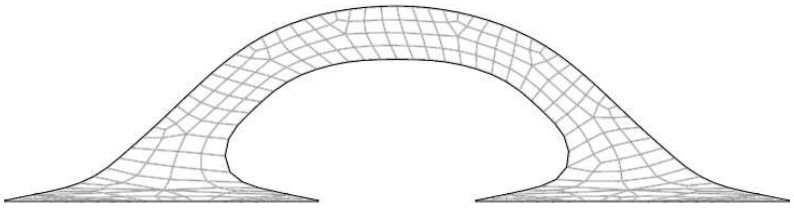


Abbildung 5.16: Freiformfläche im euklidischen Raum und in der Parameterebene

In der ursprünglichen Version des Vernetzungsalgorithmus erfolgt eine Berechnung der räumlichen Abstände und Winkel über eine skalierte Parameterebene $(\hat{u} = u \cdot f_u, \hat{v} = v \cdot f_v)$. Die Skalierungsfaktoren f_u und f_v werden über das Längenverhältnis der zwei Mittel-Isolinien (rot) zu ihren tatsächlichen Längen im euklidischen Raum festgelegt. Abbildung 5.17(a)) zeigt die skalierte Para-

meterebene samt Transformation in Flächenparameter für das 2D-Beispiel aus Abbildung 5.16. Aufgrund der starken Abweichung zwischen den skalierten und euklidischen Winkeln und Abständen resultieren auf diese Weise jedoch stark verzerrte oder sogar unbrauchbare Elemente (vgl. Abbildung 5.17(b)).

(a) Skalierung der Parameterebene



(b) Viereckshetz

Abbildung 5.17: Längen- und Winkelberechnung auf skalierter Parameterebene

Im Rahmen dieser Arbeit wurde die Berechnung der räumlichen Abstände und Winkel über die differentialgeometrischen Eigenschaften der Fläche (vgl. Kapitel 2.4.3) implementiert. Die exakten Winkel und Längenverhältnisse lassen sich so unter Einbeziehung der lokalen Ableitungen durch eine Taylorreihe nähern. Auf diese Weise werden qualitativ hochwertige Netze auf stark gekrümmten Vernetzungsgebieten generiert.

Länge einer Flächenkurve

Eine Flächenkurve C auf einer geometrischen Fläche S ist definiert durch einen Startpunkt P_A, einen Endpunkt P_B und einer Geraden innerhalb der Parameterebene (u, v) (vgl. Abbildung 5.18).

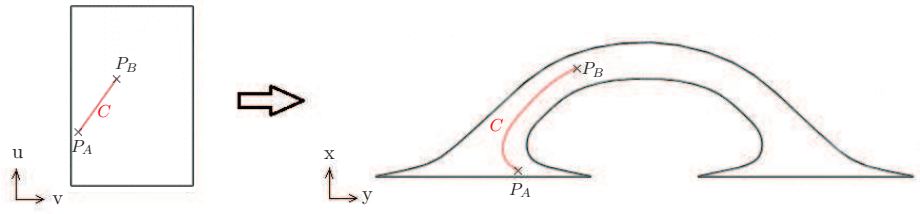


Abbildung 5.18: Länge einer Flächenkurve

Die approximierte Länge der Flächenkurve berechnet sich in der ursprünglichen Version des Algorithmus direkt aus den skalierten Parametern des Startpunktes $(\hat{u}_A, \hat{v}_A)$ und des Endpunktes $(\hat{u}_B, \hat{v}_B)$ nach

$$\|P_A P_B\| = \sqrt{(\hat{u}_B - \hat{u}_A)^2 + (\hat{v}_B - \hat{v}_A)^2} \tag{5.1}$$

Wird die Jakobimatrix (2.68) der parametrischen Flächenfunktion $f(u, v)$ punktweise in einem Flächenpunkt (u_P, v_P) ausgewertet, so lässt sich nach 2.71 der Metrik Tensor $\mathbf{M}_P$ wie folgt ermitteln

$$\tau_{1,p} = \begin{pmatrix} \frac{\partial f_x(u_P, v_P)}{\partial u} \\ \frac{\partial f_y(u_P, v_P)}{\partial u} \\ \frac{\partial f_z(u_P, v_P)}{\partial u} \end{pmatrix}; \qquad \tau_{2,p} = \begin{pmatrix} \frac{\partial f_x(u_P, v_P)}{\partial v} \\ \frac{\partial f_y(u_P, v_P)}{\partial v} \\ \frac{\partial f_z(u_P, v_P)}{\partial v} \end{pmatrix} \tag{5.2}$$

$$E_P = \tau_{1,p} \cdot \tau_{1,p} \tag{5.3a}$$
$$F_P = \tau_{1,p} \cdot \tau_{2,p} \tag{5.3b}$$
$$G_P = \tau_{2,p} \cdot \tau_{2,p} \tag{5.3c}$$

$$\mathbf{M}_P = \begin{bmatrix} E_P & F_P \\ F_P & G_P \end{bmatrix} \tag{5.4}$$

Die Länge der Flächenkurve von Punkt P_A nach Punkt P_B lässt sich mit Hilfe des Metrik-Tensors im Punkt P_A über eine nach dem ersten Glied abgebrochene Taylorreihe der metrischen Bogenlänge (2.74) annähern nach

$$\|P_A P_B\|_{M_A} \approx \sqrt{E_A(u_B - u_A)^2 + 2F_A(u_B - u_A)(v_B - v_A) + G_A(v_B - v_A)^2} \tag{5.5}$$

Die Berechnung berücksichtigt hierbei die erste Ableitung im Flächenpunkt P_A und liefert eine gute Näherung für kurze Flächenkurven und Flächenkurven mit konstanter Ableitung. Eine verbesserte Näherung erfolgt über die Mittelung der beiden Längenberechnungen mit Hilfe des Metrik-Tensors in Punkt P_A und in Punkt P_B.

$$\|P_A P_B\| \approx \frac{\|P_A P_B\|_{M_A} + \|P_A P_B\|_{M_B}}{2} \tag{5.6}$$

Winkel zwischen zwei Flächenkurven

Zwei Flächenkurven C_1 und C_2 haben einen gemeinsamen Schnittpunkt P, wenn sich die beiden Geraden in der Parameterebene (u, v) schneiden (vgl. Abbildung 5.19).

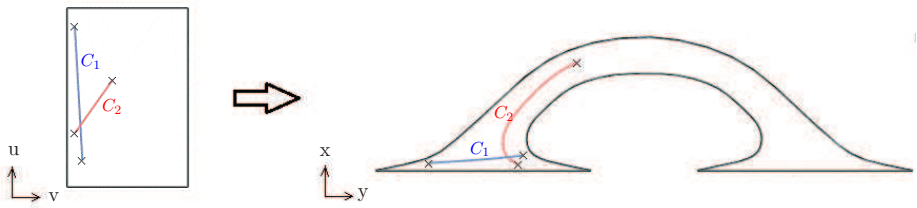


Abbildung 5.19: Winkel zwischen zwei Flächenkurven

In der ursprünglichen Version des Algorithmus berechnet sich der Winkel zwischen den Flächenkurven im Punkt P auf der skalierten Parameterebene näherungsweise nach

$$\tau_1 = \begin{pmatrix} \hat{u}_P - \hat{u}_A \\ \hat{v}_P - \hat{v}_A \end{pmatrix} \tag{5.7a}$$

$$\tau_2 = \begin{pmatrix} \hat{u}_P - \hat{u}_B \\ \hat{v}_P - \hat{v}_B \end{pmatrix} \tag{5.7b}$$

$$sin(\alpha) = \frac{\tau_1 \cdot \tau_2}{|\tau_1| \cdot |\tau_2|} \tag{5.8}$$

Eine Berechnung der exakten Winkel im euklidischen Raum in einem Flächenpunkt P erfolgt über die Richtungsableitungen der parametrischen Flächenfunktion $f(u, v)$ nach

$$\tau_1 = \begin{pmatrix} \frac{\partial f_x(u_P,v_P)}{\partial u}(u_P - u_A) + \frac{\partial f_x(u_P,v_P)}{\partial v}(v_P - v_A) \\ \frac{\partial f_y(u_P,v_P)}{\partial u}(u_P - u_A) + \frac{\partial f_y(u_P,v_P)}{\partial v}(v_P - v_A) \\ \frac{\partial f_z(u_P,v_P)}{\partial u}(u_P - u_A) + \frac{\partial f_z(u_P,v_P)}{\partial v}(v_P - v_A) \end{pmatrix} \tag{5.9a}$$

$$\tau_2 = \begin{pmatrix} \frac{\partial f_x(u_P,v_P)}{\partial u}(u_P - u_B) + \frac{\partial f_x(u_P,v_P)}{\partial v}(v_P - v_B) \\ \frac{\partial f_y(u_P,v_P)}{\partial u}(u_P - u_B) + \frac{\partial f_y(u_P,v_P)}{\partial v}(v_P - v_B) \\ \frac{\partial f_z(u_P,v_P)}{\partial u}(u_P - u_B) + \frac{\partial f_z(u_P,v_P)}{\partial v}(v_P - v_B) \end{pmatrix} \tag{5.9b}$$

$$sin(\alpha) = \frac{\tau_1 \cdot \tau_2}{|\tau_1| \cdot |\tau_2|} \tag{5.10}$$

Abbildung 5.20 illustriert den Einfluss der Längen- und Winkelberechnung unter Einbeziehung des Metrik Tensors auf die Vernetzung anhand der Beispielgeometrie aus Abbildung 5.16.

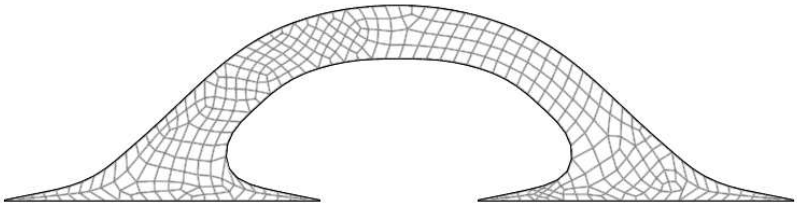


Abbildung 5.20: Viereckснetz auf Basis der lokalen Metrik

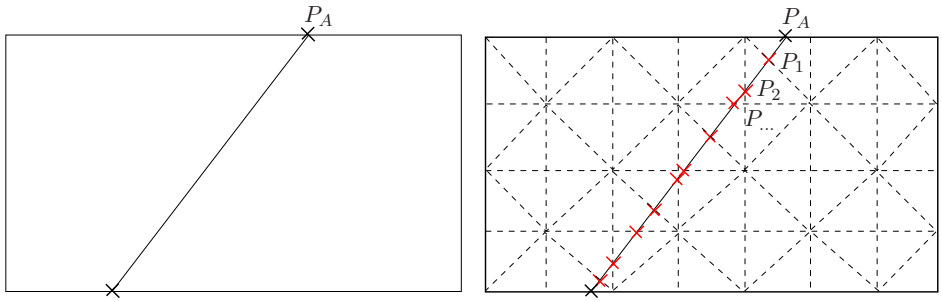
Platzierung der inneren Knoten

Die Platzierung der inneren Knoten entlang einer Randkante oder einer Flächenkante erfolgt grundsätzlich auf zwei unterschiedliche Weisen.
In einem einfachen Lauf (vgl. Abschnitt 5.2.2) werden die neuen Knoten entsprechend der h-Werte in Start- und Endknoten der Kante $\|P_A P_B\|$ platziert (vgl. Abbildung 5.21(a)), die Berechnung der Kantenlänge erfolgt über die natürliche Parametrisierung bei Randkanten oder über den Metrik Tensor in Start- und Endknoten bei Flächenkanten.
Bei einem adaptiven Lauf berechnet sich die Anzahl und Position der neuen Knoten entlang einer Rand- oder Flächenkante abschnittsweise zwischen zwei benachbarten Schnittpunkten mit dem Hintergrundnetz (vgl. Abbildung 5.21(b)). Die h-Werte in den Schnittpunkten werden aus dem Hintergrundnetz interpoliert und die Länge der Kante wird über die Länge der einzelnen Abschnitte aufsummiert. Dadurch sind im Fall von stark gekrümmten Flächenkanten deutlich genauere Längenberechnungen auf Basis des lokalen Metrik Tensors möglich

Sind auf einer Kante $\|P_A P_B\|$ (einfacher Lauf) oder auf einem Abschnitt $\|P_i P_{i+1}\|$ (adaptiver Lauf) die h-Werte in beiden Knoten identisch, so berechnet sich die Anzahl der neuen Knoten (n_K) für diesen Abschnitt nach

$$n_K = \frac{\|P_i, P_{i+1}\|}{h_i} \tag{5.11}$$

Bei unterschiedlichen h-Werten in Start- und Endknoten ermittelt sich Anzahl der neuen Knoten mit Hilfe des vordefinierten *Propagationsfaktors* p (vgl. Abschnitt 5.2.2) nach



(a) Einfacher Lauf (b) Adaptiver Lauf

Abbildung 5.21: Platzierung der inneren Knoten

$$a_{min} = \min \begin{cases} h_i \\ \frac{h_{i+1}}{p} + (1 - \frac{1}{p}) \cdot \|P_i, P_{i+1}\| \\ \|P_i, P_{i+1}\| \end{cases} \tag{5.12a}$$

$$b_{min} = \min \begin{cases} h_{i+1} \\ \frac{h_i}{p} + (1 - \frac{1}{p}) \cdot \|P_i, P_{i+1}\| \\ \|P_i, P_{i+1}\| \end{cases} \tag{5.12b}$$

$$n_K = \frac{\log\left(\frac{b_{min}}{a_{min}}\right)}{\log\left(\frac{\|P_i,P_{i+1}\|-a_{min}}{\|P_i,P_{i+1}\|-b_{min}}\right)} \tag{5.13}$$

Bei einem einfachen Lauf wird die Anzahl der neuen Knoten abgerundet

$$n_{tot} = \mathrm{INT}\left(n_K - 0.5 + \epsilon\right) \tag{5.14}$$

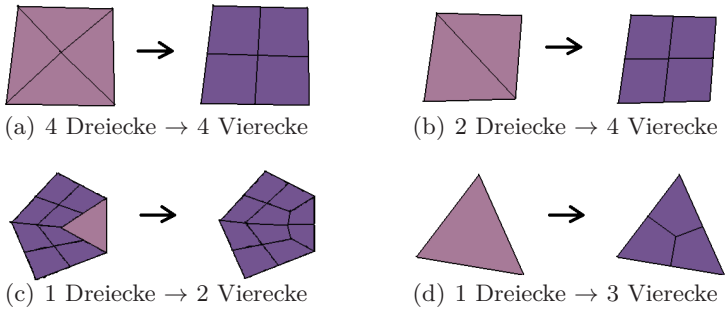
und die neuen Knoten linear oder adaptiv auf der Kante verteilt. Bei einem adaptiven Lauf wird die Anzahl der neuen Knoten aus der Summe der neuen Knoten entlang der Abschnitte ermittelt nach

$$n_{tot} = \mathrm{INT}\left(\sum_i \left(n_K^i\right) + 0.5 - \epsilon\right) \tag{5.15}$$

und die neuen Knoten abschnittsweise auf der Kante verteilt.

5.2.6 Konvertierung in ein Viereckснetz

Die Konvertierung des Dreiecksnetzes in ein Viereckснetz [83] erfolgt über vier Konvertierungsvarianten, die nach einander aufgerufen werden. Zunächst werden alle Inseln aus vier Dreiecken, die sich einen gemeinsamen Knoten teilen, in vier Vierecke konvertiert (vgl. Abbildung 5.22(a)). Anschließend werden alle Dreieckspaare, die in Elementen mit guten Innenwinkeln resultieren, in vier Vierecke konvertiert (vgl. Abbildung 5.22(b)). Die übrigen Dreiecke werden je nach Umgebung entweder in zwei (vgl. Abbildung 5.22(c)) oder in drei (vgl. Abbildung 5.22(d)) Vierecke konvertiert.



(a) 4 Dreiecke → 4 Vierecke (b) 2 Dreiecke → 4 Vierecke

(c) 1 Dreiecke → 2 Vierecke (d) 1 Dreiecke → 3 Vierecke

Abbildung 5.22: Konvertierungsvarianten

Die vier Grundvarianten basieren auf der Teilung der ursprünglichen Dreiecke, dadurch werden neue Knoten auf den Dreieckskanten erzeugt und die Elementdichte verdoppelt sich. Wird auf einem Vernetzungsgebiet ein Viereckснetz generiert, so wird zuvor ein Dreiecksnetz mit halber Elementdichte erstellt.
Abbildung 5.23 illustriert die einzelnen Konvertierungsschritte für das Beispiel aus Abbildung 5.15.

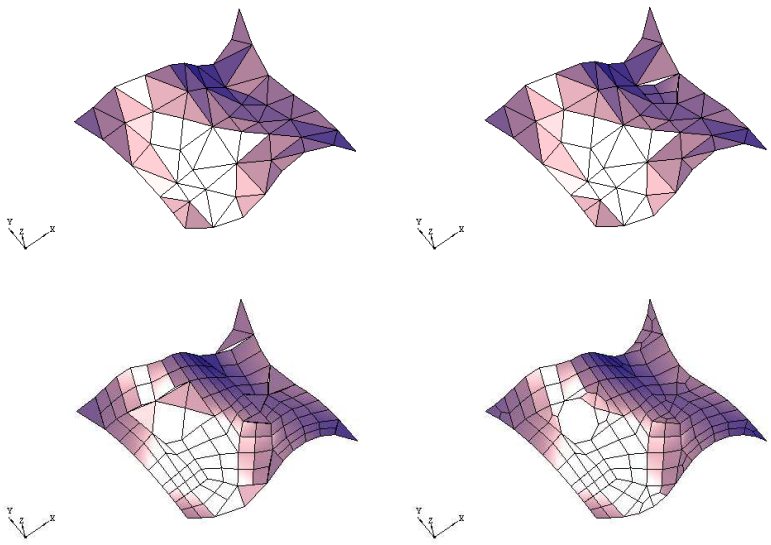
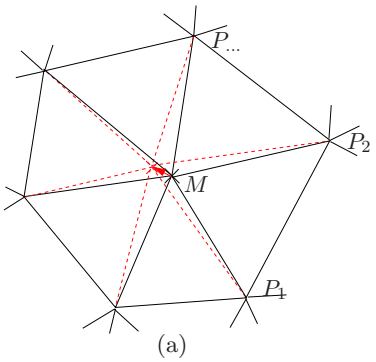


Abbildung 5.23: Konvertierung eines Dreiecksnetzes

5.2.7 Knotenrelaxation

In der abschließenden Knotenrelaxation werden alle verschieblichen Knoten im Inneren der Struktur in Richtung des Schwerpunkts der Teilfläche geschoben, die durch alle angrenzenden Dreiecks- oder Viereckselemente aufgestellt wird (vgl. Abbildung 5.24(a)). Um Übergänge von grob auf fein vernetzte Gebiete zu erlauben wird der Schwerpunkt mit den h-Werten in den Elementknoten skaliert (vgl. Abbildung 5.24(b)).



(a)

$$u_S = \sum \frac{u_i \cdot \|P_i, M\|}{h_i}$$

$$v_S = \sum \frac{v_i \cdot \|P_i, M\|}{h_i}$$

$$s = \sum \frac{\|P_i, M\|}{h_i}$$

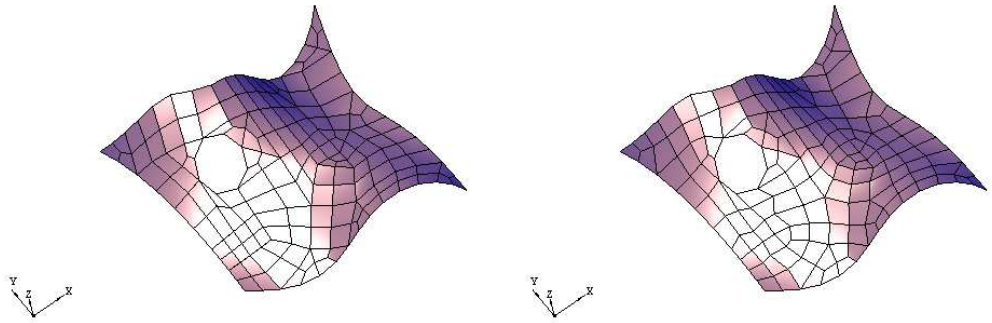
$$u_{new} = \frac{u_S}{s}$$

$$v_{new} = \frac{v_S}{s}$$

(b)

Abbildung 5.24: Knotenrelaxation

Die Relaxation ist ein iterativer Prozess und kann anhand der neuen Knoten-Koordinaten rekursiv wiederholt werden. Bei zu vielen Wiederholungen kann jedoch die Qualität der kleineren Element auf Kosten der großen Elemente sinken. Das empirisch bestimmte Optimum liegt bei drei Relaxationsschritten. Abbildung 5.25 zeigt den Einfluss der Knotenrelaxation für das Viereckssnetz aus Abbildung 5.23 mit drei Relaxationsschritten.



