



TECHNISCHE  
UNIVERSITÄT  
MÜNCHEN



**Angewandte  
Mechanik**

Prof. Dr.-Ing. habil.  
Heinz Ulbrich

## Diplomarbeit

# Virtual-Reality Methoden für ein MKS-Kontaktmodell

Verfasser: Gonjun Hong

Matrikelnr.: 1927440

Betreuer: Dipl.-Ing. Sandor Riebe, Technische Universität München

Dipl.-Ing. Gerhard Hippmann, Deutsches Zentrum für Luft- und Raumfahrt e.V.

Angefertigt in der Abteilung Fahrzeug-Systemdynamik, Institut für Aeroelastik, DLR

Garching, den 30. April 2003

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>3</b>
1.1	Kontaktmechanik in der Mehrkörperdynamik . . . . .	3
1.2	Polygonal Contact Model . . . . .	7
1.3	Aufgabenstellung . . . . .	8
<b>2</b>	<b>Grundlagen der Flächendarstellung</b>	<b>9</b>
2.1	Implizite und parametrische Darstellung . . . . .	9
2.1.1	Implizite Flächen . . . . .	9
2.1.2	Parametrische Flächen . . . . .	10
2.1.3	Vergleich parametrischer und impliziter Flächen . . . . .	10
2.2	Implizite Feldfunktionen . . . . .	11
2.2.1	Klassifizierung . . . . .	11
2.2.2	Kombination . . . . .	12
2.3	Freiformflächen . . . . .	12
2.4	Variational Interpolation . . . . .	14
2.4.1	Variationsrechnung . . . . .	14
2.4.2	Interpolation mit radialen Basisfunktionen . . . . .	15
<b>3</b>	<b>Approximation der Kontaktfläche</b>	<b>17</b>
3.1	Interpolation der unverformten Schnittfläche . . . . .	17
3.1.1	Interpolationsbedingungen . . . . .	18
3.1.2	Konstruktion der Hilfspunkte . . . . .	18
3.1.3	Einfluß der Hilfspunkte auf die Interpolation . . . . .	18
3.2	Überblendung zwischen den Kontaktflächen . . . . .	19
3.2.1	Approximation der Kontaktfläche . . . . .	19
3.2.2	Einfluss der Hilfspunkte auf die Überblendung . . . . .	20
3.3	Alternative Approximation der Konktaktfläche . . . . .	21
3.3.1	Interpolation . . . . .	21

3.3.2	Überblendung . . . . .	22
3.3.3	Praxistests . . . . .	22
<b>4</b>	<b>Diskretisierung der Kontaktfläche</b>	<b>25</b>
4.1	Triangulierung im Überblick . . . . .	25
4.1.1	Marching Cubes . . . . .	26
4.1.2	Marching Tetrahedra . . . . .	28
4.1.3	Marching Triangles . . . . .	29
4.2	Triangulierung der Kontaktfläche . . . . .	31
4.2.1	Grundidee . . . . .	31
4.2.2	Bestimmung von Flächenpunkten . . . . .	32
4.2.3	Implementierung . . . . .	33
4.2.3.1	Datenstruktur . . . . .	33
4.2.3.2	Algorithmus . . . . .	33
4.2.4	Bewertung . . . . .	38
4.3	Konstruktion der Kontaktelemente . . . . .	40
<b>5</b>	<b>Anwendung</b>	<b>43</b>
<b>6</b>	<b>Zusammenfassung</b>	<b>57</b>
<b>A</b>	<b>Numerische Grundlagen</b>	<b>59</b>
A.1	Matrizen . . . . .	59
A.1.1	Symmetrische Matrix . . . . .	59
A.1.2	Positiv/negativ definite Matrix . . . . .	59
A.2	Lösung symmetrischer, semipositiver Linearsysteme . . . . .	60
A.2.1	LU-Zerlegung mit Elimination nach Gauß . . . . .	61
A.2.2	Rücksubstitution . . . . .	62
A.3	Newton-Verfahren . . . . .	62
A.4	Koordinatentransformation . . . . .	63
A.5	Polarwinkel . . . . .	64
<b>B</b>	<b>Routinen</b>	<b>65</b>
B.1	pcm_crcevipo.c . . . . .	65
B.2	pcm_vipofun.c . . . . .	65
B.3	pcm_vipotri.c . . . . .	65
B.4	pcm_vipomath.c . . . . .	66