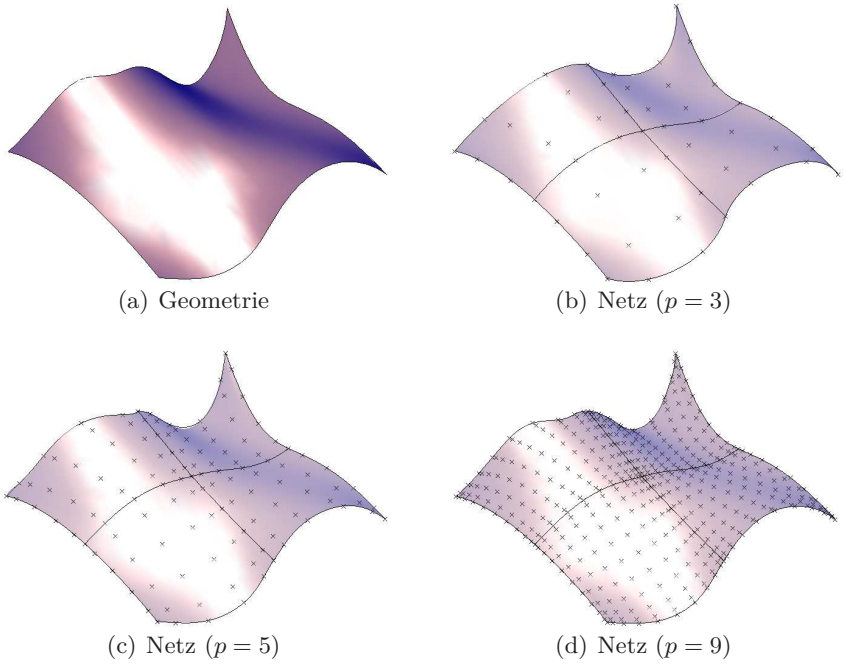
(a) Unrelaxiertes Netz (b) Netz nach drei Relaxationsschritten

Abbildung 5.25: Relaxierung eines Viereckssnetzes

5.3 Diskretisierung mit Elementen hoher Ordnung

In der p-Version der Finite-Elemente-Methode werden Ansatzfunktionen hoher Ordnung auf groben Netzen mit nur wenigen Elementen definiert (vgl. Kapitel 3.3.4). Die Elemente nehmen dabei für die Integration die exakte Geometrie

des zugrunde liegenden geometrischen Modells an. Eine Kopplung der Elemente mit der exakten Geometriebeschreibung im Geometriemodellierer führt im Fall von komplexen Freiformflächen zu teuren geometrischen Differentialoperationen. Deutlich schneller lassen sich gekrümmte Elemente numerisch integrieren, wenn jedes Element eine eigene Oberflächendiskretisierung besitzt. Hierfür werden auf der Oberfläche der Elemente Kontrollpunkte generiert, welche das Berechnungsgebiet durch eine Polynominterpolation näherungsweise beschreiben. Der Polynomgrad wird durch die Anzahl der inneren Kontrollpunkte definiert ($p = n_K+1$), welche wiederum durch das diskretisierte Interpolationspolynom interpoliert werden. Abbildung 5.26 zeigt die Diskretisierung von Viereckselementen hoher Ordnung auf einer stark gekrümmten Freiformfläche mit unterschiedlichem Polynomgrad p.

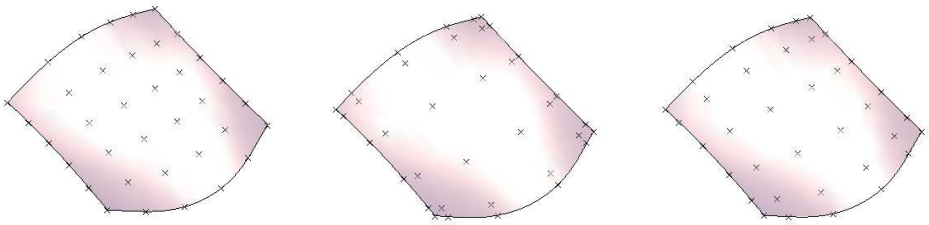


(a) Geometrie (b) Netz ($p = 3$)

(c) Netz ($p = 5$) (d) Netz ($p = 9$)

Abbildung 5.26: Diskretisierung von Viereckselementen hoher Ordnung

Die Verteilung der Kontrollpunkte hat einen entscheidenden Einfluss auf die Qualität der Näherung. Bei äquidistanter Verteilung der Kontrollpunkte (vgl. Abbildung 5.27(a)) wird das Berechnungsgebiet gleichmäßig gut aufgelöst. Eine optimierte Verteilung der Kontrollpunkte bezüglich der Integration erfolgt nach der Gaußschen Verteilung (vgl. Abbilung 5.27(b)). Bei Berechnungsgebieten mit lokalen Krümmungen oder Knicken können durch die Diskretisierung der Elementoberflächen jedoch Oszillationen entstehen. Eine optimale Verteilung der Kontrollpunkte auf dem Berechnungsgebiet bezüglich minimalen Oszillationen (vgl. Abbildung 5.27(c)) erfolgt durch die Babuška-Chen Punkte [12].



(a) Reguläre Verteilung (b) Gauss-Veteilung (c) Babuška-Chen Verteilung

Abbildung 5.27: Kontrollpunktverteilung

5.4 Element und Netzqualität

Die Form und Ausrichtung der einzelnen Elemente hat einen entscheidenden Einfluss auf die Qualität der Berechnungsergebnisse. Neben günstigen Seitenverhältnissen und möglichst regulären Innenwinkeln, ist aus mathematischer Sicht eine Ausrichtung der Elemente nach den Gradienten der physikalischen Größen vorteilhaft. Da ohne genaue Kenntnis des zu berechnenden physikalischen Problems viele der zahlreichen Einflussparameter auf die Qualität des Finite-Elemente Netzes nur schwer zu erfassen sind, wird die Netzqualität in der Praxis anhand rein geometrischer Kriterien bestimmt. Einen ersten Eindruck liefert in der Regel die Visualisierung, jedoch lassen sich bei großen Gebieten mit vielen Elementen auf diese Weise keine zuverlässigen Aussagen treffen.

Neben einer visuellen Kontrolle gibt es zahlreiche Größen die einen Hinweis auf die Qualität von Finite-Elemente Netzen liefern. So lassen sich beispielsweise die Innenwinkel oder Seitenverhältnisse der einzelnen Elemente ermitteln und statistisch auswerten. Aufgrund der unterschiedlichen Anforderungen an Netze für eine Berechnung mit der h- und der p-Version der Finite-Elemente-Methode lassen sich aus diesen Werten jedoch keine allgemeingültigen Aussagen zur Netzqualität treffen.

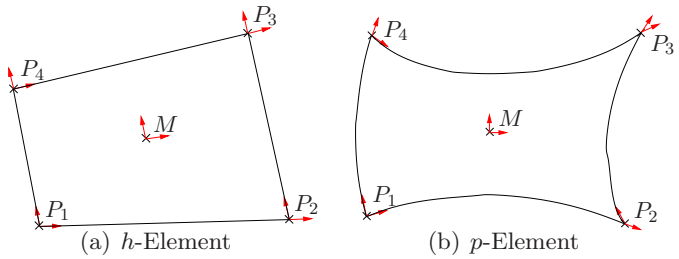
Eine weit verbereitete und beliebte Größe zur Bestimmung der Netzqualität, die sowohl für h als auch für p Elemente aussagekräftige Werte liefert, ist die skalierte Jakobi-Metrik. Dieser Abschnitt erklärt die Ermittlung der Netzqualität anhand der skalierten Jakobi Metrik und illustriert den Einfluss der in diesem Kapitel beschriebenen Vernetzungsparameter auf die Güte der generierten Netze.

5.4.1 Skalierte Jakobi Metrik

Die numerische Integration eines Elementes erfolgt auf der lokalen Parameterebene des Standardelementes (vgl. Kapitel 3.3). Die Berücksichtigung der tatsächlichen Geometrie erfolgt über die Jakobi-Matrix der Abbildungsfunktion Q_e, die in den beiden Ableitungsvektoren

$$\tau_1 = \begin{pmatrix} \frac{\partial N_x(u,v)}{\partial u} \\ \frac{\partial N_y(u,v)}{\partial u} \\ \frac{\partial N_z(u,v)}{\partial u} \end{pmatrix}; \qquad \tau_2 = \begin{pmatrix} \frac{\partial N_x(u,v)}{\partial v} \\ \frac{\partial N_y(u,v)}{\partial v} \\ \frac{\partial N_z(u,v)}{\partial v} \end{pmatrix} \tag{5.16}$$

resultiert. Der Winkel zwischen den beiden Ableitungsvektoren τ_1 und τ_2 entspricht dabei dem Winkel zwischen den beiden lokalen Parameterrichtungen **u** und **v** im euklidischen Raum (vgl. Abbildung 5.28).



(a) *h*-Element (b) *p*-Element

Abbildung 5.28: Jakobi Metrik

Aus den beiden Ableitungsvektoren lässt sich für jeden beliebigen Punkt P innerhalb des Standardelementes die skalierte Jakobi-Metrik J_P berechnen nach

$$J_P = \frac{\sqrt{(\tau_{1,p} \cdot \tau_{1,p}) \cdot (\tau_{2,p} \cdot \tau_{2,p}) - (\tau_{1,p} \cdot \tau_{2,p})^2}}{\sqrt{(\tau_{1,p} \cdot \tau_{1,p}) \cdot (\tau_{2,p} \cdot \tau_{2,p})}} \tag{5.17}$$

Die Qualität eines Elementes entspricht dem Minimum der skalierten Jakobi-Metrik innerhalb des Standardelementes, welches in der Regel in einem der Elementknoten P_i auftritt. Eine Ermittlung der Elementqualität eines einzelnen Elementes erfolgt nach

$$J_{ref} = \min\left(J_{P_i}\right) \tag{5.18}$$

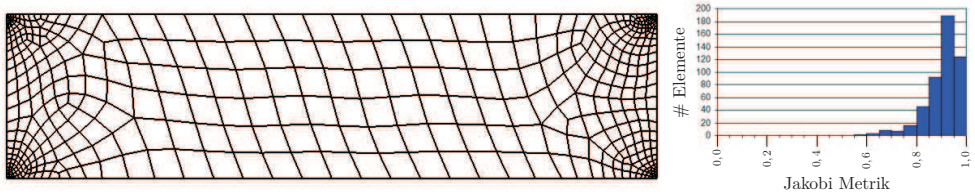
Eine Einschätzung der Qualität eines Netzes anhand der skalierten Jakobi-Metrik erfolgt nach folgender Einstufung:

- *1.0 - 0.8:* optimal
- *0.8 - 0.6:* gut
- *0.6 - 0.4:* ausreichend

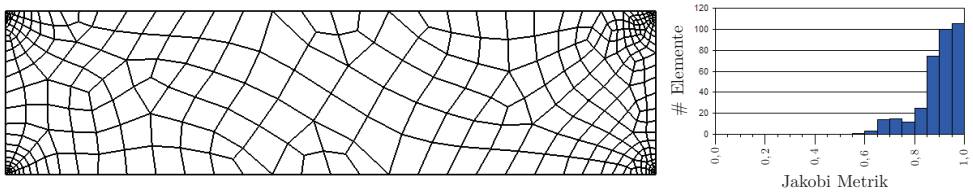
- *0.4 - 0.2:* schlecht
- *0.2 - 0.0:* sehr schlecht
- *< 0.0:* inakzeptable

5.4.2 Beispiele

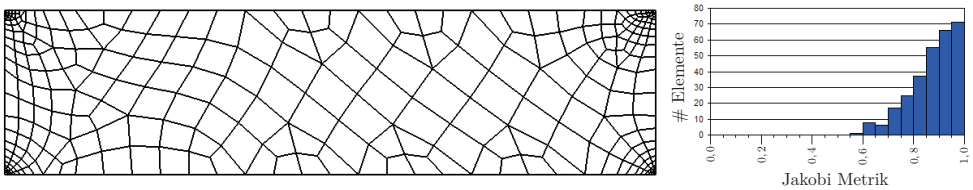
Das erste Beispiel demonstriert den Einfluss unterschiedlicher *Propagationsfaktoren* (vgl. Abschnitt 5.2.2) auf die Element- und Netzqualität. Das adaptiv vernetzte Gebiet ($h_{lokal} = 0,05; h_{global} = 1,00$) resultiert in allen vier Fällen in optimalen bis ausreichenden Elementen, wobei der Anteil der ausreichenden Elemente mit steigendem *Propagationsfaktor* zunimmt (vgl. Abbildung 5.29).



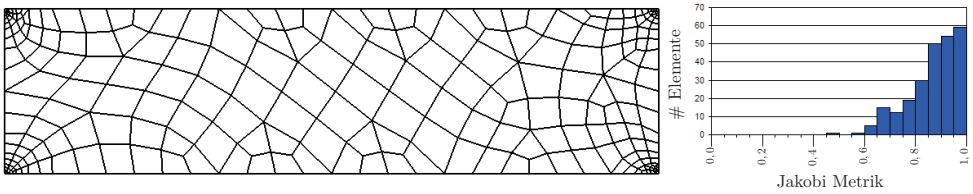
(a) $Propagationsfaktor = 1,4$



(b) $Propagationsfaktor = 1,5$



(c) $Propagationsfaktor = 1,6$

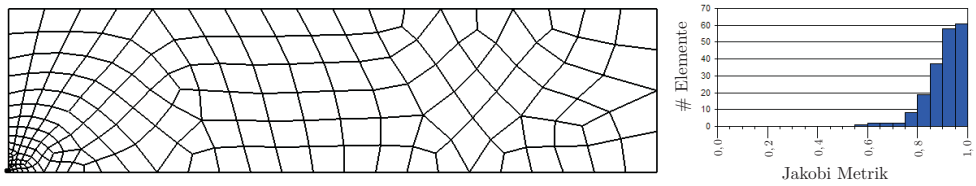


(d) $Propagationsfaktor = 1,7$

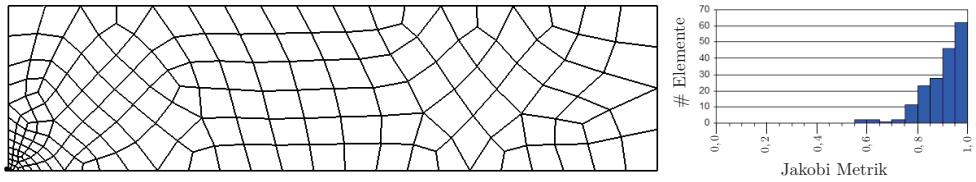
Abbildung 5.29: Beispiel 1: $Propagationsfaktor$

Das zweite Beispiel verdeutlicht den Einfluss des *Randkantenfaktors* (vgl. Abschnitt 5.2.2) auf einem Vernetzungsgebiet mit uniformen h-Wert ($h_{global} = 1,0$)

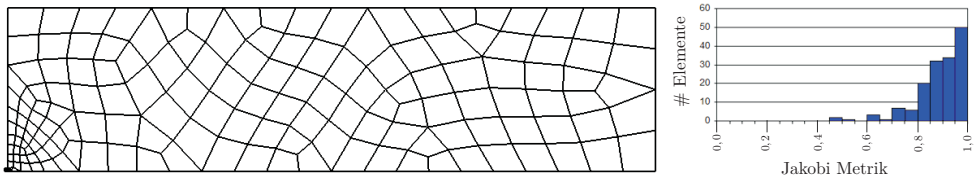
und im Verhältniss dazu kurzen Randkanten ($l_{min} = 0,05$). In den ersten drei Fällen entstehen bei einem *Randkantenfaktor* von $4,0$ und weniger Netze mit überwiegend sehr guten bis guten und wenigen ausreichenden Elementen. Eine Steigerung des *Randkantenfaktors* auf $10,0$ wie im vierten Beispiel der Fall resultiert nun auch in schlechten bis sehr schlechten Elementen im Bereich der Verfeinerung (vgl. Abbildung 5.30).



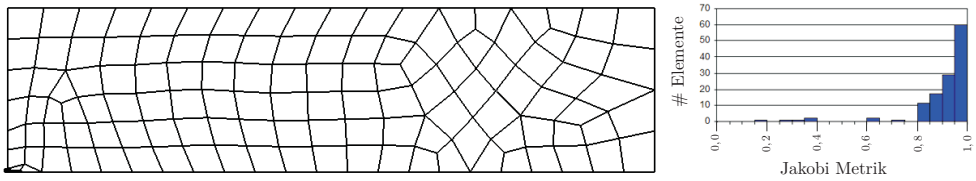
(a) $Randkantenfaktor = 1,4$



(b) $Randkantenfaktor = 2,0$



(c) $Randkantenfaktor = 4,0$

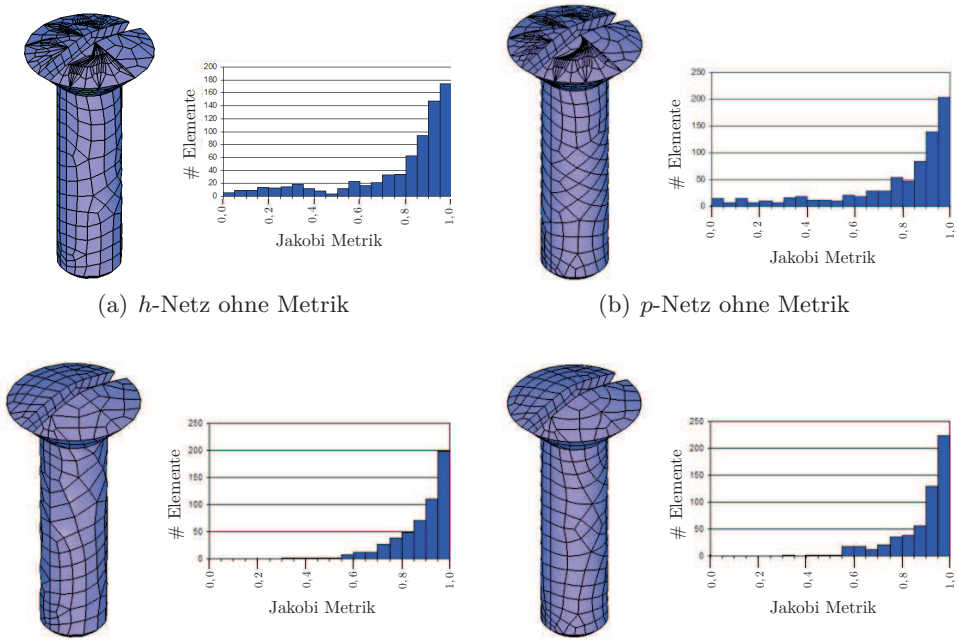


(d) $Randkantenfaktor = 10,0$

Abbildung 5.30: Beispiel 2: *Randkantenfaktor*

Das dritte Beispiel zeigt den Einfluss der Metrik-Vernetzung (vgl. Abschnitt 5.2.5) anhand des gekrümmten Oberflächenmodells einer Schraube. Das Modell wurde sowohl mit linearen Elementen als auch mit gekrümmten Elementen hoher Ordnung vernetzt und ausgewertet. Aufgrund der starken Verzerrungen zwischen Parameterebene und euklidischem Raum entstehen bei der Vernetzung mit der ursprünglichen Längen- und Winkelberechnung besonders im Bereich des Schraubenkopfs zahlreiche unbrauchbare Elemente. Die Vernetzung auf Basis der lokalen

Metrik hingegen resultiert in qualitativ deutlich hochwertigeren Netzen aus Elementen mit überwiegend ausreichender bis sehr guter Qualität.



(a) *h*-Netz ohne Metrik

(b) *p*-Netz ohne Metrik

(c) *h*-Netz mit Metrik

(d) *p*-Netz mit Metrik

Abbildung 5.31: Beispiel 3: Metrik Vernetzung

Kapitel 6

Generierung von Volumennetzen

Aufbauend auf den im vorangegangenen Kapitel gezeigten Techniken zur Generierung von Oberflächennetzen sollen nun die in das Rahmenwerk *TUM.GeoFrame* (vgl. Kapitel 4) implementierten Algorithmen zur Generierung von Tetraeder- und Hexaeder-Netzen für die h-Version (vgl. Kapitel 3.3.3) und die p-Version (vgl. Kapitel 3.3.4) der Finite-Elemente-Methode auf unterschiedlichen geometrischen Volumenmodellen (vgl. Kapitel 2.3) erläutert werden. Im speziellen wurden ein Algorithmus zur Generierung von gekrümmten Tetraeder-Elementen auf indirekten Volumenmodellen als Erweiterung des Tetraeder-Netzgenerators *Netgen* [90] und eine neue entwickelte Methode zur Generierung von gekrümmten Hexaeder-Elementen auf dünnwandigen Extrusionsmodellen entwickelt und implementiert. Die Leistungsfähigkeit der beiden Verfahren wird anhand einer Reihe von Beispielen illustriert.

6.1 Einführung

Die Generierung von Finite-Elemente Netzen zur Diskretisierung von dreidimensionalen Strukturen gliedert sich ähnlich wie die Generierung von Oberflächennetzen in Techniken zur Erstellung von strukturierten Gittern und Techniken zur Erstellung von unstrukturierten Netzen.

Bei rein strukturierten Vernetzungstechniken wird das Netz entweder durch komplexe iterative Glättungstechniken an den Rand der Struktur angepasst [87, 41, 77], oder ein umhüllendes blockstrukturiertes Gitter erzeugt, welches mit dem Rand der Struktur nicht zusammen fällt. Erstere Variante liefert zwar qualitativ hochwertige Finite-Elemente-Netze, lässt sich jedoch nur auf eine kleine Zahl von geometrischen Modellen anwenden. Die zweite Variante ist auf jedem beliebigen geometrischen Modell anwendbar, allerdings müssen die Randbedingungen unabhängig vom generierten Gitter beschrieben werden. Dies ist in der klassischen Finite-Elemente-Methode jedoch nicht möglich. Beispiele für numerische Berech-

nungsverfahren auf Basis von umhüllenden Gittern sind die *Lattice-Boltzmann-Methode* [34] in der numerischen Strömungsmechanik und die *Embedded Domain Methoden* [76, 86] in der Strukturmechanik.

Unstrukturierte Vernetzungstechniken können in einer Vielzahl von unterschiedlichen Elementtypen resultieren. Hierbei stellen einerseits die klassischen Tetraeder-Elemente aufgrund der verhältnissmäßig einfachen und allgemeingültigen Generierbarkeit und andererseits die Hexaeder-Elemente aufgrund ihrer mathematischen Vorteile bei numerischen Simulationsverfahren [80] die wichtigsten Elementtypen dar.

Obwohl der Übergang von zweidimenionalen in dreidimensionale Algorithmen mit einer deutlich gesteigerten Komplexität verbunden ist, lassen sich unstrukturierte Tetraeder-Netze über ähnliche Verfahren generieren wie unstrukturierte Dreiecksnetze. Die wichtigsten Grundtechniken sind dabei

- die Oktalbaum-Technik [114, 92], in der ein die geometrische Struktur umgebender Würfel rekursiv geteilt wird und Tetraeder-Elemente aus den unterteilten Zellen generiert werden,
- die Delaunay-Methode [17, 111], bei der analog zur Dreiecksvernetzung Tetraeder-Elemente anhand eines Umkugelkriteriums [37] auf einer zuvor generierten Menge an inneren Punkten definiert werden, und
- die Advancing-Front-Methode [89, 62], bei der die Tetraeder-Elemente ausgehend von einem triangulierten Oberflächennetz in Richtung Strukturinneres generiert werden.

In der Oktalbaum-Technik [88, 53] wird ein die Struktur umhüllender Würfel so lange rekursiv geteilt, bis eine gewünschte Auflösung erreicht ist. Die Tetraeder-Elemente werden sowohl auf den irregulären mit dem Rand verschnittenen Zellen, als auch auf den regulären inneren Zellen generiert. Durch Begrenzung der Differenz der Rekursionstiefe von benachbarten Zellen lassen sich glatte Übergänge von groben auf feine Netzbereiche erzeugen. Eine weitere Verbesserung der Elementqualität wird über zusätzliche Glättungs- und Säuberungstechniken erzielt [44]. Obwohl die Oktalbaum-Technik ohne ein vorab generiertes Oberflächennetz auskommt, machen die vielen benötigten Verschneidungsoperationen den Algorithmus bei komplexen Geometrien teuer. Zudem sind die Elemente in Randnähe oft stark verzerrt. Daher wird die Oktalbaumtechnik in der Praxis eher selten verwendet.

In der Delauney-Technik erfolgt die Generierung der inneren Knoten wie bei der Dreiecksvernetzung auf unterschiedliche Weise. Beispielsweise lassen sich die neuen Knoten vom Rand ausgehend in Richtung Strukturinneres erzeugen [65], oder entlang Kanten definieren, die durch die Struktur gelegt werden [28], oder durch die iterative Verfeinerung eines groben Tetraeder-Netzes bestimmen [46], welches aus der Unterteilung der Ausgangsgeometrie in konvexe Teilstrukturen

entsteht [74]. Das Delauney-Kriterium für dreidimensionale Problemstellungen [37] besagt nun, dass kein Knoten innerhalb der Umkugel liegen darf, die durch die vier Knoten eines nicht angrenzenden Tetraeder-Elementes definiert ist.

Die Advancing-Front-Methode zur Generierung von Tetraeder-Elementen [89, 62] beruht auf einem vorab generierten Oberflächennetz aus Dreieckselementen. Die Tetraeder-Elemente werden ausgehend vom aktuellen Strukturrand in Richtung Strukturinneres abgespalten. Die ideale Position des vierten Knotens wird anhand der entsprechenden Dreiecksfläche des aktuellen Strukturrandes ermittelt. Zudem werden als möglicher vierter Knoten auch die Knoten eines gegenüber liegenden Strukturrandes berücksichtigt. Der Algorithmus generiert so entweder einen neuen Knoten, oder er verbindet durch Einfügen eines neuen Tetraeders den aktuellen Gebietsrand mit einem gegenüber liegenden Gebietsrand. Neben einer Kalibrierungs-Funktion zur Bestimmung der Elementgröße [39] benötigt die Advancing-Front-Methode teure Verschneidungsfunktionen, um sicher zu stellen, dass ein Tetraeder nicht über den Gebietsrand hinaus geht.

Die Generierung von unstrukturierten Hexaeder-Netzen gilt als die Königsdiziplin der Netzgenerierung und birgt nach wie vor viele ungelöste Fragen und Problemstellungen [24, 93]. Die Vernetzung in Hexaeder-Elemente gliedert sich in

- Abbildungstechniken, bei denen unstrukturierte Vierecks-Netze auf den Mittelflächen eines Volumenmodells erzeugt werden und in Hexaeder-Netze extrudiert werden,
- indirekte Techniken, bei denen zunächst ein einfach zu generierendes Tetraeder-Netz erzeugt wird, welches anschließend in ein Hexaeder-Netz konvertiert wird, und
- direkte Techniken, bei denen die Hexaeder-Elemente direkt auf dem geometrischen Modell ohne vorheriges Vernetzen der Ausgangsgeometrie platziert werden.

Die Abbildungstechnik erzeugt das Hexaeder-Netz durch die Extrusion eines Viereckснetzes, welches auf der Mittelfläche eines dreidimensionalen Volumenmodells generiert wird. Der erste Schritt beinhaltet üblicherweise die Dimensionsreduzierung des Volumenmodells in ein zweidimensonales Flächenmodell [94, 10, 57, 11]. Der Extrusionsalgorithmus verbindet anschließend topologisch getrennte Mittelflächen durch Verknüpfung der inneren Knoten [23, 56, 97, 66], so dass ein konformes Hexaeder-Netz entsteht. Obwohl die Abbildungstechnik sehr stabil ist und im Allgemeinen sehr gute Ergebnisse liefert, liegen die Hauptschwierigkeiten bei diesem Verfahren in der Dimensionsreduzierung des Volumenmodells, was nicht für jede beliebige Geometrie möglich ist.

Indirekte Techniken generieren das Hexaeder-Netz durch die Konvertierung eines Tetraeder-Netzes. Die einfachste Form der indirekten Hexaeder-Vernetzung ist die Zerlegung eines Tetraeder-Elementes in jeweils vier Hexaeder-Elemente.

Diese Technik resultiert jedoch in einer sehr schlechten Element- und Netzqualität. Daher wird dieses Verfahren als Basis für numerische Berechnungen im Allgemeinen abgelehnt. Ein vollständiger Algorithmus zur Generierung von reinen Hexaeder-Netzen durch Zusammenfassen von einzelnen Tetraeder-Elementen, ist in der Literatur bisher noch nicht dargelegt worden. Es existieren zwar Schemen, die durch zusammenfassen Netze aus überwiegend Hexaeder-Elementen erzeugen [73]. In den seltensten Fällen entstehen dabei jedoch reine Hexaeder-Netze. Ein indirektes Schema, welches ein reines Hexaeder-Netz durch Zusammenfassen von Tetraeder-Elementen generiert, müsste jeweils fünf oder mehr Tetraeder-Elemente zu einem Hexaeder-Element vereinen. Dieses Problem wurde bis heute noch von keiner Methode angemessen gelöst.

Die Generierung von unstrukturierten Hexaeder-Netzen mit direkten Methoden wird in der Literatur von drei Grundtechniken dominiert,

- den gitterbasierten-Techniken, welche ein strukturiertes Gitter im Inneren der Struktur generieren und dieses mit Übergangselementen an den Rand anbinden,
- den Pflasterungstechniken, welche Hexaeder-Elemente durch rekursives „Pflastern“ von neuen Elementreihen erzeugen, und
- den Web-Techniken, bei denen zunächst die Topologie des Hexaeder-Netzes innerhalb der Struktur generiert wird und anschließend die Hexaeder-Elemente eingepasst werden.

Bei der gitterbasierten Technik [115, 87, 41] wird im Inneren der Struktur zunächst ein angepasstes strukturiertes Gitter erzeugt. Anschließend werden zwischen dem strukturiertem Gitter und dem Strukturrand Hexaeder-Elemente eingefügt, um die Lücke entsprechend auszufüllen. Obwohl diese Methode sehr robuste Ergebnisse liefert, werden auf diese Weise speziell in Randnähe sehr verzerrte Elemente generiert. Zudem hängt das generierte Netz stark von der Ausrichtung des strukturierten inneren Gitters ab. Darüber hinaus sind Netze mit unterschiedlicher Elementgröße auf diese Weise nur sehr begrenzt möglich.

Bei der Pflasterungstechnik [26, 25, 98, 99] werden Hexaeder-Elemente reihenweise vom Rand ausgehend in Richtung Gebietsinneres durch das „Pflastern“ von neuen Elementreihen generiert. Eine Reihe von Prozeduren bestimmt die Reihenfolge, in der neue Elementformationen gesetzt werden. Ähnlich zur Advancing-Front-Methode wird nach Einfügen einer neuen Elementreihe der Strukturrand aktualisiert. Neben dem Ermitteln der Reihenfolge der Elementreihen muss bei der Pflasterungsmethode der Anschluß an bereits existierende Knoten ermittelt werden, sowie verschnittene Flächen erkannt werden. Im Laufe des Algorithmus können komplexe innere Hohlräume entstehen, welche in manchen Fällen nicht mit reinen Hexaeder-Elementen gefüllt werden können. Um das Generieren von Hexaeder-Elementen im Inneren der Struktur zu ermöglichen, müssen daher in manchen Fällen die bereits gesetzten Elementreihen modifiziert oder wieder entfernt werden. Obwohl bei jeder Ausgangsgeometrie zunächst eine große Zahl von

Elementreihen erfolgreich „gepflastert“ werden kann, ist eine vollständige Auflösung der Struktur nur in den seltensten Fällen möglich.

Die Web-Technik [104, 67, 68] basiert auf der Idee, zunächst nur die Topologie des Hexaeder-Netzes zu erzeugen und anschließend die Elemente geometrisch einzupassen. Die Topologie des Netzes wird aus sich schneidenden Flächen beschrieben. Die einzelnen Flächen werden durch die Web-Technik zunächst nur topologisch ermittelt, so dass keine teueren Verschneidungsoperationen benötigt werden. Erst anschließend wird das Topologie-Modell an die Oberfläche der Struktur angepasst und Hexaeder-Elemente eingefügt. Dabei entsteht in jedem Schnittpunkt von drei topologischen Flächen ein Hexaeder-Element. Obwohl die Web-Technik für viele Beispiele erfolgreiche Lösungen liefert, ist auch diese Methode nicht auf allen geometrischen Szenarien anwendbar.

Neben den einzelnen Grundtechniken gibt es zahlreiche Methoden zur Konvertierung von Hexaeder-Netzen. Ziel dieser Methoden ist die Konvertierung von nicht konformen Element-Oberflächen in konforme Element-Oberflächen. Auf diese Weise lassen sich komplexe dreidimensionale Strukturen in einfachere Unterstrukturen zerlegen und einzeln mit der jeweils bestmöglichen Methode vernetzen. Die Teilnetze werden anschließend derart konvertiert, dass die Elemente auf den gemeinsamen Oberflächen zusammenfallen.

6.2 Tetraeder-Vernetzung auf indirekten Volumenmodellen

Im nachfolgenden Abschnitt wird der im Rahmen dieser Arbeit entwickelte Algorithmus zur Generierung von Tetraeder-Elementen für die Finite-Elemente-Methode hoher Ordnung (vgl. Abschnitt 3.3.4) dargelegt. Der Algorithmus basiert auf den in Kapitel (5.2) vorgestellten Methoden zur Generierung von Dreicksnetzen und der frei verfügbaren *Netgen*-Bibliothek [90] zur Generierung von Tetraeder-Elementen. Die Kopplung dieser beiden Prozeduren unter Berücksichtigung der topologischen Aspekte des geometischen Modells (vgl. Kapitel 2.5) und die Diskretisierung der gekrümmten Tetraeder-Oberflächen ist Teil des in das Rahmenwerk *TUM.GeoFrame* (vgl. Kapitel 4) implementierten Netz-Generators *MeshGen*.

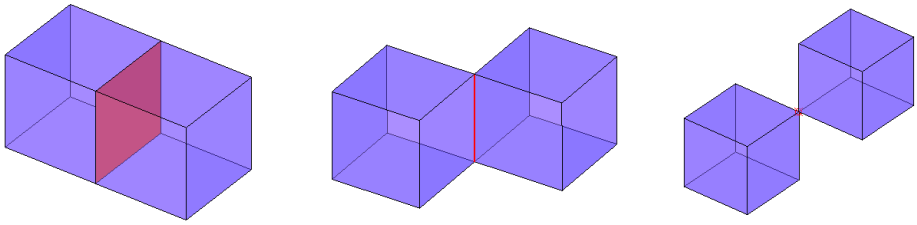
6.2.1 Topologisches Modell

Das topologische Modell der zu vernetzenden Struktur basiert auf dem Randdarstellungsschema (vgl. Kapitel 2.3.2) unter Einbeziehung des erweiterten Oberflächenmodells für Regionen aus Kapitel 5.2.1. Die Struktur besteht dabei aus einem oder mehreren Volumenkörpern, die durch eine geschlossene äußere Mantelfläche und optionalen inneren Mantelflächen (Hohlräume), bestehend aus einzelnen Regionen, begrenzt werden (vgl. Abbildung 6.1).

Besteht ein Modell aus mehreren Volumenkörpern, so muss die Konnektivität zwischen benachbarten Volumenkörpern durch gemeinsame topologische Entitä-

Abbildung 6.1: Topologisches Volumenmodell

ten wie z.B. gemeinsame Regionen (vgl. Abbildung 6.2(a)), gemeinsame Kanten (vgl. Abbildung 6.2(b)) oder gemeinsame Knoten (vgl. Abbildung 6.2(c)) definiert sein.



(a) Gemeinsame Region (b) Gemeinsame Kante (c) Gemeinsamer Knoten

Abbildung 6.2: Konnektivität von indirekten Volumenkörpern

6.2.2 *Netgen*-Bibliothek

Netgen [90] ist ein frei verfügbarer Tetraeder Netzgenerator, basierend auf einer Advancing-Front-Methode mit abstrakten Regeln zur Berechnung der inneren Knoten. *Netgen* kann als alleinstehende Anwendung kompiliert werden und als externe Bibliothek in andere Programme eingebunden werden. Die *Netgen*-Bibliothek verfügt dabei über zwei Schnittstellen. Die erste Schnittstellenprozedur generiert aus einem konstruktiven Festkörpermodell (CSG-Modell) oder einem Facettenmodell (vgl. Kapitel 2.3) ein Oberflächennetz aus Dreiecken. Die zweite Schnittstellenprozedur generiert aus einem geschlossenen Dreiecksnetz, welches einen beliebigen Volumenkörper berandet, die inneren Tetraeder-Elemente (vgl. Abbildung 6.3).

Um eine universale und leistungsfähige Anbindung von *Netgen* zu gewährleisten, wird nur die zweite *Netgen*-Prozedur zur Generierung der Tetraeder-Elemente aufgerufen. Auf diese Weise kann die Oberfläche des Tetraeder-Netzes anhand der internen geometrischen Datenbasis direkt mit den in Kapitel 5 gezeigten Vernetzungstechniken generiert werden und das geometrische Modell muss nicht über aufwändige Konvertierungsalgorithmen in eines der *Netgen*-kompatiblen Geometrieformate überführt werden. Das Dreiecksnetz, welches den zu beschreibenden Volumenkörper zunächst berandet, muss für eine Vernetzung mit *Netgen* geschlossen sein, darf keine Klaffungen oder Überlappungen aufweisen und die

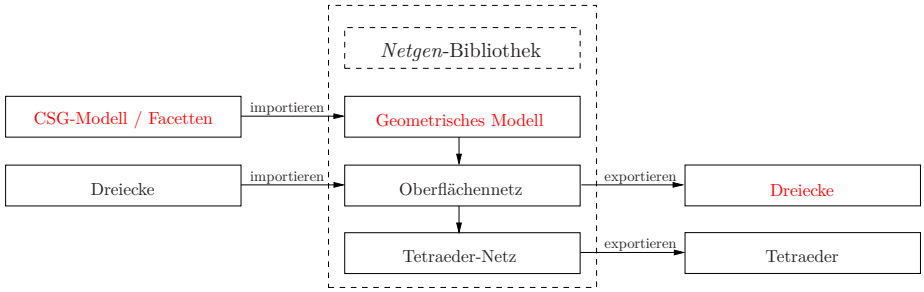
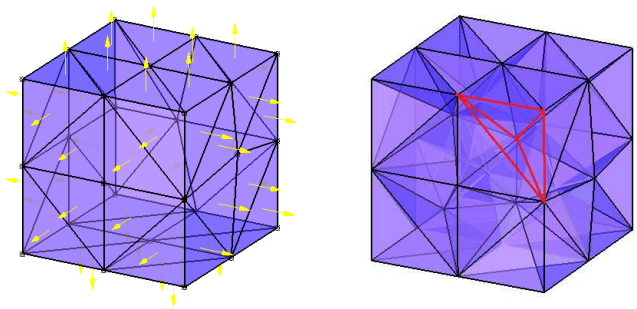


Abbildung 6.3: Schnittstellen der *Netgen* Bibliothek

Dreieckselemente müssen so orientiert sein, dass die Flächennormale der einzelnen Elemente nach außen zeigt (vgl. Abbildung 6.4).

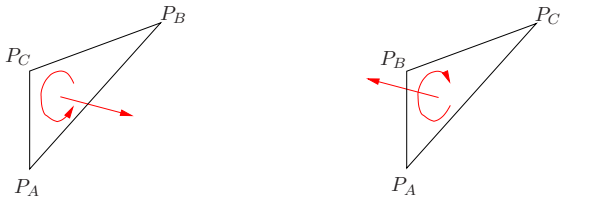


(a) Gerichtetes Dreiecksnetz (b) Tetraeder-Netz

Abbildung 6.4: Orientierung des Oberflächennetzes

6.2.3 Orientierung der Flächennormalen

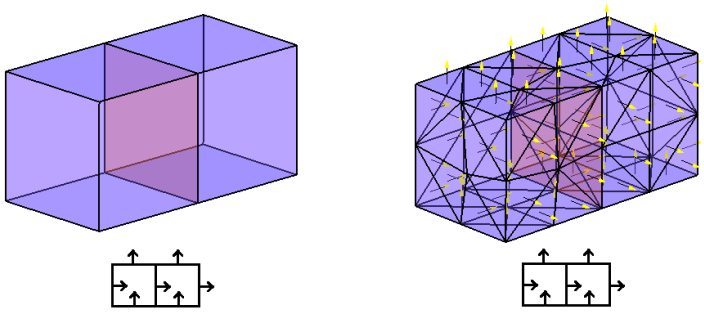
Die Orientierung der Flächennormalen eines einzelnen Dreieckselementes ist definiert durch den Umlaufsinn der drei Knoten. Die Flächennormale kann mit Hilfe der *Rechten-Faust-Regel* bestimmt werden, d.h. zeigen die gekrümmten Finger der rechten Hand in Richtung des Umlaufsinns eines Dreiecks, so zeigt der Daumen in Richtung der Flächennormalen. Durch Umdrehen der Reihenfolge der drei Knoten lässt sich die Flächennormale des Dreieckselementes einfach invertieren (vgl. Abbildung 6.5).



(a) Uhrzeigersinnn (b) Gegen-Uhrzeigersinnn

Abbildung 6.5: Orientierung eines Dreieckselements

Die anfängliche Ausrichtung der Flächennormalen der Dreieckselemente nach Abschluss der automatischen Dreiecksvernetzung (vgl. Kapitel 5.2.5) entspricht der Ausrichtung der Flächennormalen der einzelnen Regionen. Für eine Vernetzung mit *Netgen* müssen die Flächennormalen bezogen auf den Volumenkörper jedoch nach außen zeigen. Modelle, die aus mehreren Volumenkörpern zusammen gesetzt sind, besitzen dabei Regionen, zu denen keine eindeutige Aussage bezüglich der Ausrichtung der Flächennormalen nach innen bzw. nach außen getroffen werden kann (vgl. Abbildung 6.7).