# Kapitel 1

## Einleitung

### 1.1 Kontaktmechanik in der Mehrkörperdynamik

In allen Phasen des Entwicklungsprozesses (Concurrent Engineering) finden moderne, rechnergestützte Entwicklungswerkzeuge Verwendung. So liefern in der Konstruktionsphase CAD-Systeme Fertigungsunterlagen der Einzelkomponenten, und gleichzeitig ermöglicht die numerische Simulation von Mehrkörpersystemen (MKS) bereits einen ersten Überblick über das Zusammenwirken der Einzelteile. Simulationsprogramme erlauben eine Überprüfung bzw. Erprobung der Konstruktion im Rechner, lange bevor das erste Versuchsexemplar gebaut ist.

Modelldarstellung und Auswertungsmethoden etablierter Simulationsprogramme sind auf die Bedürfnisse von Berechnungsingenieuren zugeschnitten, und für bedeutende Spezialanwendungen wie Straßen- oder Schienenfahrzeuge stehen ausgereifte Bibliotheken und Module zur Verfügung.

In zahlreichen Anwendungen haben Berührungen von Körpern essenziellen Einfluss auf das Systemverhalten. Beispielsweise bestimmen sie ganz wesentlich die Eigenschaften von Wälzlagern, Kurvengetrieben, greifenden Robotern und Laufmaschinen. Ein grundlegendes Problem bei der MKS-Simulation von Fahrzeugen ist die Darstellung der Traktion zwischen Rädern und Lauffläche. Und auch in der Biomechanik treten häufig Berührungen auf, etwa in Gelenken oder bei Unfällen. Für diese und viele weitere Anwendungen werden im MKS-Modell Elemente benötigt, welche die *Kontaktmechanik* geeignet darstellen.

Im ersten Schritt erfordert die mathematisch-physikalische Behandlung von Kontaktvorgängen eine Modellbildung der zu betrachtenden Vorgänge. Die dabei getroffenen Voraussetzungen und Annahmen sind bestimmt für den Gültigkeitsbereich und die Übertragbarkeit der gewonnenen Ergebnisse.

Zur Beschreibung von Kontaktvorgängen existieren sehr unterschiedliche Ansätze und Implementierungen, die sich erheblich hinsichtlich Spezialisierungsgrad, physikalischer Modellqualität, Komplexität, Effizienz und Robustheit unterscheiden.

#### **Eindimensionale Modelle:**

Bei eindimensionalen Modellen werden Kontaktvorgänge anhand eindimensionaler physikalischer Gesetzmäßigkeiten dargestellt. Voraussetzung für ihre Anwendung ist ein repräsentativer Angriffspunkt mit zugehöriger Wirkrichtung.

- **Kraftelemente**

Kraftelemente gehören neben Starrkörpern und Gelenken zu den MKS-Grundelementen. Sie werden als masselos angenommen und dienen dazu, eingeprägte Kräfte und Momente ins System einzubringen. Klassische Kraftelemente wirken zwischen zwei Koppelpunkten, die auf verschiedenen Körpern oder interialfest liegen können, in Abhängigkeit ihrer Relativkinematik. Zur Beschreibung von Kontakten werden häufig spezielle Kraftelemente verwendet, die in Normalenrichtung als einseitiges Feder-Dämpfer-Element wirken und in Tangentialrichtung ein regularisiertes Coulombsches Kraftgesetz einbringen [16].

- **Starrkörperkontakt**

Bei der am weitesten abstrahierenden Darstellung von Kontaktvorgängen wird von völlig starren Körpern ausgegangen. Daraus folgen mengenwertige Kraftgesetze, die durch „eckige“ Kennlinien charakterisiert sind. Weil geschlossene Kontakte als zusätzliche Bindungen wirken, erhält man ein strukturvariables System, welches bei der Implementierung zu deutlich erhöhtem Verwaltungsaufwand führt. Um elastisches Materialverhalten abbilden zu können, muss man auf Stoßhypothesen zurückgreifen. Darüber hinaus muss die numerische Intergration mit Hilfe von zustandsabhängigen Schaltfunktionen stets unterbrochen werden, wenn sich ein Kontaktzustand ändert; anschließend ist eine konsistente Kombination aller Kontaktzustände zu ermitteln [5].