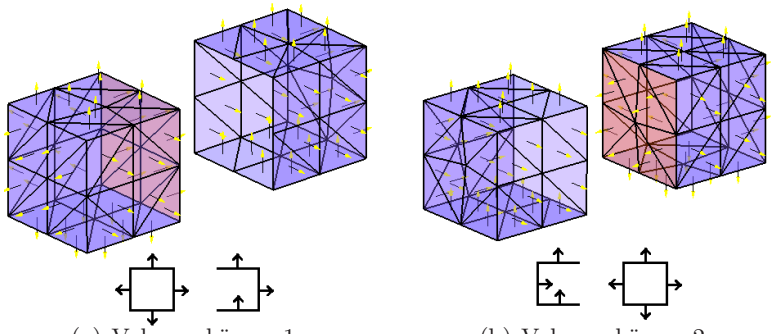


(a) Geometrisches Modell (b) Oberflächennetz

Abbildung 6.6: Orientierung des Dreiecksnetzes

Bei der Vernetzung von geometrischen Modellen aus mehreren Volumenkörpern wird zunächst ein Dreiecksnetz auf allen geometrischen Regionen generiert. Anschließend werden die Volumenkörper einzeln mit Tetraeder-Elementen vernetzt. Auf diese Weise lassen sich die Oberflächenelemente für jeden Volumenkörper einzeln nach außen ausrichten und an *Netgen* übergeben.

Die Ermittlung der anfänglichen Ausrichtung der Elemente auf einem Volumenkörper erfolgt durch das Aufstellen der Flächen-Kanten Matrix des Volumenkörpers und der Berechnung des dazu gehörigen Nullraums (vgl. Kapitel 2.5). Zeigen die Flächennormalen einer Region auf einem Volumenkörper nach innen, werden die Knoten aller Elemente auf dieser Region in umgekehrter Reihenfolge an *Netgen* übergeben (vgl. Abbildung 6.7).



(a) Volumenkörper 1 (b) Volumenkörper 2

Abbildung 6.7: Ausrichtung der Elemente

6.2.4 Diskretisierung der Tetraeder-Oberflächen

Bei der p-Version der Finite-Elemente-Methode müssen die Tetraeder-Elemente den Rand des Berechnungsgebietes möglichst genau beschreiben. Im Fall von gekrümmten Rändern werden die Oberflächen der Tetraeder-Elemente über eine Polynominterpolation mit Kontrollpunkten diskretisiert (vgl. Kapitel 5.3). Die Verteilung der Kontrollpunkte im Inneren der Oberflächen-Dreiecke wird über das lokale Referenzelement und eine vorgegebene Knotenverteilung [12] definiert (vgl. Abbildung 6.8).

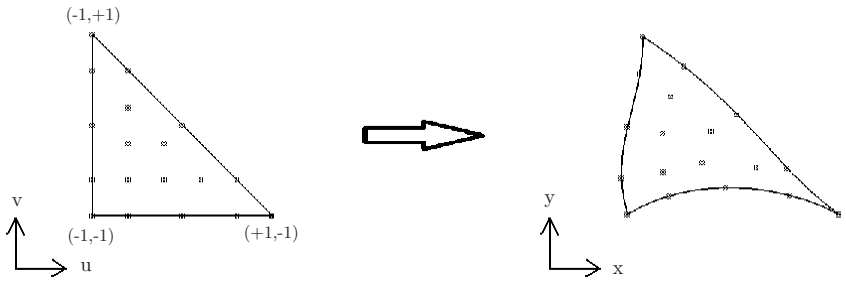


Abbildung 6.8: Standard-Dreieck und gekrümmtes Element

Die Diskretisierung von gekrümmten Tetraeder-Elementen hoher Ordnung erfolgt durch die Diskretisierung aller Dreiecksflächen und der Berücksichtigung der polynomialen Abbildungsfunktionen der einzelnen Oberflächen in der Abbildung des Standard-Tetraeder-Elementes (vgl. Abbildung 6.9).

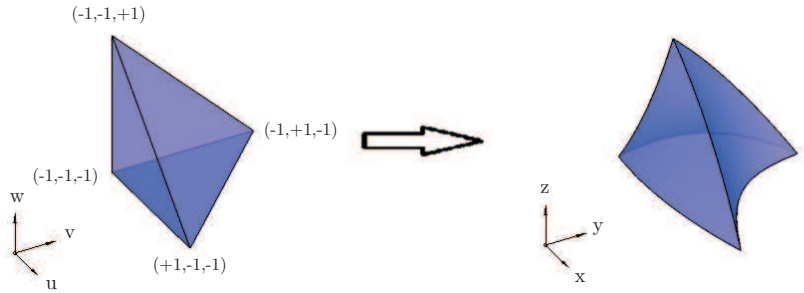


Abbildung 6.9: Standard-Tetraeder und gekrümmtes Element

Dabei können grundsätzlich Elemente entstehen, die für eine Berechnung mit der *p*-Version der Finite-Elemente-Methode geometrisch nicht geeignet sind. Zur Überprüfung der Tauglichkeit eines Elementes werden die partiellen Ableitungen der Tetraeder-Abbildungsfunktion berechnet und die Determinante der dazu gehörigen Jakobimatrix berechnet (vgl. Kapitel 2.4). Diese muss innerhalb des gesamten Elementes positiv sein. Ist dies nicht der Fall, so weist das Element beispielsweise Überlappungen, Selbstüberschneidungen oder unzulässige Verwindungen auf und resultiert in unbrauchbaren Berechnungsergebnissen. Solche geo-

metrisch fehlerhaften Elemente können in der Regel durch lokale Netzverfeinerung eliminiert werden.

6.2.5 Tetraeder-Vernetzung

Der vollständige Algorithmus zur Vernetzung von indirekten Volumenmodellen mit Tetraeder-Elementen erfolgt in folgenden Schritten:

1. *Importiere Oberflächenmodell:* Das Randdarstellungsmodell (BRep-Modell) wird importiert und der Nullraum der Flächen-Kanten Matrix (vgl. Kapitel 2.5.2) auf Gültigkeit überprüft.

2. *Generiere Oberflächennetz:* Die Oberfläche des Modells wird mit dem in Kapitel 5.2 eingeführten Vernetzungsalgorithmus in Dreieckselemente unterteilt.

3. *Netgen:* Die Oberflächen-Dreiecke der im ersten Schritt identifizierten Volumina werden ausgerichtet (vgl. Abschnitt 6.2.3) und volumenweise an *Netgen* übergeben. Tetraeder-Elemente werden aus *Netgen* importiert.

4. *Diskretisierung der Tetraeder-Oberflächen:* Die Tetraeder-Oberflächen werden durch polynomiale Abbildungsfunktionen innerhalb des Tetraeder-Standardelementes diskretisiert (vgl. Abschnitt 6.2.4).

6.2.6 Beispiele

Schraube

Das erste Beispiel zeigt das Modell einer Schraube in der Randdarstellung (vgl. Abbildung 6.10(a)). Abbildung 6.10(b) zeigt das Oberflächennetz aus Dreieckselementen, welches mit dem Dreiecksvernetzer *DoMesh* erstellt wurde. Nach Ausrichtung der Elemente und der Vernetzung mit *Netgen* entsteht das lineare Tetraeder-Netz (vgl. Abbildung 6.10(c)), welches abschließend durch eine Diskretisierung der Oberflächen in ein Teraedernetz hoher Ordnung überführt wird (vgl. Abbildung 6.10(d)).

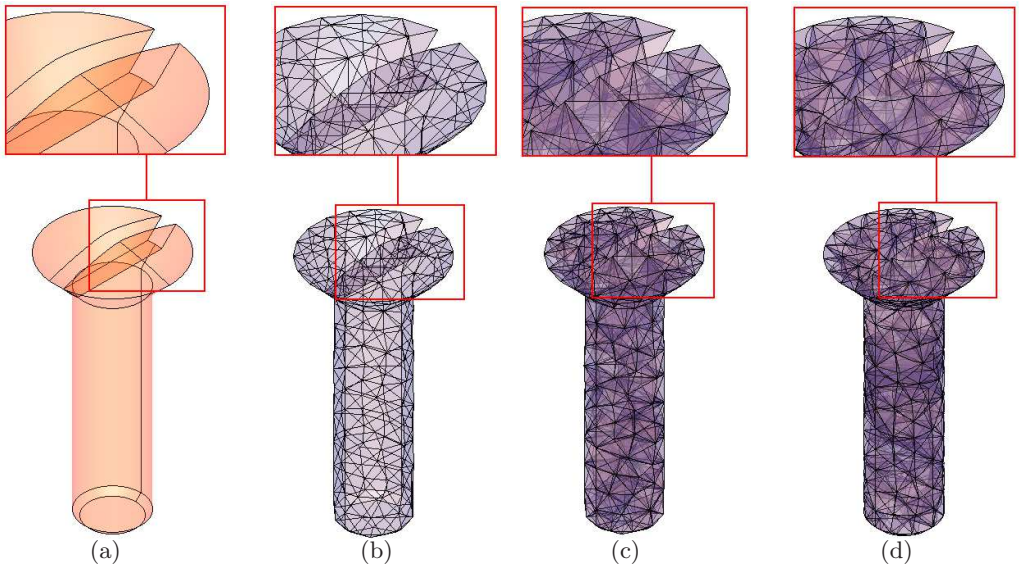


Abbildung 6.10: Schraube

Verbindungsstange

Das zweite Beispiel zeigt das Randdarstellungmodell einer Verbindungsstange (vgl. Abbildung 6.11(a)), das dazu gehörige Dreiecksnetz (vgl. Abbildung 6.11(b)), das lineare Teraeder-Netz (vgl. Abbildung 6.11(c)) und das Tetraeder-Netz hoher Ordnung (vgl. Abbildung 6.11(d)).

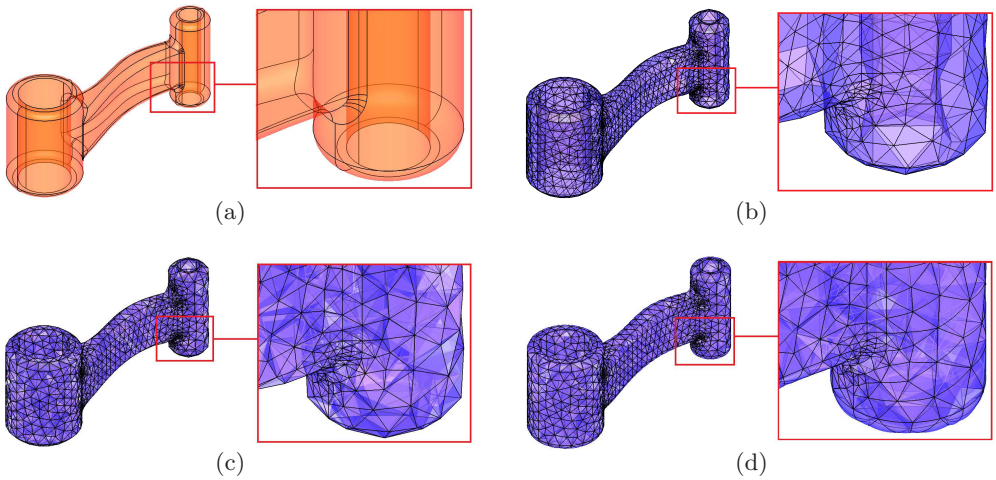
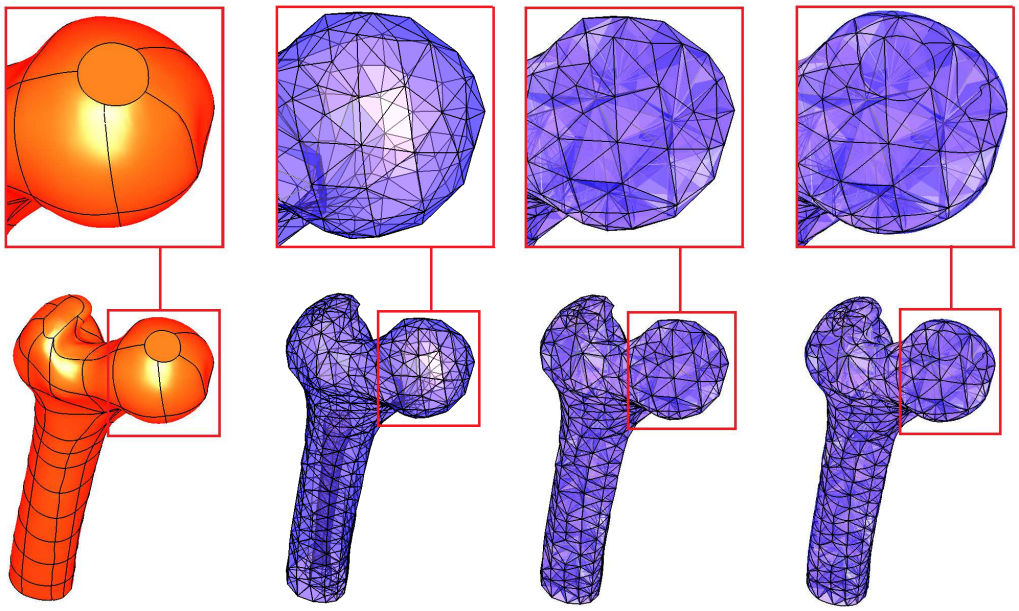


Abbildung 6.11: Verbindungsstange

Oberschenkelknochen

Das dritte Beispiel zeigt das Modell eines Oberschenkelknochens (vgl. Abbildung 6.12(a)), samt Dreiecksnetz (vgl. Abbildung 6.12(b)), linearem Teraeder-Netz

(vgl. Abbildung 6.12(c)) und dem abschließenden Tetraeder-Netz hoher Ordnung (vgl. Abbildung 6.12(d)).



(a) (b) (c) (d)

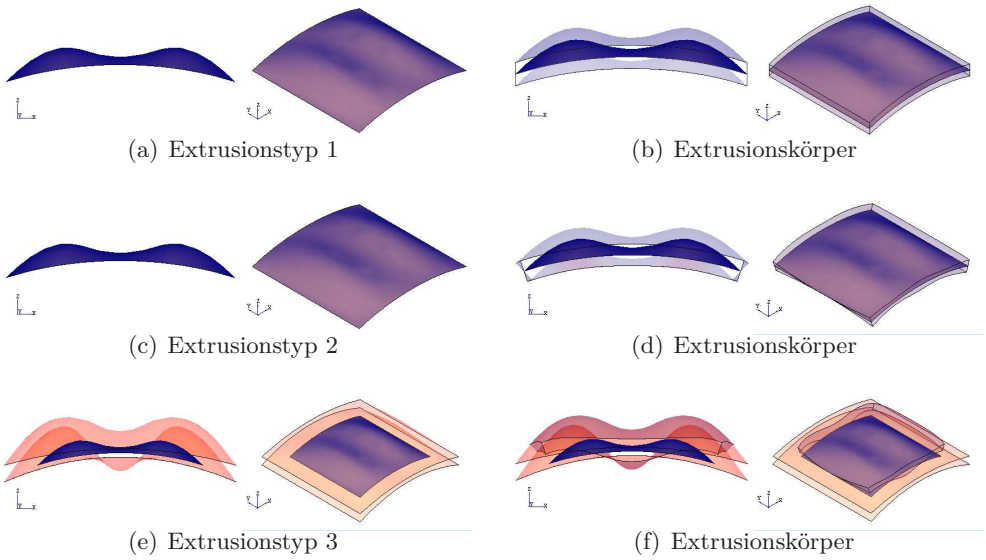
Abbildung 6.12: Knochen

6.3 Hexaeder-Vernetzung von dünnwandigen Extrusionsmodellen

Im nächsten Abschnitt wird der im Rahmen dieser Arbeit entwickelte Algorithmus zur Generierung von Hexaeder-Netzen für die Finite-Elemente-Methode hoher Ordnung (vgl. Kapitel 3.3.4) dargelegt. Der Algorithmus basiert auf einfachen Abbildungstechniken für schalenartige Flächenmodelle. Üblicherweise liegt der Schwerpunkt der Forschung bei der Vernetzung mit Abbildungstechniken in der automatischen Generierung von Mittelflächenmodellen für beliebige dreidimensionale Strukturen [38, 94, 57, 11]. Der implementierte Algorithmus geht von einem eigens entwickelten Modell aus Referenzflächen als Ausgangsgeometrie aus. Dafür konzentriert er sich auf die Vernetzung von unterschiedlichen geometrischen Szenarien, in denen sich die Referenzflächen parallel, senkrecht und schiefwinklig berühren oder schneiden. Die vereinfachenden geometrischen Annahmen haben auf die computergestütze Konstruktion des Modells keinen wesentlichen Einfluß, da komplexe Platten- und Schalenmodelle in der Regel sowieso anhand von Mittelflächen modelliert werden.

6.3.1 Topologisches Modell

Das topologische Modell der zu vernetzenden Struktur basiert auf den drei Extrusionsmodellen aus Kapitel 2.3.3 (vgl. Abbildung 6.13,6.14).



(a) Extrusionstyp 1 (b) Extrusionskörper

(c) Extrusionstyp 2 (d) Extrusionskörper

(e) Extrusionstyp 3 (f) Extrusionskörper

Abbildung 6.13: Einfache Extrusionsmodelle

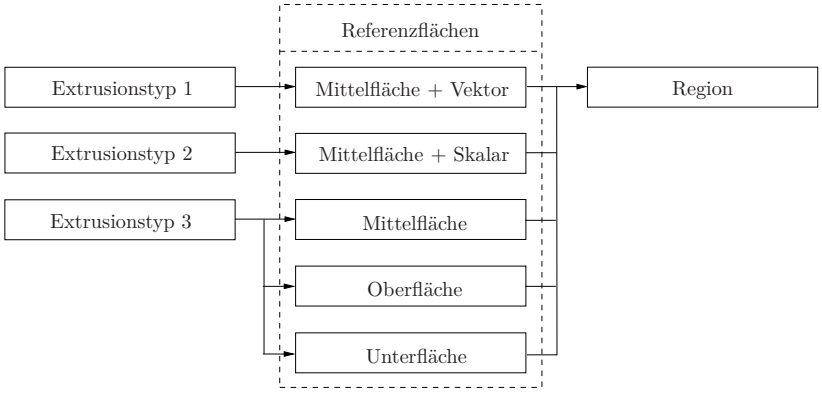
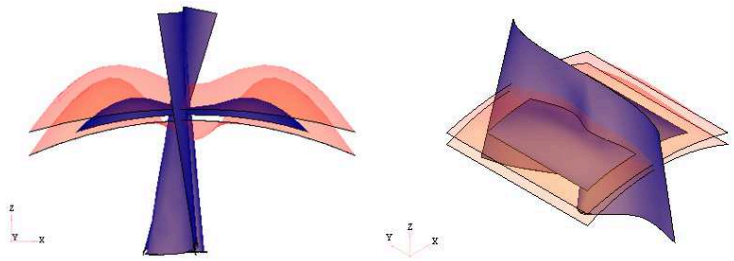


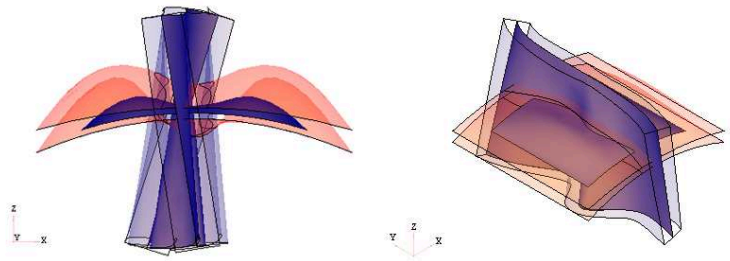
Abbildung 6.14: Topologisches Extrusionsmodell

Dabei lassen sich diese drei Extrusionstypen auf unterschiedliche Art und Weise miteinander kombinieren. Volumenkörper, die aus mehreren Extrusionskörpern zusammen gesetzt sind, resultieren in konformen Netzen, sobald sich die Regionen der Mittelflächen gemeinsame topologische Kanten oder Knoten teilen. Die Mittelflächen bei solchen multiplen Extrusionsmodellen dürfen sich dabei parallel, senkrecht oder schiefwinklig berühren und können zu unterschiedlichen Extrusionstypen gehören. Der entwickelte Algorithmus kann in der aktuellen Implementierung bis zu vier Mittelflächen pro Kante und bis zu 12 Mittelflächen pro Knoten miteinander verbinden. Auf diese Weise lässt sich jedoch eine Vielzahl an realen Systemen für die Vernetzung mit Hexaeder-Elementen beschreiben. Abbildung (6.15(a)) zeigt ein Beispiel für ein multiples Extrusionsmodell, bestehend aus einem Extrusionskörper von Extrusionstyp 2 und einem Extrusionskörper

von Extrusionstyp 3, mit sich schneidenden Mittelflächen. Abbildung (6.15(b)) zeigt den resultierenden Volumenkörper.



(a) Mittelflächenmodell



(b) Extrusionskörper

Abbildung 6.15: Multiples Extrusionsmodell

6.3.2 Versatzabbildung

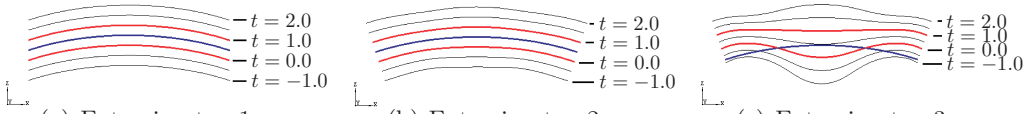
Zunächst wird der jeder einzelnen Mittelfläche anhaftende Extrusionsraum als parametrische Abbildungsfunktion z formuliert. Diese Abbildungsfunktion ordnet jedem Punkt auf der Mittelfläche $\mathbf{x}$ über einen Versatzparameter t die globale Position $\hat{\mathbf{x}}$ innerhalb des Extrusionskörpers zu.

$$z(\mathbf{x}, t) = \hat{\mathbf{x}}, \qquad t \in \mathbb{R} \tag{6.1}$$

Der Versatzparameter t ist dabei so definiert, dass er unabhängig von Form und Dicke des jeweiligen Extrusionskörper auf der unteren Randfläche immer den Wert 0.0 und auf der oberen Randfläche immer den Wert 1.0 annimmt. Die inverse Versatzabbildung z^{-1} bestimmt für jeden Punkt $\hat{\mathbf{x}}$ innerhalb des Extrusionskörpers die Referenzkoordinaten $\mathbf{x}$ auf der Mittelfläche sowie den Versatzparameter t.

$$z^{-1}(\hat{\mathbf{x}}) = \mathbf{x}, t \tag{6.2}$$

Abbildung 6.16 zeigt die Isolinien der Versatzabbildung auf einer der Randkanten für die drei Extrusionsmodelle aus Abbildung 6.13. Der Raum innerhalb der

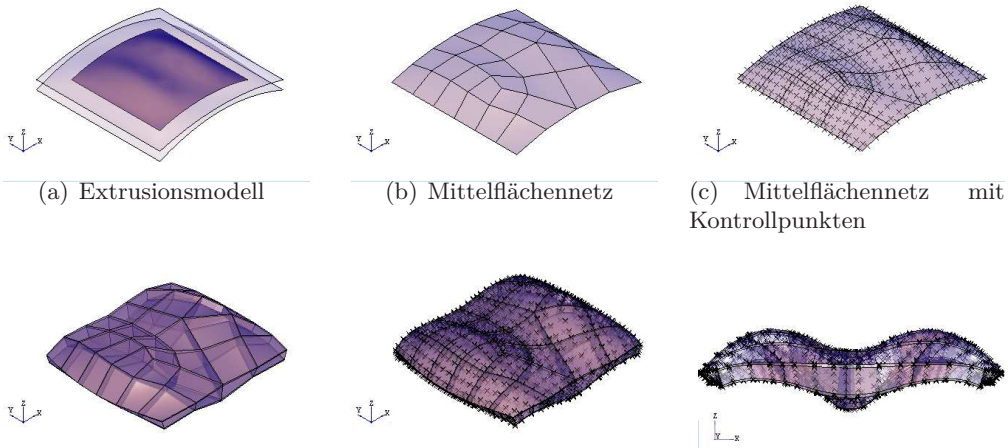


(a) Extrusionstyp 1 (b) Extrusionstyp 2 (c) Extrusionstyp 3

Abbildung 6.16: Isolinien

Isolinien $t = 0.0$ und $t = 1.0$ definiert dabei den vorderen Rand des finalen Extrusionskörpers.

Bei einer Extrusion des Mittelflächennetzes mit mehreren Elementschichten in Dickenrichtung können die Schichtgrenzen der Elemente durch fest vorgegebene innere Versatzparameter t_i vordefiniert werden mit $0.0 < t_i < 1.0$. Abbildung 6.17 zeigt die Vernetzung des Extrusionsmodells aus Abbildung 6.13(e) mit den vordefinierten Schichtgrenzen ($t_0 = 0.0, t_1 = 0.1, t_2 = 0.9, t_3 = 1.0$). Die Extrusion der Knoten und der Kontrollpunkte erfolgt durch Ansetzen der Versatzabbildung (6.1) auf die Koordinaten der Mittelfläche.



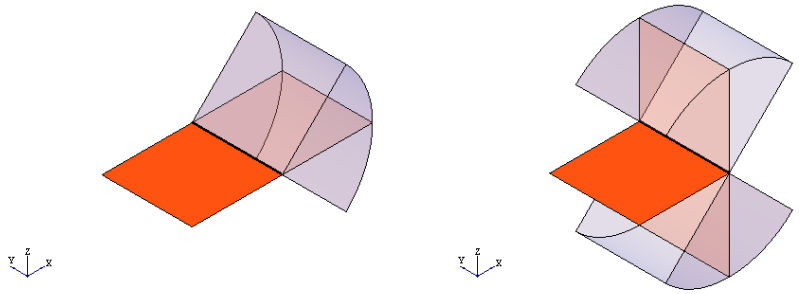
(a) Extrusionsmodell (b) Mittelflächennetz (c) Mittelflächennetz mit Kontrollpunkten

(d) Extrudiertes Netz (e) Extrudiertes Netz mit Kontrollpunkten (f) Extrudiertes Netz mit Kontrollpunkten

Abbildung 6.17: Extrusion mit vordefinierten Schichtgrenzen

Bei multiplen Extrusionsmodellen besitzen Kanten, die zu mehreren Mittelflächen gleichzeitig gehören, eine Versatzabbildung pro angrenzender Mittelfläche. Die einzelnen Versatzabbildungen werden zu einem gemeinsamen Versatzabbildungsraum vereinheitlicht. Hierfür wird der Schnittwinkel zwischen den Mittelflächen ausgewertet und für eine Klassifizierung in *quasi-parallele* ($|\alpha| \leq \frac{1}{4} \cdot \pi$) und *quasi-senkrechte* ($\frac{1}{4} \cdot \pi < |\alpha| < \frac{3}{4} \cdot \pi$) Verbindungskanten angesetzt (vgl. Abbildung 6.18).

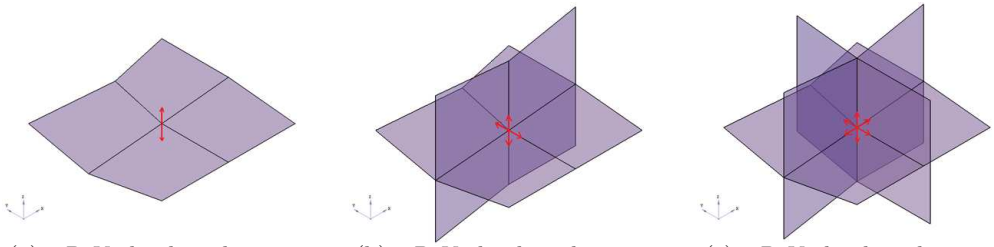
Knoten, die zu mehr als einer Mittelfläche gleichzeitig gehören, resultieren in unterschiedlichen Arten von Verbindungsknoten. Knoten, die nur *quasi-parallele* Mittelflächen verbinden werden als *1D Verbindungsknoten* angesetzt (vgl. Abbil-



(a) *quasi-parallele* Verbindungskante (b) *quasi-senkrechte* Verbindungskante

Abbildung 6.18: Klassifizierung der Kanten-Verbindungen

dung 6.19(a)). Zwei verschiedenen Sätze von nur *quasi-parallelen* Mittelflächen resultieren in *2D Verbindungsknoten* (vgl. Abbildung 6.19(b)), bei drei verschiedenen *quasi-parallelen* Mittelflächensätzen entstehen *3D Verbindungsknoten* (vgl. Abbildung 6.19(c)).



(a) *1D Verbindungsknoten* (b) *2D Verbindungsknoten* (c) *3D Verbindungsknoten*

Abbildung 6.19: Klassifizierung der Knoten-Verbindungen

1D Verbindungsknoten vereinheitlichen die parametrischen Versatzabbildungen der benachbarten Mittelflächen in eine globale *1D Versatzabbildung*. Hierfür werden die Schnittpunkte der extrudierten Mittelflächen ermittelt und der Abbildungsparameter t zwischen den Schnittpunkten linear interpoliert ($t_a = 0.0, t_b = 1.0$). Abbildung 6.20 zeigt ein Beispiel für zwei sich berührende Mittelflächen, die in einer *1D Versatzabbildung* entlang der gemeinsamen Kante resultieren.

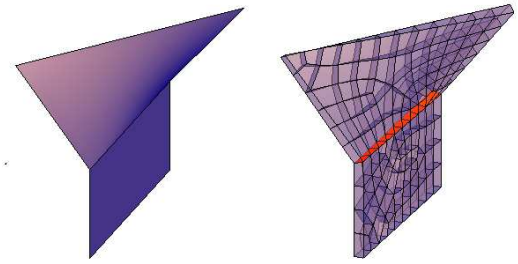
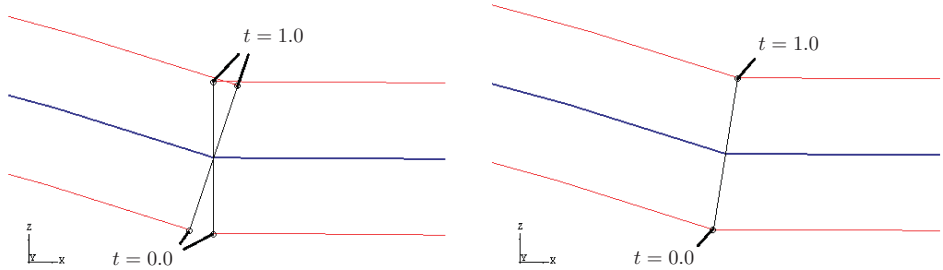


Abbildung 6.20: Mittelflächen-Geometrie und Netz (*1D Versatzabbildung*)

Abbildung 6.21 zeigt die Abbildungsparameter der einzelnen Mittelflächen und den gemeinsamen Abbildungsparameter nach Vereinheitlichung der *1D Versatzabbildung* für einen beliebigen Punkt auf der Kante.



(a) Abbildungsparameter - Mittelflächen (b) Gemeinsamer Abbildungsparameter

Abbildung 6.21: 1D Versatzabbildung

Die einzelnen Versatzabbildungen in *2D Verbindungsknoten* werden zu einer globalen *2D Versatzabbildung* zusammen gefasst. Hierfür werden die Schnittpunkte zwischen den extrudierten *quasi-senkrechten* Mittelflächen ermittelt und bilinear auf einen zweidimensionalen Abbildungsraum interpoliert. Abbildung 6.22 zeigt ein Beispiel für zwei sich schneidende Mittelflächen, die in einer *2D Versatzabbildung* entlang der gemeinsamen Kante resultieren.

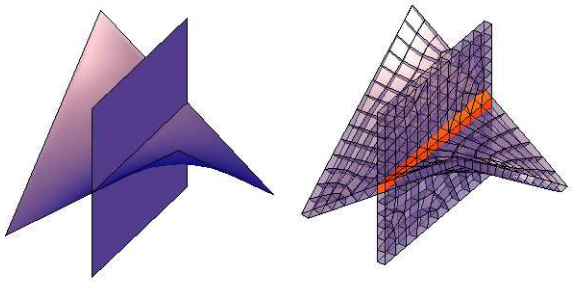
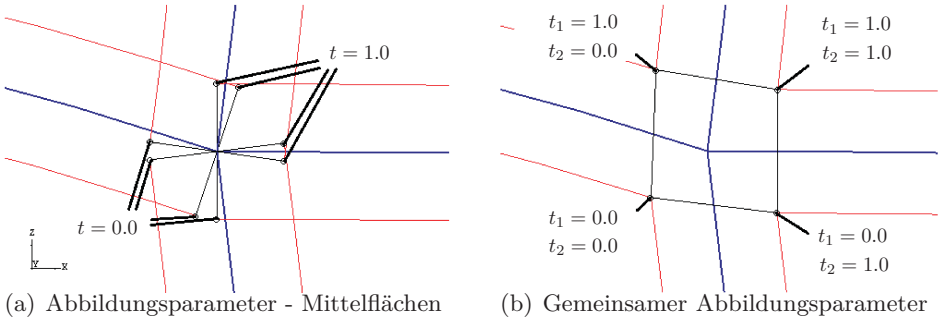


Abbildung 6.22: Mittelflächen-Geometrie und Netz (*2D Versatzabbildung*)

Abbildung 6.23 zeigt die Abbildungsparameter von vier unterschiedlichen, sich berührenden Mittelflächen und den gemeinsamen Abbildungsparameter nach Vereinheitlichung in eine *2D Versatzabbildung*.

Bei *3D Verbindungsknoten* entsteht aus den einzelnen Versatzabbildungen eine globale *3D Versatzabbildung*. Zunächst werden die acht Eckpunkte aus den Schnittpunkten der extrudierten Mittelflächen ermittelt, anschließend wird die *3D Versatzabbildung* durch eine trilineare Interpolationsfunktion formuliert. Abbildung 6.24 zeigt ein Beispiel für drei sich schneidende Mittelflächen, die in einer *3D Versatzabbildung* innerhalb des gemeinsamen Schnittpunktes resultieren.



(a) Abbildungsparameter - Mittelflächen (b) Gemeinsamer Abbildungsparameter

Abbildung 6.23: 2D Versatzabbildung

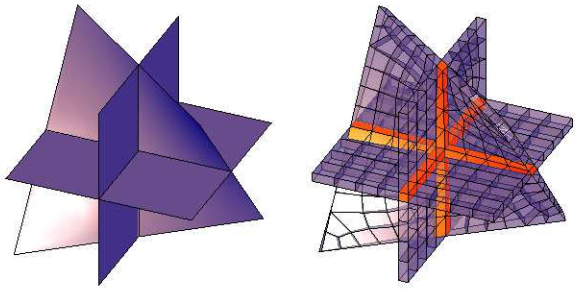
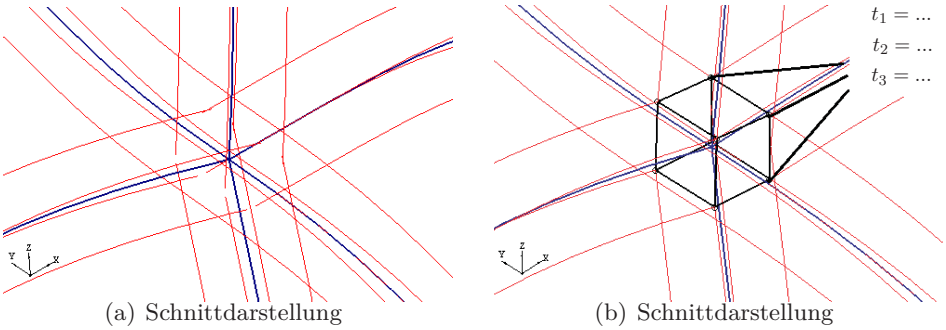


Abbildung 6.24: Mittelflächen-Geometrie und Netz (*3D Versatzabbildung*)

Abbildung 6.25 zeigt die Abbildungsparameter von 12 unterschiedlichen, sich berührenden Mittelflächen und den gemeinsamen Abbildungsparameter nach Vereinheitlichung in eine *3D Versatzabbildung*.



(a) Schnittdarstellung (b) Schnittdarstellung

Abbildung 6.25: 3D Versatzabbildung

6.3.3 Versatzgeometrie

Bei der Vernetzung von multiplen Extrusionsmodellen wird zunächst die Geometrie und die Topologie der Mittelflächen an den Verbindungsstellen manipuliert und eine Versatzgeometrie generiert. Die Generierung der Versatzgeometrie erfolgt an allen *quasi-senkrechten* Verbindungsstellen und spielt eine zentrale Rolle

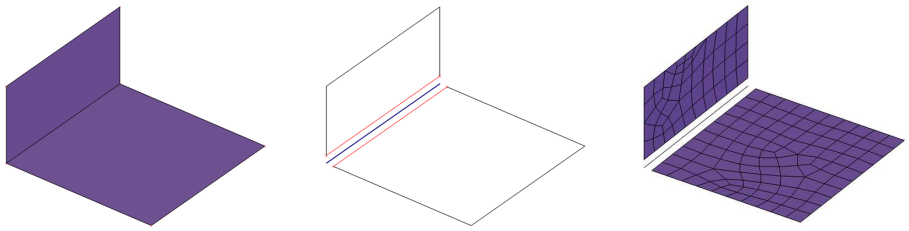
in der Vernetzung von multiplen Extrusionsmodellen. In einem ersten Schritt werden alle Verbindungskanten auf den Mittelflächen dupliziert und über die Versatzabbildung an die jeweiligen Rand der Verbindungsstelle im extrudierten Volumenkörper projeziert. Die Projektion erfolgt über eine Kopplung der Abbildung und Umkehrabbildung der Originalkante im Bogenlängenmaß (vgl. Abschnitt 2.4.2) mit der Versatzabbildung der zugrundeliegenden Mittelfläche bei einem vorgegebenen Versatzparameter t_{fixed}, welcher der Isolinie der Zielkurve entspricht (vgl. Abbildung 6.16).

$$\mathbb{R} \rightarrow \mathbb{R}^3 \begin{cases} \mathbf{y}(s) = \mathbf{x} \\ z(\mathbf{x}, t_{fixed}) = \hat{\mathbf{x}} \end{cases} \tag{6.3}$$

$$\mathbb{R}^3 \rightarrow \mathbb{R} \begin{cases} z^{-1}(\hat{\mathbf{x}}) = \mathbf{x}, t_{fixed} \\ y^{-1}(\mathbf{x}) = s \end{cases} \tag{6.4}$$

In einem zweiten Schritt werden die Verbindungskanten auf den angrenzenden Mittelflächen durch die Versatzkanten ersetzt und neue Randkantenzüge auf den Mittelflächen definiert. Die Versatzkanten werden über eine referenzierte Teilung als Slave-Kanten an die Verbindungskante als Master-Kante gekoppelt (vgl. Abschnitt 5.2.4). Dadurch werden *quasi-senkrechte* Mittelflächen topologisch voneinande getrennt, werden jedoch während der Vernetzung äquivalent unterteilt. Der in dieser Arbeit entwickelte Algorithmus unterscheidet zwischen drei möglichen Szenarien, je nachdem, ob die Verbindungskante zum Rand, zu einer Lochkante, oder zu einer Zwangskante auf der jeweiligen Mittelfläche gehört.

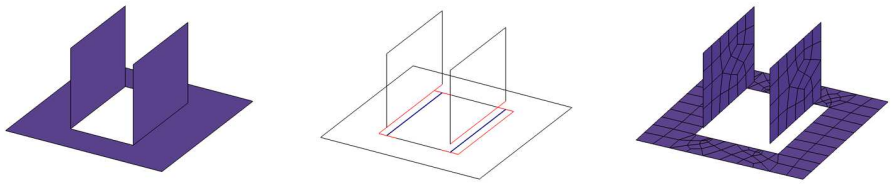
Verbindungskanten, die Teil einer äußeren Regionsberandung sind (vgl. Abbildung 6.26(a)) projezieren die Versatzkante in das Innere der Region und trimmen die angrenzenden Kanten an den Knoten der Versatzkante (vgl. Abbildung 6.26(b)). Der Versatzparameter t_{fixed} nimmt dabei je nach Orientierung der Mittelfläche entweder den Wert 0.0 oder den Wert 1.0 an. Das Oberflächennetz wird nun auf der Versatzgeometrie generiert und die Versatzkanten äquivalent unterteilt (Abbildung 6.26(c)).



(a) Mittelflächen (b) Versatzgeometrie (c) Oberflächennetz

Abbildung 6.26: Verbindung durch Randkanten

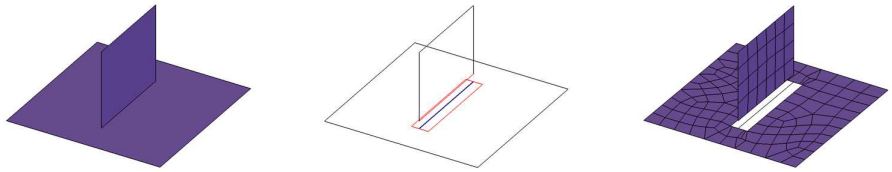
Verbindungskanten, die Teil einer inneren Regionsberandung sind (vgl. Abbildung 6.27(a)), projezieren die Versatzkante in das Innere der Region und trimmen die angrenzenden Kanten an den Knoten der gegenüberliegenden Versatzkante (vgl. Abbildung 6.27(b)). Der Randkantenzug wird durch die Einführung von neuen Kanten, welche die Knoten der gegenüberliegenden Versatzkanten verbinden, geschlossen. Um den Elementgenzen der angrenzenden *quasi-senkrechten* Mittelfläche nach der Extrusion zu entsprechen, werden diese Kanten mit einer vorgegebenen Unterteilung geteilt (vgl. Kapitel 5.2.4), und eine weitere Unterteilung der Teilkanten während der Oberflächen-Vernetzung unterdrückt (vgl. Abbildung 6.27(c)).