### Oberflächenmodelle:

Oberflächenmodelle sind durch ihre Wirkungsweise entlang der Oberflächen der kontaktierenden Körper gekennzeichnet. Der Verformungszustand in ihrem Inneren wird dabei nicht näher betrachtet. Es wird von homogenen, isotropen, linear-elastischen Körpern ausgegangen, deren Materialverhalten allein durch zwei Stoffkonstanten, z. B. Schubmodul  $G$  und Querkontraktionszahl  $\nu$ , beschrieben wird. Bei der Kontaktanalyse verwendet man kombinierte Werte der beiden beteiligten Körper:

$$G^* = \frac{1}{\frac{1}{G_1} + \frac{1}{G_2}} \quad (1.1)$$

$$\nu^* = \left( \frac{\nu_1}{G_1} + \frac{\nu_2}{G_2} \right) G^* \quad (1.2)$$

Der Elastizitätsmodul lässt sich am einfachsten mit dem bekannten Zusammenhang  $E^* = 2G^*(1 + \nu^*)$  ermitteln.

- **Halbraumapproximation**

Die Halbraumapproximation [11, 12] ist anwendbar, wenn das Kontaktgebiet relativ klein gegenüber den charakteristischen Abmessungen der Körper ist. Dann lässt sich das elastische Feld im Nahbereich der Berührfläche in guter Näherung durch einen elastischen Halbraum (Bild 1.1) ersetzen. Man geht vom Abstandsprofil  $g(x, y)$  der unverformten Oberflächen in einem Koordinatensystem aus, dessen  $z$ -Achse normal

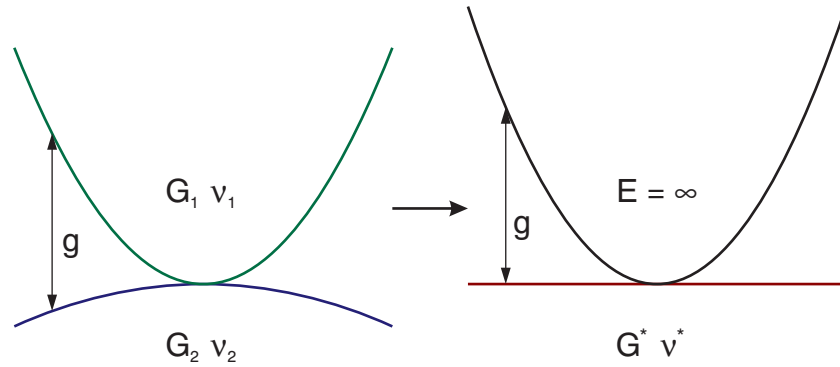


Bild 1.1: Übergang von der Kontaktpaarung zur Halbraumapproximation

zur angenommenen Berührebene verläuft. Im Weiteren wird ein Ersatzmodell betrachtet, in dem ein starrer Körper mit dem Abstandsprofil als Oberflächen in einen linear-elastischen Halbraum mit den kombinierten Stoffkonstanten (1.1), (1.2) eindringt.

Die Grundlage der Halbraumapproximation stellt die Gleichung

$$u_n(x, y) = \frac{1 - \nu}{2\pi G} \int_S \int \frac{p(x_p, y_p)}{r} dx_p dy_p \quad (1.3)$$

dar. Sie gibt die Einsinkung  $u_n$  des Halbraums aufgrund einer an der Oberfläche wirkenden Normalkraftverteilung  $p(x, y)$  entsprechend der Elastizitätstheorie an.  $r = \sqrt{(x - x_P)^2 + (y - y_P)^2}$  steht dabei für den Abstand zwischen dem betrachteten Ort und dem Angriffspunkt  $(x_P, y_P)$ .