(a) Mittelflächen (b) Versatzgeometrie (c) Oberflächennetz

Abbildung 6.27: Verbindung durch Lochkanten

Verbindungskanten, die Teil einer Zwangskante sind (vgl. Abbildung 6.28(a)), werden durch eine geschlossene inner Regionsberandung (Loch) ersetzt. Der Kantenzug für die innere Berandung wird durch beiden Versatzkanten ($t_{1,fixed} = 0.0, t_{2,fixd} = 1.0$) definiert und analog zum Vorgehen bei Lochkanten durch unterteilte Verbindungskanten geschlossen (vgl. Abbildung 6.28(b)). Nach Generierung des Oberflächennetzes, fallen die unterteilten Verbindungskanten mit den Elementgrenzen des extrudierten *quasi-senkrechten* Mittelflächennetzes zusammen (vgl. Abbildung 6.28(c)).



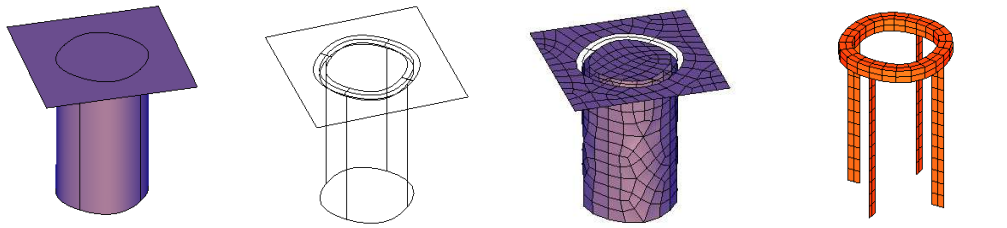
(a) Mittelflächen (b) Versatzgeometrie (c) Oberflächennetz

Abbildung 6.28: Verbindung durch Zwangskanten

6.3.4 Verbindungsnetz

Um die topologisch getrennten Oberflächennetze auf der Versatzgeometrie während der Extrusion wieder zusammenzufügen, wird zunächst ein strukturiertes Verbindungsnetz entlang der ursprünglichen Verbindungskanten bestehend aus Viereckselementen und Hexaeder-Elementen generiert. Das Skelett dieses Verbindungsnetzes wird durch die unterteilten Verbindungskanten definiert. Das Aufspannen der Elemente erfolgt über die Versatzabbildungen, die jedem Punkt auf

dem Skelett zugeordnet sind (vgl. Abschnitt 6.3.2). An Stellen mit nur *quasi-parallelen* Mittelflächen entstehen dabei Viereckselemente. Sind *quasi-senkrechte* Mittelflächen in einem Punkt verbunden, so entstehen auf diese Weise Hexaeder-Elemente (vgl. Abbildung 6.29).

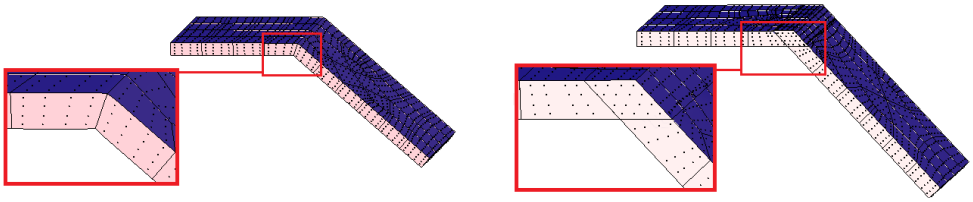


(a) Mittelflächenmodell (b) Versatzgeometrie (c) Oberflächennetz (d) Verbindungsnetz

Abbildung 6.29: Generierung des Verbindungsnetzes

6.3.5 Extrusion

Die Extrusion der Oberflächennetze erfolgt durch Ansetzen der Versatzabbildung auf allen Knoten und Kontrollpunkten des Mittelflächennetzes. Knoten und Kanten, die auf einer der Verbindungsstellen liegen werden dabei jedoch nicht direkt extrudiert, sondern importieren die entsprechenden Zielknoten und Zielkanten aus dem Verbindungsnetz. Um eine Überlappung der Kontrollpunkte in den Elementen, die an eine schiefwinklige Verbindungsstelle angrenzen, während der Extrusion zu vermeiden, werden die Kontrollpunkte in diesen Elementen während der Extrusion korrigiert (vgl. Abbildung 6.30).



(a) *quasi-parallele* Verbindungsstelle (b) *quasi-senkrechte* Verbindungsstelle

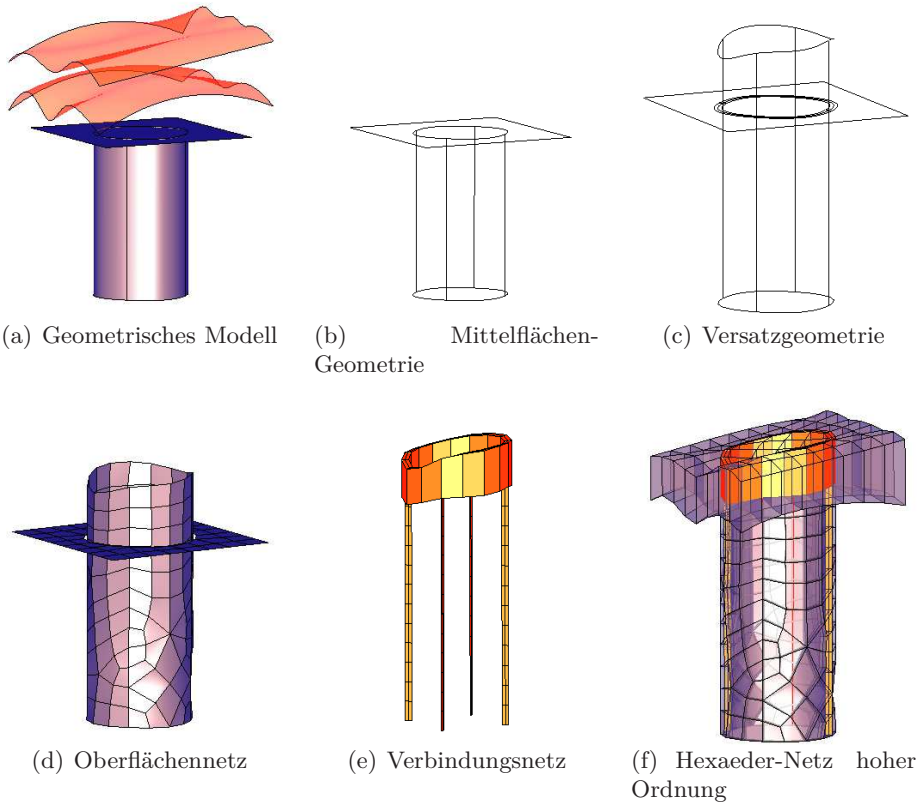
Abbildung 6.30: Kontrollpunkte im Anschlussbereich

Der komplette Algorithmus zur Vernetzung von multiplen Extrusionsmodellen erfolgt in folgenden Schritten:

1. Importiere Oberflächenmodell (vgl. Abbildung 6.31(a)).
2. Importiere Kantenzüge der Referenzflächen. (vgl. Abbildung 6.31(b)).
3. Generiere Versatzgeometrie (vgl. Abbildung 6.31(c)).
4. Generiere Oberflächennetz (vgl. Abbildung 6.31(d)).

5. Generiere Verbindungsnetz (vgl. Abbildung 6.31(e)).

6. Extrudiere Oberflächennetz und verknüpfe Verbindungsnetz (vgl. Abbildung 6.31(f)).



(a) Geometrisches Modell (b) Mittelflächen-Geometrie (c) Versatzgeometrie

(d) Oberflächennetz (e) Verbindungsnetz (f) Hexaeder-Netz hoher Ordnung

Abbildung 6.31: Vernetzung von multiplen Extrusionsmodellen

In dem in Abbildung 6.31 gezeigten Beispiel werden die Zylinderflächen durch die Projektion bis zur unteren Begrenzungsfläche (rot) verlängert. Dies ist möglich, da die Zylinderflächen ursprünglich aus einer Trimm-Operation resultieren und auf der neu gewonnenen Fläche eine Abbildung und Umkehrabbildung existiert.

6.3.6 Beispiele

Schiffshülle

Die Leistungsfähigkeit des Algorithmus wird zunächst an einem multiplen Mittelflächenmodell einer Schiffshülle mit Quer- und Längsspanten demonstriert (vgl. Abbildung 6.32). Die Schiffshaut (blau) wird durch *BSpline-Flächen* beschrieben, welche die Mittelflächen eines Extrusionskörpers vom Extrusionstyp 2 darstellen. Die Quer- und Längsspanten (rot), werden als ebene Mittelflächen vom Extrusionstyp 1 modelliert und *quasi-senkrecht* an die Schiffshaut angeschlossen.

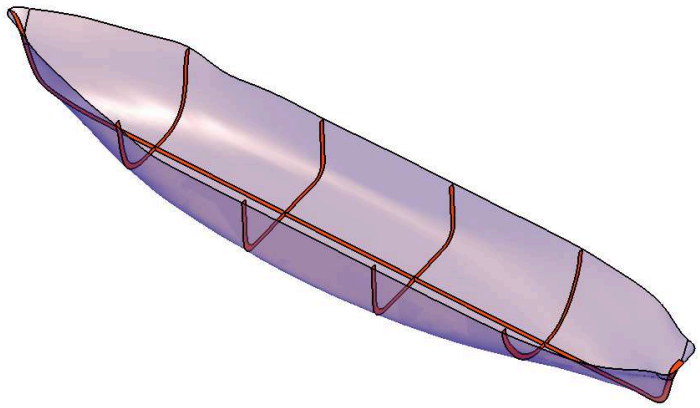
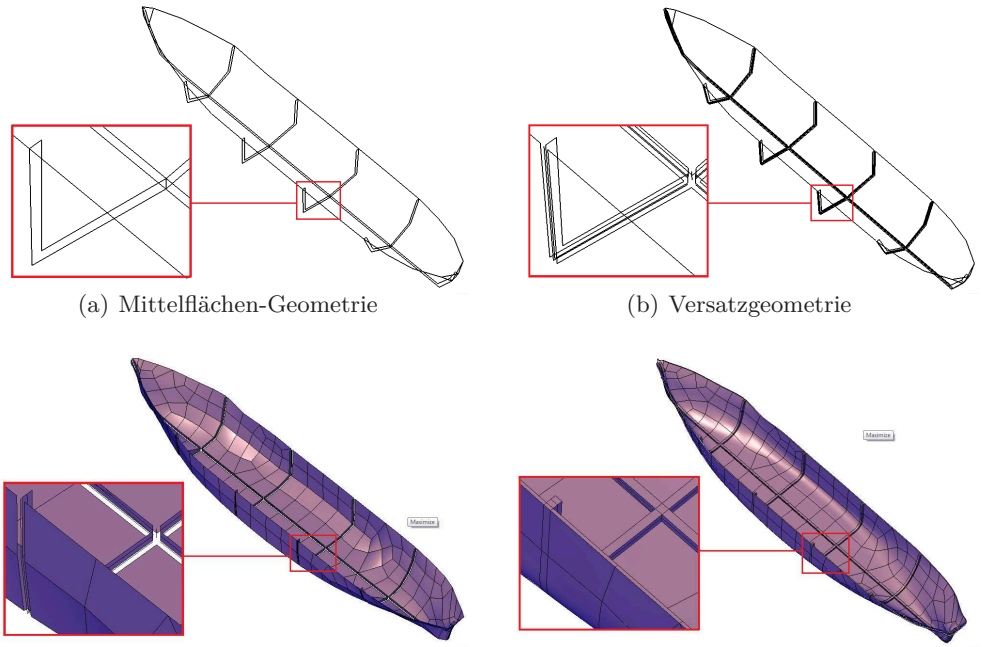


Abbildung 6.32: Mittelflächenmodell einer Schiffshülle mit Quer- und Längsspanten

Der Extrusionsvektor der Mittelflächen an den Quer- und Längsspanten entspricht der Dicke der Spanten. Die Dicke der Schiffshaut geht als Skalar in das zugrunde liegende Mittelflächenmodell ein. Abbildung 6.33 zeigt die einzelnen Schritte des Netzgenerierungsprozesses und das finale Netz mit gekrümmten Hexaeder-Elementen hoher Ordnung.



(a) Mittelflächen-Geometrie

(b) Versatzgeometrie

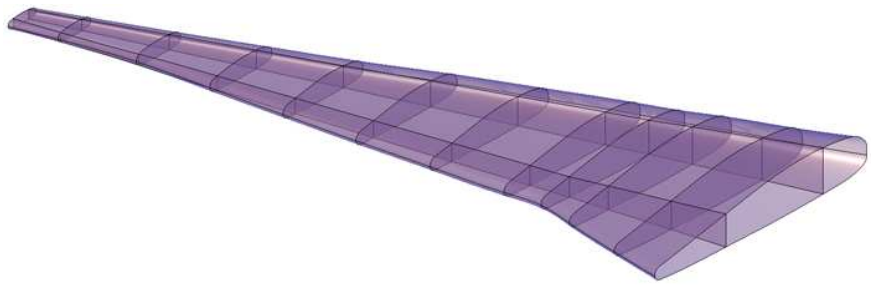
(c) Oberflächennetz

(d) Hexaeder-Netz hoher Ordnung

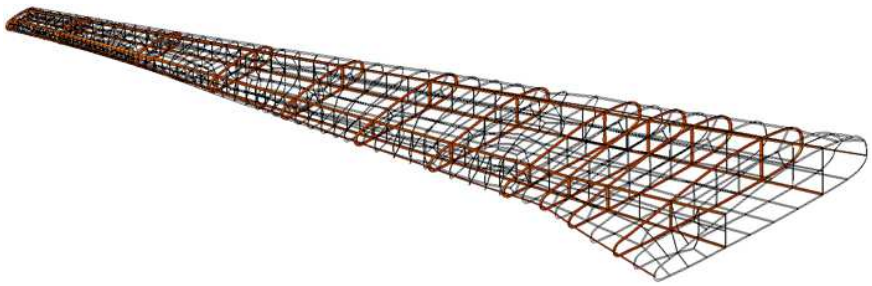
Abbildung 6.33: Vernetzung der Schiffshülle

Flugzeugflügel

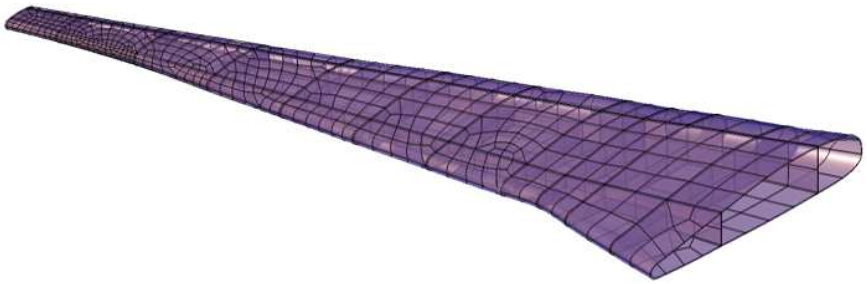
Das zweite Beispiel besitzt einen ähnlichen Aufbau wie das Modell der Schiffshülle und beschreibt einen Flugzeugflügel mit Beplankung, Rippen und Stegen. Die Beplankung ist als Extrusionskörper vom Extrusionstyp 2 modelliert, die Rippen und Stege als Extrusionskörper vom Extrusionstyp 1. Abbildung 6.34 zeigt das Modell des Flügels und das finale Netz.



(a) Versatzgeometrie



(b) Verbindungsnetz



(c) Hexaeder-Netz hoher Ordnung

Abbildung 6.34: Vernetzung des Flugzeugflügels

Windrad

Ein drittes Beispiel zeigt das multiple Mittelflächenmodell eines Windrades (vgl. Abbildung 6.35) und illustriert den Anschluss von *quasi-senkrechten* Verbindungen an doppelt gekrümmten Kanten. Die Flügel des Windrades (blau) sowie die Nabe (rot) sind beide als Extrusionskörper vom Extrusionstyp 2 modelliert, besitzen jedoch unterschiedliche Dicken.

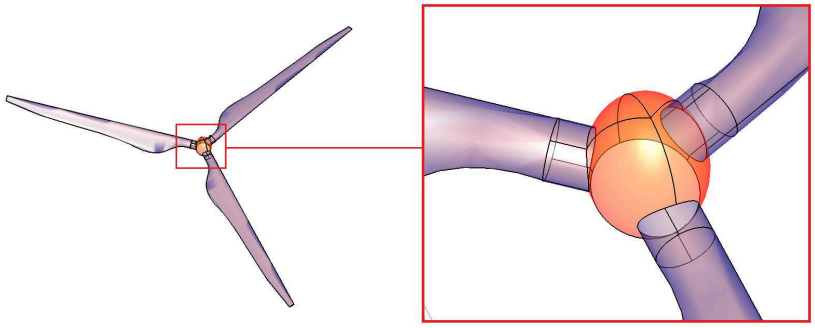
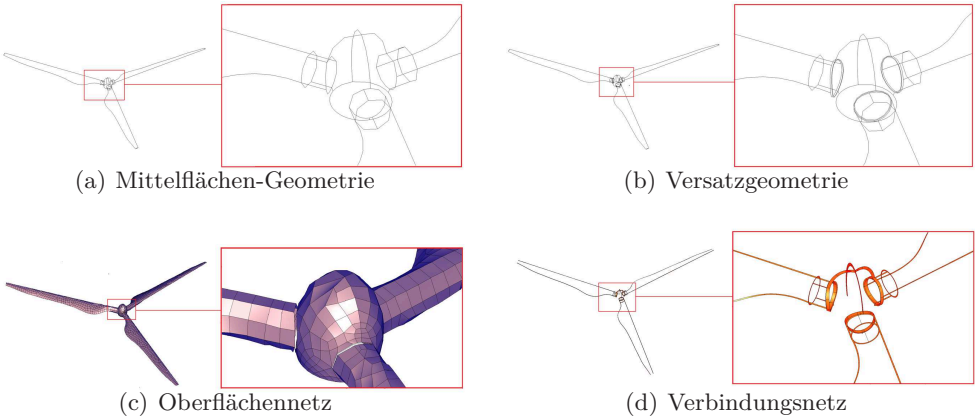


Abbildung 6.35: Mittelflächenmodell eines Windrades

Abbildung 6.36 zeigt die Zwischenschritte des Vernetzungsprozesses von der Mittelflächen-Geometrie (vgl. Abbildung 6.36(a)) über die Versatzgeometrie (vgl. Abbildung 6.36(b)), zum Oberflächennetz (vgl. Abbildung 6.36(c)), sowie dem Verbindungnetz (vgl. Abbildung 6.36(d)).



(a) Mittelflächen-Geometrie (b) Versatzgeometrie

(c) Oberflächennetz (d) Verbindungsnetz

Abbildung 6.36: Vernetzung des Windrades

Abbildung 6.37 zeigt das finale Netz bestehend aus gekrümmten Hexaeder-Elementen hoher Ordnung.

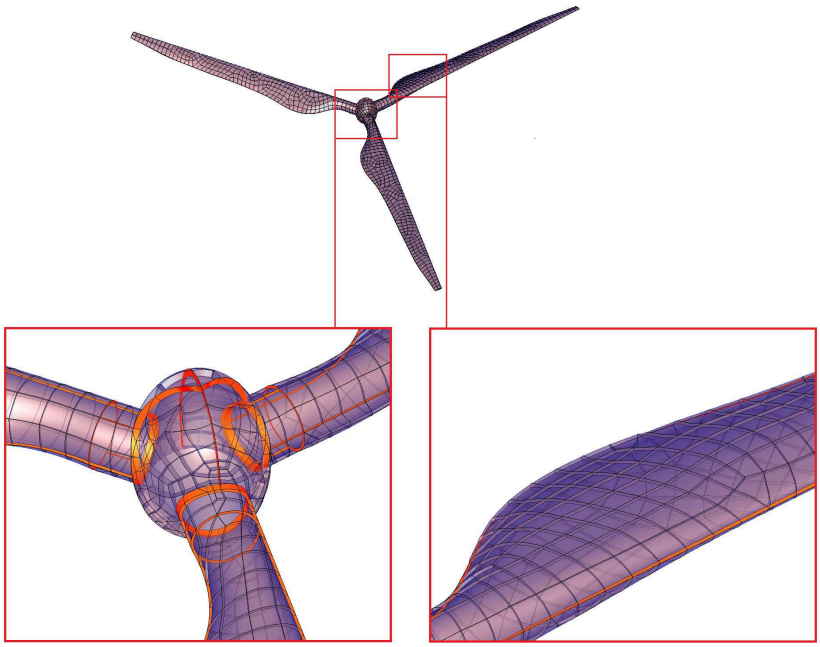


Abbildung 6.37: Hexaeder-Netz hoher Ordnung

Geigenkörper

Das vierte und abschliessende Beispiel beschreibt einen Geigenkörper (vgl. Abbildung 6.38). Es demonstriert die Leistungsfähigkeit des Algorithmus bei der Vernetzung von komplexen, gekrümmten Extrusionskörpern. Die Ober- und die Unterseite des Geigenkörpers ist als Extrusionskörper vom Extrusionstyp 3 mit zwei jeweils ebenen Mittelflächen und den dazugehörigen Ober- und Unterflächen (rot) als Begrenzungsflächen des Geigenkörpers modelliert. Die Begrenzungsflächen beschreiben dabei die tatsächliche Form des 'echten' Geigenkörpers. Die internen Strukturen innerhalb des Geigenkörpers, wie z.B. die f-Löcher, der Bassbalken und der Stimmstock, werden durch innere Randkanten und Zwangskanten auf den Mittelflächen definiert. Die Seitenwände des Geigenkörpers sind als Extrusionskörper vom Extrusionstyp 2 mit konstanter Dicke t modelliert. Die Konnektivität zwischen den Seitenwänden und der Ober- und Unterseite des Geigenkörpers wird durch die Verbindung der Seitenflächen an die Mittelflächen des Geigenbodens und Geigendeckels mit Hilfe von Zwangskanten definiert.