Um den Verformungs- und Spannungszustand numerisch zu berechnen, wird das Kontaktgebiet durch Dreiecke oder Rechtecke diskretisiert, sodass (1.3) zu einer Matrixgleichung führt, die die Verformungen und Spannungen der Kontaktelemente über Einflusskoeffizienten verknüpft. Sie kann durch iterative Matrixinversion oder Variationsrechnung gelöst werden.

Mit der Halbraumapproximation lässt sich bei erhöhtem Aufwand zusätzlich das Coulombsche Reibungsmodell darstellen.

- **Randschichtmodell**

Ähnlich wie bei der Halbraumapproximation wird beim Randschichtmodell [11, 12] mit einem Ersatzmodell gearbeitet, bei dem ein starrer Körper mit dem aufsummierten Profil der Kontaktpaarung in eine elastische Ebene eindringt. Man geht jedoch nicht von einem elastischen, sondern ebenfalls starren Halbraum aus, der mit einer dünnen, linear-elastischen Randschicht überzogen ist. Eine erhebliche Vereinfachung ergibt sich dabei durch Vernachlässigung der Scherspannungen.

Anstelle einer Einflussfunktion (1.3), bei der die Verformung durch Integration über das gesamte Berührgebiet ermittelt werden muss, besteht ein linearer Zusammenhang zwischen Eindringung und Druckspannung am selben Ort:

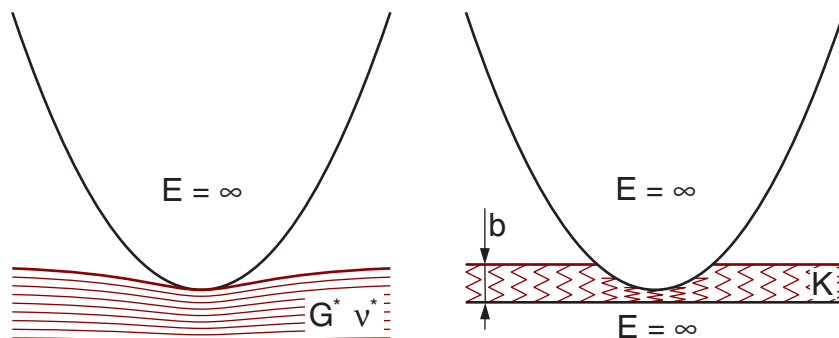


Bild 1.2: Vergleich von Halbraumapproximation und Randschichtmodell

$$u_n(x, y) = \frac{b}{K} \cdot p(x, y) \quad (1.4)$$

Dabei steht  $b$  für die Tiefe und

$$K = \frac{1 - \nu}{(1 + \nu)(1 - 2\nu)} \cdot E \quad (1.5)$$

für den elastischen Modul der Randschicht, die man sich vorstellen kann wie eine Matratze, deren Feldern unabhängig voneinander wirken (Bild 1.2).

Mit (1.4) lässt sich unmittelbar die Spannung aufgrund der Verformung, welche hier mit der Eindringung des unverformten Profils übereinstimmt, berechnen.

Das Randschichtmodell ist völlig analog auch in tangentialer Richtung anwendbar. Wie bei der Halbraumapproximation ist aber auch hier nicht klar, wie die Berührebene für beliebige Oberflächengeometrien zu definieren ist. Darüber hinaus muss der Faktor  $b/K$  durch Versuche oder Abschätzung bestimmt werden, sofern nicht tatsächlich eine dünne, relativ elastische Randschicht vorliegt.

### Festkörpermechanik:

- **Elastizitätstheorie**

Die klassische Aufgabenstellung bei der Anwendung der Elastizitätstheorie auf Kontaktprobleme ist die Bestimmung des Verzerrungs- und Spannungsfelds in einem elastischen Halbraum bei gegebenen Verschiebungen oder Spannungen an seiner Oberfläche. Bei dreidimensionaler Betrachtung lassen sich mit Hilfe der Cerutti-Potentialfunktion für nur wenige, sehr einfache Fälle geschlossene Lösungen angeben [8, 11].