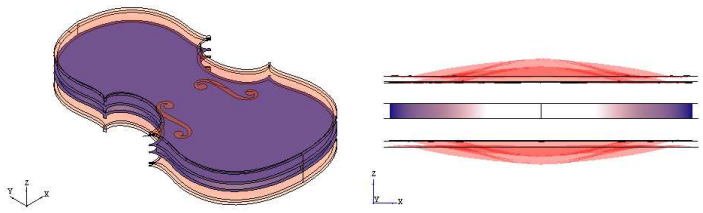
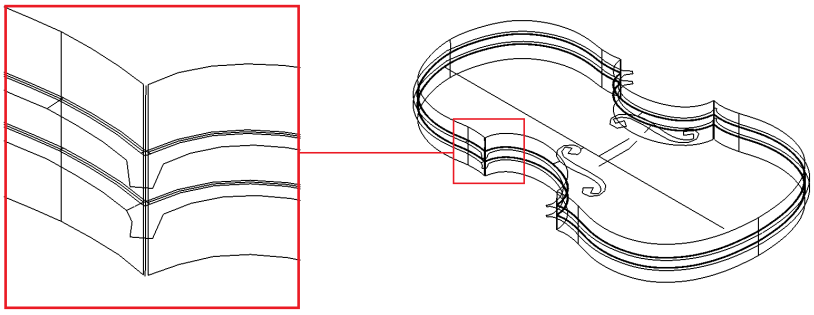
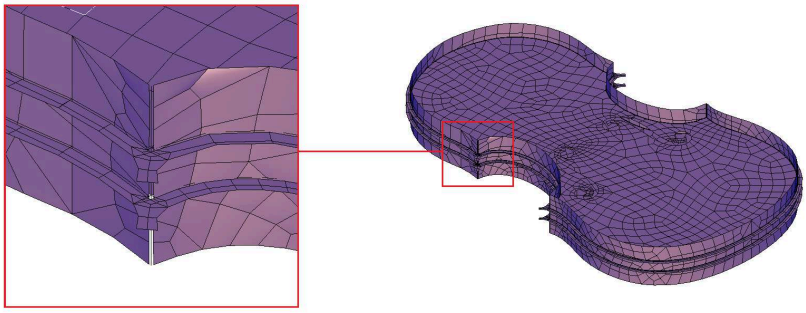


Abbildung 6.38: Mittelflächenmodell eines Geigenkörpers

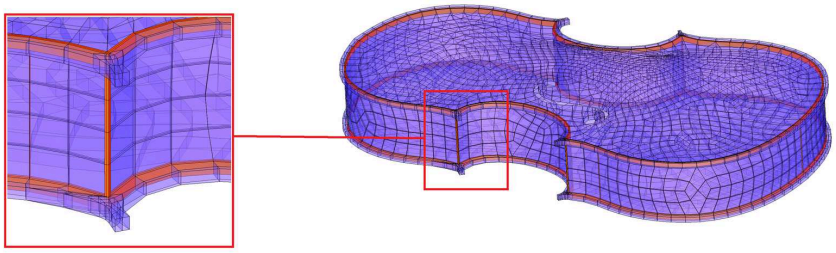
Abbildung 6.39 zeigt die einzelnen Schritte der Vernetzung von der Versatzgeometrie (vgl. Abbildung 6.39(a)) über das Oberflächennetz (vgl. Abbildung 6.39(b)), bis hin zum Hexaeder-Netz samt Verbindungsnetz (vgl. Abbildung 6.39(c)).



(a) Versatzgeometrie



(b) Oberflächennetz



(c) Hexaeder-Netz

Abbildung 6.39: Vernetzung des Geigenkörpers

Abbildung 6.40 zeigt das endgültige Netz des Geigenkörpers, bei dem Gegendeckel- und Geigenboden mit variierender Dicke zwischen 0.35 cm und 0.5 cm inklusive f-Löcher modelliert sind. Die Positionen des Bassbalkens und des Stimmstocks sind als Zwangskanten vorgegeben und können so konstruktiv durch eine Erhöhung der Steifigkeit innerhalb der Simulation berücksichtigt werden.

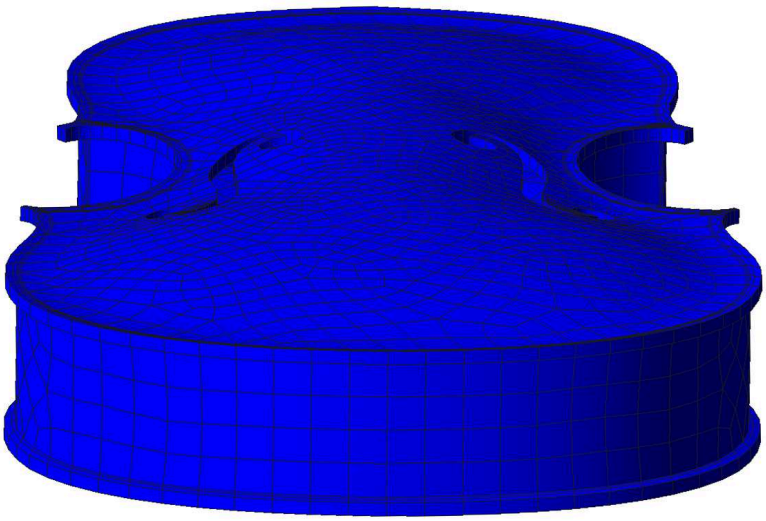


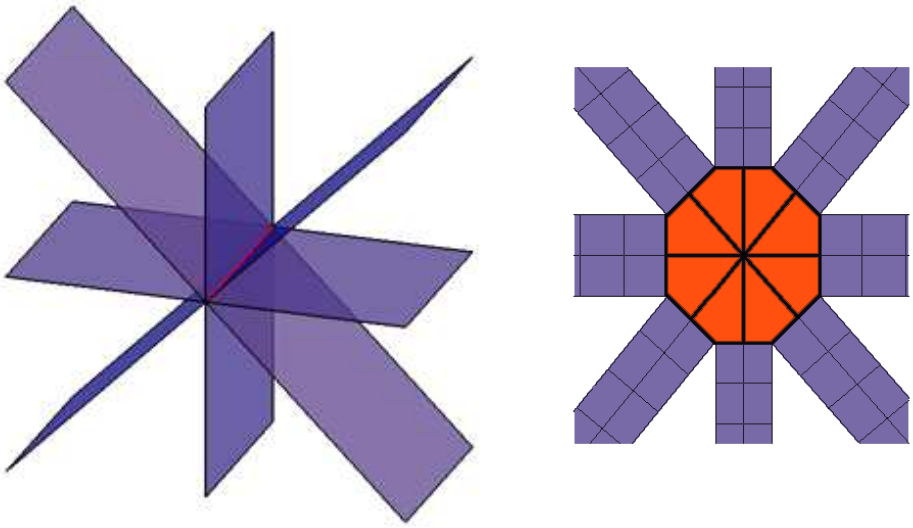
Abbildung 6.40: Hexaeder-Netz

6.3.7 Einschätzung

Mit dem gezeigten Algorithmus lässt sich eine Vielzahl von komplexen, schalenartigen Strukturen konform mit Hexaeder-Elementen hoher Ordnung vernetzen. Es lassen sich jedoch auch Szenarien generieren, für die der Algorithmus in seiner aktuellen Ausführung keine Lösung findet, oder bei denen Netze mit fehlerhaften Übergängen entstehen.

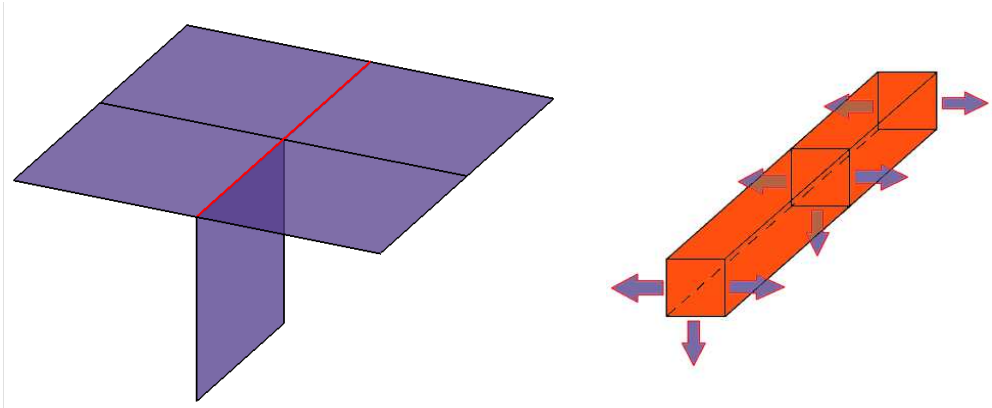
Durch die Beschränkung auf ein rein reguläres Schnittstellennetz lassen sich in jedem Schnittstellen-Punkt maximal vier Flächen zu einer gemeinsamen Abbildungsfunktion vereinen. Abbildung 6.41(a) zeigt ein Beispiel für eine unzulässige Verbindungskante. Durch die Einführung von unstrukturierten Verbindungsnetzen könnten solche Anschlüsse jedoch konform vernetzt werden. Abbildung 6.41(b) zeigt eine mögliche Ausführung für ein entsprechendes Schnittstellennetz, es sind jedoch auch zahlreiche andere Formen vorstellbar. Alternativ könnte ein solches Anschluss-Probleme auch durch eine lokale Ummodellierung und eine aneinander Reihung von regulären Schnittstellen gelöst werden.

Eine weitere Einschränkung der aktuellen Implementierung ist der Übergang von 1D- auf 2D-Verbindungskanten (vgl. Abbildung 6.42(a)). Durch die Einführung des zweidimensionalen Schnittstellennetzes aus Viereckselementen zur Verbindung von nur *quasi-parallelen* Mittelflächen können beim Übergang von Vierecks- auf Hexaeder-Schnittstellen Lücken entstehen. Diese müssen aktuell mit prismatischen Elementen gefüllt werden. Um solche Übergangsstellen konform mit nur Hexaeder-Netzen auflösen zu können, könnte beispielsweise ein reines Hexaeder Schnittstellen-Netz eine geeignete Lösung liefern (vgl. Abbildung 6.42(b)).



(a) Verbindungskante mit acht Flächen (b) Mögliches Schnittstellennetz

Abbildung 6.41: Unzulässige Verbindungskante



(a) Übergang von 1D auf 2D Verbindungskante (b) Mögliches Schnittstellennetz

Abbildung 6.42: Unzulässiger Schnittstellen-Übergang

Kapitel 7

Zusammenfassung und Ausblick

Die Methode der finiten Elemente ist das seit Jahren wichtigste und am häufigsten eingesetzte Werkzeug zur Berechnung verschiedenster physikalischer Phänome aus den unterschiedlichsten Ingenieursdisziplinen. Ausgangspunkt einer jeden Finite-Elemente-Berechnung ist ein Netz, bestehend aus einer endlichen Zahl von Elementen, welche das Berechnungsgebiet näherungsweise beschreiben. Die Elemente selber können dabei von unterschiedlicher Dimension, unterschiedlichem Elementtyp und unterschiedlicher Ordnung sein. Um qualitativ gute Berechnungsergebnisse mit angemessenem Zeitaufwand zu erzielen, bestehen je nach Finite-Elemente-Ansatz unterschiedliche Anforderungen an die Form und Ausrichtung der einzelnen Elemente [9, 13, 80, 118]. Eine der Hauptschwierigkeiten der Finite-Elemente-Analyse ist die vollautomatische Generierung eines geeigneten Finite-Elemente-Netzes für beliebig geformte Berechnungsgebiete. Das erklärte Ziel dieser Arbeit ist die Untersuchung der Netzgenerierungsproblematik, sowie die Entwicklung von Diskretisierungsmethoden zur Generierung von Finite-Elemente-Netzen, welche das Berechnungsgebiet mit gekrümmten Elementen beschreiben, um für Finite-Elemente-Ansätze hoher Ordnung [14, 15, 40, 81] geeignet zu sein.

- Üblicherweise werden die geometrischen Modelle der zu simulierenden Strukturen mit Hilfe von computergestützten Konstruktionsprogrammen erzeugt. Diesen Programmen liegen unterschiedliche Modellierungskonzepte zu Grunde, mit welchen komplexe zwei- oder dreidimensionale Gebiete einfach und effektiv beschrieben werden. In Kapitel 2 wurden die wichtigsten Modellierungskonzepte zur Beschreibung unterschiedlicher geometrischer Modelle erläutert. Zudem wurde ein Einblick in die wichtigsten differentialgeometrischen Eigenschaften der zu beschreibenden Modelle gegeben, sowie verschiedene topologische Aspekte aufgezeigt.
- Die Finite-Elemente-Methode kann zur Lösung von Randwertproblemen auf unterschiedliche Art und Weise formuliert werden. Neben den klas-

sischen Ansätzen, deren Genauigkeit hauptsächlich von der Feinheit des Finite-Elemente-Netzes abhängt, existieren Ansätze hoher Ordnung, welche insbesondere bei Problemen mit glatten Lösungen schon auf sehr groben Netzen mit nur wenig Elementen hochwertige Ergebnisse liefern [40, 81]. Kapitel 3 widmet sich der allgemeinen Beschreibung eines Randwertproblems, sowie der Einführung der h- und der p-Version der Finite-Elemente-Methode zur Berechnung seiner schwachen Form.

- Der Schlüssel zur Beschreibung von komplexen geometrischen Modellen für eine Vernetzung in Finite-Elemente ist eine leistungsfähige Datenbasis. Diese muss neben einem schnellen Zugriff auf das Modell und seine Entitäten eine detaillierte geometrische Modellbeschreibung inklusive geometrischen Operationen wie Verschneidung, Abbildung, etc. bereit halten. Komplexe Ingenieursanwendung besitzen aufbauend auf einer solchen Datenbasis in der Regel Module zur Visualisierung, Bearbeitung und Simulation der geometrischen Daten. $TUM.GeoFrame$ ist eine Rahmen-Software, welche im Zuge dieser Arbeit entwickelt wurde, um als Fundament für die entwickelten Algorithmen zu dienen. In Kapitel 4 wurden der Aufbau und die Funktionsweise von $TUM.GeoFrame$ erklärt, sowie das Zusammenspiel der einzelnen Programmmodule innerhalb des Rahmenwerks erläutert.

- Die Gebietsteilungstechnik ist ein Verfahren zur Generierung von Finite-Elemente-Netzen auf beliebigen zweidimensionalen Berechnungsgebieten. Besteht das Berechnungsgebiet aus gekrümmten Freiformflächen, so wird die Parameterebene der einzelnen Flächen für die Vernetzung angesetzt. Um exakte Längen- und Winkelberechnungen auf der Parameterebene durchführen zu können, werden die differentialgeometrischen Eigenschaften der einzelnen Flächen ausgenutzt. In Kapitel 5 wurden grundlegende Techniken zur Generierung von Oberflächennetzen aufgezeigt. Zudem wurde die Funktionsweise des Netzgenerators $DoMesh$ erläutert, der im Rahmen dieser Arbeit erweitert wurde, um auf beliebig geformten Freiformflächen qualitativ hochwertige Elemente zu generieren.

- Die Generierung von Finite-Elemente-Netzen auf dreidimensionalen Berechnungsgebieten ist im Gegensatz zur Vernetzung von zweidimensionalen Flächen mit ungleich höherem Aufwand verbunden. Neben dem zugrundeliegenden geometrischen Modell hat der zu generierende Elementtyp einen entscheidenden Einfluss auf die Schwierigkeit dieser Aufgabenstellung. Die klassischen Tetraeder-Elemente sind die am einfachsten zu generierenden Elementtypen und lassen sich auf so gut wie jedem geometrischen Modell erzeugen. Deutlich schwieriger ist die Generierung von Hexaeder-Elementen, für die in der Literatur noch keine allgemeingültige Lösung bekannt ist. Allerdings besitzen Hexaeder-Elemente gegenüber Tetraedern mathematische Vorteile. In Kapitel 6 wurde zunächst eine Strategie zur Generierung von Tetraeder-Elementen auf beliebigen dreidimensionalen Modellen aufgezeigt, welche für eine Berechnung mit der p-Version der Finite-Elemente-Methode geeignet sind. Anschließend wurde eine neue Methode zur Generierung von Hexaeder-Netzen auf dimensionsreduzierten Flächenmodellen gezeigt. Diese

Methode ist in der Lage, eine Vielzahl von komplexen, schalenartigen Strukturen für eine Berechnung mit der p-Version der Finite-Elemente-Methode in Hexaeder-Elemente zu vernetzen, für die nach Kenntnis des Autors bis heute keine geeignete Vernetzungsmethoden bekannt waren. Das gezeigte Verfahren profitiert dabei von einem eigens entwickelten geometrischen Flächenmodell, welches die Schwierigkeiten der Dimensionsreduzierung, die den bekannten Extrusionsverfahren üblicherweise anhaften, bereits auf der Modellebene umgeht. Die in dieser Arbeit vorgeschlagene Herangehensweise, schalenartige Strukturen für eine 3D Vernetzung über Mittel- und Referenzflächen zu modellieren, wie bei dimensionsreduzierten Berechnungsmethoden ohnehin schon üblich, ist nach Kenntnis des Autors bis zum heutigen Zeitpunkt einzigartig.

Der im Rahmen dieser Arbeit entwickelte Algorithmus zur Generierung von Hexaeder-Netzen ist nur für eine beschränkte Menge von geometrischen Modellen anwendbar. Eine allgemeingültige Methode zur Generierung von qualitativ hochwertigen Hexaeder-Elementen auf beliebigen dreidimensionalen Berechnungsgebieten bleibt nachwievor ein ungelöstes Problem und Ziel vieler weiterer Forschungsvorhaben. Die gezeigte Methode ist nach Kenntnis des Autors jedoch das einzige bekannte Verfahren, welches Hexaeder-Netze auf dimensionsreduzierten Flächenmodellen generiert, die für eine Berechnung mit der p-Version der Finite-Elemente-Methode geeignet sind.

Der Netzgenerierungsalgorithmus wurde prototypisch implementiert und birgt in seiner aktuellen Form zahlreiche weiterführende Fragen. So lässt sich die Komplexität der vernetzbaren Strukturen durch den Ausbau des geometrischen Modells, sowie durch eine Erweiterung der Kombinationsmöglichkeiten verschiedener sich schneidender Flächen weiter steigern.

Darüber hinaus besteht die Frage einer möglichen Anbindung an Netzgenerierungstechniken zur Vernetzung von beliebigen dreidimensionalen Berechnungsgebieten. Eine Methode zur vollautomatischen Generierung von zusammenhängenden Hexaeder-Netzen auf gemischten Strukturen bestehen aus dünnwandigen Schalen und dicken Volumenkörpern, so wie sie in vielen strukturmechanischen System vorkommen, ist in der Literatur bisher noch nicht bekannt.

Neben einer Berechnung auf zusammenhängenden Berechnungsgebieten werden in der Praxis häufig Verfahren zur Kopplung sich berührender aber nicht zusammen fallender, also nicht konformer Finite-Elemente-Netze angesetzt [20, 43, 22, 113]. Werden solche Kontaktstellen mit der p-Version der Finite-Elemente-Methode berechnet, können die Kontaktflächen beliebig gekrümmt sein. Zur Optimierung der Rechengenauigkeit ist es in solchen Fällen wichtig, den geometrischen Fehler entlang der Kontaktflächen durch eine geeignete Diskretisierung zu minimieren. Die Generierung von nicht konformen, zusammenfallenden Netzen mit gekrümmten p-Elementen von unterschiedlichem Elementtyp auf Basis der gezeigten Algorithmen steht daher im Blickfeld weiterführender Forschungsvorhaben.

Literaturverzeichnis

[1] BRep - File Format Specification. Posted on www at http://www.cs.princeton.edu/courses/archive/spr98/cs598d/programs/bsps.html.

[2] IGES - File Format Specification. Posted on www at http://ts.nist.gov/standards/iges/.

[3] STEP standarization: ISO 10303. Posted on www at http://en.wikipedia.org/wiki/ISO 10303-21.

[4] STL - File Format Specifiation. Posted on www at http://www.rapidtoday.com/stl-file-format.html.

[5] VRML - File Format Specification. Posted on www at http://graphcomp.com/info/specs/sgi/vrml/spec/.

[6] Abramowitz, M. & Stegun, I. (1964). *Handbook of Mathematical Functions*. Washington, D.C.: National Bureau of Standards, Applied Mathematics Series 55.

[7] Abramowski, S. & Müller, H. (1991). *Geometrisches Modellieren*. BI Wissenschaftsverlag.

[8] Agoston, M. K. (2005). *Computer Graphics and Geometric Modeling - Implementation and Algorithms*. Springer-Verlag.

[9] Ainsworth, M. & Oden, J. (2000). *A posteriori error estimation in finite element analysis*. John Wiley & Sons.

[10] Armstrong, C., McKeag, R., Ou, H. & Price, M. (2000). Geometric processing for analysis. In: *Proceedings, Geometric modeling and processing 2000. Theory and Applications*, S. 45–56.

[11] Armstrong, C., Robinson, T. & Ou, H. (2008). Recent advances in CAD/-CAE technologies for thin-walled structures design and analysis. In: *Proceedings, 5th International Conference on Thin-Walled Structures*, S. 251–268.

[12] Babuška, I. & Chen, Q. (1995). Approximate optimal points for polynomial interpolation of real functions in an interval and in a triangle. *Computer Methods in Applied Mechanics and Engineering* 128, S. 405–417.

[13] Babuška, I. & Rheinboldt, W. (1978). A posteriori error estimates for the finite element method. 12, S. 1597–1615.

[14] Babuška, I. & Suri, M. (1987). The optimal convergence rate of the p-version of the finite element method. 24, S. 750–776.

[15] Babuška, I., Szabo, B. A. & Katz, I. N. (1981). The *p*-Version of the Finite Element Method. 18, S. 515–545.

[16] Baehmann, P. L., Wittchen, S. L., Shephard, M. S., Grice, K. R. & Yerry, M. A. (1987). Robust Geometrically-based, Automatic Two-Dimensional Mesh Generation. *International Journal for Numerical Methods in Engineering* 24, S. 1043–1078.

[17] Baker, T. J. (1989). Automatic Mesh Generation for Complex Three-Dimensional Regions Using a Constrained Delaunay Triangulation. *Engineering with Computers* 5, S. 161–175.

[18] Bank, R. E. (1990). PLTMG, A software package for solving elliptic partial differential equations. In: *(SIAM Frontiers in Applied Mathematics)*, Volume 7. Society for Industrial & Applied Mathematics.

[19] Bathe, K. (2002). *Finite-Elemente-Methoden*. Springer-Verlag.

[20] Belgacem, F. & Maday, Y. (1997). The mortar element method for three-dimensional finite elements. *RAIRO Modél. Math. Anal* 2(31), S. 289–302.

[21] Benzley, S. E., Perry, E., Merkley, K. & Clark, B. (1995). A Comparison of All Hexagonal and All Tetrahedral Finite Element Meshes for Elastic and Elasto-plastic Analysis. In: *Proceedings of the 4th International Meshing Roundtable*.

[22] Bernardi, C., Maday, Y. & Patera, A. (1994). A New Non Conforming Approach to Domain Decomposition: The Mortar Element Method. In: H. Brezis & J.-L. Lions (Hrsg.), *Collège de France Seminar*. Pitman. Dieser Artikel erschien als technischer Report fünf Jahre früher.

[23] Blacker, T. D. (1996). The Cooper Tool. In: *Proceedings of the 5th International Meshing Roundtable*, S. 13–29.

[24] Blacker, T. D. (2001). Automated conformal hexahedral meshing constraints, challenges and opportunities. *Engineering with Computers* 17, S. 201–210.

[25] Blacker, T. D. & Meyers, R. J. (1993). Seams and wedges in plastering: A 3-D hexahedral mesh generation algorithm. *Engineering with Computers* 9, S. 83–93.

[26] Blacker, T. D. & Stephenson, M. B. (1991). Paving: A new approach to automated quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering* 32, S. 811–847.

[27] Bobenko, A. (2011). Differentialgeometrie von Kurven und Flüchen. `http://page.math.tu-berlin.de/~bobenko/Lehre/Skripte/KuF.pdf`.

[28] Borouchaki, H., Hecht, F., Saltel, E. & George, P. L. (1995, Oktober). Reasonably Efficient Delaunay Based Mesh Generator in 3 Dimensions. In: *4th International Meshing Roundtable*, S. 3–14.

[29] Buchele, S. F. & Crawford, R. H. (2004). Three-dimensional halfspace constructive solid geometry tree construction from implicit boundary representations. *Computer-Aided Design* 36, S. 1063–1073.

[30] Burger, M. (2009). Finite element approximation of elliptic partial differential equations on implicit surfaces. *Computing and Visualization in Science* 12, S. 87–100. 10.1007/s00791-007-0081-x.

[31] Carmo do, M. P. (1976). *Differential Geometry of Curves and Surfaces*. Prentice-Hall, Englewood Cliffs, N.J.

[32] Cass, R. J., Benzley, S. E., Meyers, R. J. & Blacker, T. D. (1996). Generalized 3-D Paving: An Automated Quadrilateral Surface Mesh Generation Algorithm. *International Journal for Numerical Methods in Engineering* 39, S. 1475–1489.

[33] Chae, S.-W. & Jeong, J.-H. (1997). Unstructured Surface Meshing Using Operators. In: *6th International Meshing Roundtable*, S. 281–291.

[34] Chen, S. & Doolen, G. D. (1998). Lattice Boltzmann method for fluid flows. *Annual Review of Fluid Mechanics* 30(1), S. 329–364.

[35] Chew, P. L. (1989, April). Guaranteed-Quality Triangular Meshes. *TR 89-983, Department of Computer Science, Cornell University, Ithaca, NY*.

[36] Cifuentes, A. & Kalbag, A. (1992). A performance study of tetrahedral and hexahedralelements in 3-D finite element structural analysis. *Finite Elements in Analysis and Design* 12, S. 313–318.

[37] Delauney, B. N. (1934). Sur la Sphere. *Vide. Izvestia Akademia Nauk SSSR* 7, S. 793–800. VII Seria, Otdelenie Matematicheskii i Estestvennyka Nauk.

[38] Dey, T. K. & Zhao, W. (2004). Approximate medial axis as a Voronoi subcomplex. *Computer Aided Design* 36, S. 195–202.

[39] Du, Q. & Wang, D. (2003). Tetrahedral mesh generation and optimization based on centroidal Voronoi tessellations. *International Journal for Numerical Methods in Engineering* 56, S. 1355–1373.

[40] Düster, A., Bröker, H. & Rank, E. (2001). The p-version of the finite element method for three-dimensional curved thin walled structures. 52, S. 673–703.

[41] Edgel, J. (2010). An Adaptive Grid-Based All Hexahedral Meshing Algorithm Based on 2-Refinement. Master's Thesis, Brigham Young University.

[42] Eschauer, H. & Schnell, W. (1993). *Elastizitätstheorie*. BI Wissenschaftsverlag.

[43] F.B., B. (1999). The Mortar Finite Element Method with Lagrange Multipliers. 84, S. 173–197.

[44] Freitag, L. A. & Ollivier-Gooch, C. (1998). Tetrahedral mesh improvement using swapping and smoothing. *International Journal for Numerical Methods in Engineering* 40, S. 3979–4002.

[45] Frischmann, F. (2011). Topological and Geometric Healing on Solid Models. Master's Thesis, Technische Universität München.

[46] George, P., Hecht, F. & Saltel, E. (1991). Automatic Mesh Generator with Specified Boundary. *Computer Methods in Applied Mechanics and Engineering* 92, S. 269–288.

[47] Goldman, R. (2009). *An Integrated Introduction to Computer Graphics and Geometric Modeling*. CRC Press, Inc. Boca Raton, FL, USA.

[48] Gomes, A., Voiculescu, I., Jorge, J., Wyvill, B. & Galbraith, C. (2009). *Implicit Curves and Surfaces: Mathematics, Data Structures and Algorithms*. Springer Verlag.

[49] Gordon, W. & Hall, C. (1973). Transfinite element methods: blending function interpolation over arbitrary curved element domains. *Numerische Mathematik* 21, S. 109–129.

[50] Gründemann, H. (2011). Grundlagen der Differentialgeometrie. `http://www.staff.hs-mittweida.de/~hgruende/Files/diffgeo.pdf`.

[51] Hu, S.-M., Wang, G.-Z. & Jin, T.-G. (1996). Properties of two types of generalized ball curves. *Computer-Aided Design* 28, S. 125–133.

[52] Hughes, T., Cottrell, J. & Bazilevs, Y. (2005). Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering* 194, S. 4135–4195.

[53] Ito, Y., Shih, A. M. & Soni, B. K. (2009). Octree-based reasonable-quality hexahedral mesh generation using a new set of refinement templates. *International Journal for Numerical Methods in Engineering* 77, S. 1809–1833.

[54] Johnston, B. P., Sullivan, J. M. J. & Kwasnik, A. (1991). Automatic Conversion of Triangular Finite Element Meshes to Quadrilateral Elements. *International Journal for Numerical Methods in Engineering* 31, S. 67–84.

[55] Kiràlyfalvi, G. & Szabò, B. (1997). Quasi-regional mapping for the p-version of the finite element method. *Finite elements in analysis and design* 27(1), S. 85–97.

[56] Knupp, P. (1998). Next-Generation Sweep Tool: A Method for Generation All-Hex Meshes on Two-and-One-Half Dimensional Geometries. In: *Proceedings of the 7th International Meshing Roundtable*, S. 505–513.

[57] Kwon, K., Lee, B. & Chae, S. (2006). Medial surface generation using chordal axis transformation in shell structures. *Computers & Structures* 84, S. 1673–1683.

[58] Labelle, F. & Shewchuk, J. (2003). Anisotropic voronoi diagrams and guaranteed-quality anisotropic mesh generation. In: *Proceedings of the nineteenth annual symposium on Computational geometry*, S. 191–200.

[59] Lawson, C. L. (1977). Software for C^1 Surface Interpolation. *Mathematical Software III*, S. 161–194.

[60] Lee, C. & Lo, S. (1994). A New Scheme for the Generation of a Graded Quadrilateral Mesh. *Computers and Structures* 52, S. 847–857.

[61] Löhner, R. (1996). Interactive Generation of Unstructured Grid for Three Dimensional Problems. *Engineering with Computers* 12, S. 186–210.

[62] Löhner, R. & Parikh, P. (1988). Generation of three-dimensional unstructured grids by the advancing-front method. *Journal for Numerical Methods in Fluids* 8, S. 1135–1149.

[63] Löhner, R., Parikh, P. & Gumbert, C. (1988). Interactive Generation of Unstructured Grid for Three Dimensional Problems. *Numerical Grid Generation in Computational Fluid Mechanics*, S. 687–697.

[64] Manson, S. (2008, January). Automated translation of high order complex geometry from a CAD model into a surface based combinatorial geometry format.

[65] Marcum, D. L. & Weatherill, N. P. (1995, September). Unstructured Grid Generation Using Iterative Point Insertion and Local Reconnection. *AIAA Journal* 33(9), S. 1619–1625.

[66] Mingwu, L., Benzley, S. E. & D.R., W. (2000). Automated hexahedral mesh generation by generalized multiple source to multiple target sweeping. *International Journal for Numerical Methods in Engineering* 49, S. 261–275.

[67] Müller-Hannemann, M. (1999). Hexahedral Mesh Generation by Successive Dual Cycle Elimination. *Engineering with Computers* 15, S. 269–279.

[68] Murdoch, P., Benzley, S., Blacker, T. D. & Mitchell, S. (1997). The spatial twist continuum: a connectivity based method for representing all-hexahedral finite element meshes. *Finite elements in analysis and design* 28, S. 137–149.

[69] Noor, A. K. & Babuška, I. (1987). Quality assessment and control of finite element solutions. *Finite Elements in Analysis and Design* 3, S. 1–26.

[70] Nowottny, D. (1997). Quadrilateral Mesh Generation via Geometrically Optimized Domain Decomposition. In: *6th International Meshing Roundtable*, S. 309–320.

[71] O.C.C. (2012). OPEN CASCADE S.A.S. Open CASCADE Technology - http://www.opencascade.org/.

[72] Owen, S. J. (1998). A survey of unstructured mesh generation technology. In: *Proceedings of the 7th International Meshing Roundtable*, S. 239–267.

[73] Owen, S. J. & Saigal, S. (2000). H-Morph: an indirect approach to advancing front hex meshing. *International Journal for Numerical Methods in Engineering* 49, S. 289–312.

[74] Owen, S. J., Staten, M. L., Canann, S. A. & Saigal, S. (1992). Three-dimensional boundary-constrained triangulations. In: E. N. Houstis & J. R. Rice (Hrsg.), *13th IMACS World Congress*, S. 215–222. Elsevier Science Publishers. Artificial Intelligence, Expert Systems, and Symbolic Computing.

[75] Owen, S. J., Staten, M. L., Canann, S. A. & Saigal, S. (1998). Advancing Front Quad Meshing Using Local Triangle Transformations. In: *7th International Meshing Roundtable*.

[76] Parvizian, J., Düster, A. & Rank, E. (2007). Finite cell method. *Computational Mechanics* 41, S. 121–133.

[77] Paudel, G., Owen, S. J. & Benzley, S. E. (2012). Hexahedral Mesh Refinement Using an Error Sizing Function. In: W. R. Quadros (Hrsg.), *Proceedings of the 20th International Meshing Roundtable*, S. 143–159. Springer Berlin Heidelberg.

[78] Pirzadeh, S. (1993). Unstructured Viscous Grid Generation by Advancing-Layers Method. *AIAA Journal 3453-CP* 34, S. 420–434.

[79] Q.T. (2012). Nokia. Qt - http://qt.nokia.com/.

[80] Rank, E. (1992). Adaptivity and accuracy estimation for FEM and BIEM. In: Brebbia & Aliabadi (Hrsg.), *Adaptive Meshing and Error Estimates*, Chapter 1. Elsevier.

[81] Rank, E., Bröker, H., Düster, A. & Rücker, M. (2000). Modellieren und Berechnen mit der p-version der FEM. In: *SOFiSTiK 98 und 99*, Balkema, Rotterdam, S. 185–192.

[82] Rank, E., Rücker, M. & Schweingruber, M. (1994). Automatische Generierung von Finite-Element-Netzen. 69(10), S. 373–379.

[83] Rank, E., Schweingruber, M. & Sommer, M. (1993). Adaptive mesh generation and transformation of triangular to quadrilateral meshes. *Communications in Numerical Methods in Engineering* 9, S. 121–129.

[84] Rebay, S. (1993). Efficient Unstructured Mesh Generation by Means of Delaunay Triangulation and Bowyer-Watson Algorithm. *Journal Of Computational Physics* 106, S. 125–138.

[85] Ruppert, J. (1992). A New and Simple Algorithm for Quality 2-Dimensional Mesh Generation. *Technical Report UCB/CSD 92/694, University of California at Berkely.*

[86] Rvachev, I., Shapiro, V., Sheiko, T. & Tsukanov, I. (2000). On Completeness of RFM Solution Structures. *Computational Mechanics, special issue on meshfree methods* 25, S. 305–316.

[87] Schneiders, R. (1996). A grid-based algorithm for the generation of hexahedral element meshes. *Engineering with Computers* 12, S. 168–177.

[88] Schneiders, R., Schindler, R. & Weiler, R. (1996). Octree-based generation of hexahedral element meshes. In: *Proceedings of the 5th International Meshing Roundtable*, S. 205–216.

[89] Schöberl, J. (1997). NETGEN An advancing front 2D/3D-mesh generator based on abstract rules. *Computing and Visualization in Science* 1, S. 41–52. 10.1007/s007910050004.

[90] Schöberl, J. (2012). Netgen: An automatic 3d tetrahedral mesh generator. http://sourceforge.net/projects/netgen-mesher/.

[91] Schweingruber-Straten, M. (1999). *Generierung von Oberflächennetzen nach der Gebietsteilungstechnik.* Dissertation, Fach Numerische Methoden und Informationsverarbeitung, Universität Dortmund.

[92] Shephard, M. S. & Georges, M. K. (1991). Three-Dimensional Mesh Generation by Finite Octree Technique. *International Journal for Numerical Methods in Engineering* 32, S. 709–749.

[93] Shepherd, J. & Johnson, C. (2008). Hexahedral mesh generation constraints. *Engineering with Computers* 24, S. 195–213. 10.1007/s00366-008-0091-4.

[94] Sherbrooke, E., Patrikalakis, N. & Brisson, E. (1996). An algorithm for the medial axis transform of 3D polyhedral solids. *Visualization and Computer Graphics* 2, S. 44–61.

[95] Shewchuk, J. R. (1996). Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. *http://www.cs.cmu.edu/ quake/triangle.html.*

[96] Sorger, C., Tsukanov, I. & Rank, E. (2012). A generic framework for embedded domain methods discretizing B-rep models. In: *European Congress on Computational Methods in Applied Sciences and Engineering*, Wien, Österreich.

[97] Staten, M., Canann, S. & Owen, S. J. (1999). BMSweep: Locating Interior Nodes During Sweeping. *Engineering with Computers* 15, S. 212–218.

[98] Staten, M., Owen, S. J. & Blacker, T. D. (2005). Unconstrained Paving & Plastering: A New Idea for All Hexahedral Mesh Generation. In: B. W. Hanks (Hrsg.), *Proceedings of the 14th International Meshing Roundtable*, S. 399–416. Springer Berlin Heidelberg.

[99] Staten, M. L., Kerr, R. A., Owen, S. J., Blacker, T. D., Stupazzini, M. & Shimada, K. (2010). Unconstrained plastering–Hexahedral mesh generation via advancing-front geometry decomposition. *International Journal for Numerical Methods in Engineering* 81, S. 135–171.

[100] Szabó, B. & Babuška, I. (1991). *Finite element analysis*. John Wiley & Sons.

[101] Szabo, B. & Schnell, W. (1984). Estimation and Control Error Based on P-Convergence. *Accuracy estimates and adaptive refinements in finite element computations*, S. 61–70.

[102] Talbert, J. & Parkinson, A. (1991). Development of an Automatic, Two Dimensional Finite Element Mesh Generator using Quadrilateral Elements and Bezier Curve Boundary Definitions. *International Journal for Numerical Methods in Engineering* 29, S. 1551–1567.

[103] Tam, T. K. H. & Armstrong, C. G. (1991). 2D Finite Element Mesh Generation by Medial Axis Subdivision. *Advances in Engineering Software* 13, S. 313–324.

[104] Tautges, T., Blacker, T. D. & Mitchell, S. (1996). The whisker weaving algorithm: a connectivity-based method for constructing all-hexahedral finite element meshes. *International Journal for Numerical Methods in Engineering* 39, S. 3327–3350.

[105] Thompson, J. F. (1985). Numerical Grid Generation, Foundation and Applications. In: *Posted on www at http://www.erc.msstate.edu/education/gridbook/index.html*.

[106] Thompson, J. F. (1996, April). A Reflection on Grid generation in the 90s: Trends Needs and Influences. In: *5th International Conference on Numerical Grid Generation in Computational Field Simulations*, Mississippi State University, USA.

[107] Tristano, J. R., Owen, S. J. & Canann, S. A. (1998). Advancing front surface mesh generation in parametric space using a Riemannian surface definition. In: *Proceedings of the 7th International Meshing Roundtable*.

[108] Veith, S. (2012). Entwicklung eines kompilierten Hilfemoduls für TUM.GeoFrame. Bachelor thesis, Technische Universität München.

[109] V.T.K. (2012). Kitware, Inc. VTK - Visualization Toolkit - http://www.vtk.org/.

[110] Watson, D. F. (1981). Computing the Delaunay Tesselation with Application to Voronoi Polytopes. *The Computer Journal* 24(2), S. 167–172.

[111] Weatherill, N. P. & Hassan, O. (1994). Efficient Three-dimensional Delaunay Triangulation with Automatic Point Creation and Imposed Boundary Constraints. *International Journal for Numerical Methods in Engineering* 37, S. 2005–2039.

[112] White, D. R. & Kinney, P. (1997). Redesign of the Paving Algorithm: Robustness Enhancements through Element by Element Meshing. In: *6th International Meshing Roundtable*, S. 323–335.

[113] Wohlmuth, B. (2000). A mortar finite element method using dual spaces for the Lagrange multiplier. 38, S. 989–1012.

[114] Yerry, M. A. & Shephard, M. S. (1984). Three-Dimensional Mesh Generation by Modified Octree Technique. *International Journal for Numerical Methods in Engineering* 20, S. 1965–1990.

[115] Zhang, H. & Zhao, G. (2007). Adaptive hexahedral mesh generation based on local domain curvature and thickness using a modified grid-based method. *Finite Elements in Analysis and Design* 43, S. 691–704.

[116] Zhu, J., Zienkiewicz, O., Hinton, E. & Wu, J. (1991). A New Approach to the Development of Automatic Quadrilateral Mesh Generation. *International Journal for Numerical Methods in Engineering* 32, S. 849–866.

[117] Zienkiewicz, O. & Taylor, R. (2000). *The Finite Element Method - Basic Formulations and Lienar Problems*. volume 2. McGraw-Hill Book Company, 5. Auflage.

[118] Zienkiewicz, O. & Zhu, J. (1987). A simple error estimator and adaptive procedure for practical engineering analysis. 24, S. 337–357.