- **Finite Elemente Methode**

Um Kontaktvorgänge mit FEM zu analysieren, werden elastische FE-Modelle der kontaktierenden Körper erstellt und bei Berührung die Randbedingungen verändert, sodass das Prinzip der Inpenetrabilität sowie ggf. ein Reibgesetz berücksichtigt wird. So lassen sich nahezu beliebig geformte Oberflächen darstellen, Spannungen und

Wellenausbreitung innerhalb der Körper studieren und mit entsprechenden Materialgesetzen und speziellen, nichtlinearen Algorithmen auch plastische Verformungen analysieren. Im Vergleich zur MKS-Simulation sind typischerweise wesentlich größere Gleichungssysteme zu lösen, die allerdings eine einfachere Struktur aufweisen. Die Rechenzeiten zur dynamischen Simulation dreidimensionaler Kontaktprobleme liegen üblicherweise bei mehreren Stunden [4].

Im Weiteren werden bei der Betrachtung von Kontaktvorgängen folgende Voraussetzungen zugrunde gelegt [10]:

1. Kontakt kann global als Prozess betrachtet werden, bei dem kinetische Energie umgewandelt wird in dissipative Energien wie Wärme, Schall und plastische Verformung, sowie wiederum kinetische Energie. Für die Dynamik eines MKS ist dabei von entscheidender Bedeutung, welchen Einfluss die Kontaktkräfte auf das System ausüben. Neben den Kräften ist insbesondere der kinematische Zustand des MKS nach dem Kontakt von Interesse. Innere Spannungen und Verformungen der sich berührenden Körper, thermodynamische Prozesse, Tribologie, Oberflächenrauheit und Verschleiß werden dagegen nicht näher betrachtet. Auch auf hydrodynamische Effekte, die bei sehr hohen Kollisionsgeschwindigkeiten auftreten, wird hier nicht eingegangen.
2. Nach dem *Prinzip von St. Venant* hängen Spannungen und Verformungen eines elastischen Körpers in hinreichender Entfernung vom Angriffsbereich äußerer Kräfte nicht mehr von ihrer speziellen Verteilung, sondern nur noch von ihrer Resultierenden ab. Für kontaktmechanische Problemstellungen in MKS bedeutet dies, dass die Kontaktspannungen berührender Körper, welche typischerweise in einem relativ kleinen Gebiet auftreten, zum resultierenden Kraftwinder aufintegriert bzw. aufsummiert werden können, um als eingeprägte Kräfte und Momente berücksichtigt zu werden. In diesem Sinne kann man weiterhin davon ausgehen, dass Veränderungen von Schwerpunktlage und Trägheitstensor infolge kleiner Verformungen durch Kontakt der Körper vernachlässigbar sind.
3. Zur Behandlung der Elastizität wird das *Hookesche Materialgesetz* verwendet, welches linear-elastisches Materialverhalten modelliert. Dabei wird von homogenem, isotropem Material ausgegangen, dessen Elastizität alleine durch den Elastizitätsmodul  $E$  und den Schubmodul  $G$  beschrieben wird.

## 1.2 Polygonal Contact Model

Obwohl es viele verschiedene Kontaktmodelle gibt, zeigen sich gängige MKS-Programme bei der Analyse von Systemen mit Berührungsvorgängen sehr eingeschränkt. So findet man einerseits ausgesprochen genaue und effiziente Methoden, die allerdings auf ganz spezielle Anwendungen beschränkt sind, wie zum Beispiel den Rad-Schiene Kontakt. Andererseits bilden allgemein gehaltene Modelle den Kontakt in so hohem Maße vereinfacht ab, dass die Ergebnisse der Simulation häufig zu deutlich von der Realität abweichen. Außerdem erweist sich in der Praxis meist als sehr hinderlich, dass viele Verfahren nur bestimmte



Oberflächengeometrien der Körper verarbeiten können, beispielsweise primitive Formen wie Ebenen, Zylinder und Ellipsoide oder aber ausschließlich konvexe Flächen.

In der Abteilung Fahrzeug-Systemdynamik des Deutschen Zentrums für Luft- und Raumfahrt (DLR) wird deshalb ein neuartiges *Polygonal Contact Model* (PCM) entwickelt, welches eine Beschreibung der Körperoberflächen in Polygondarstellung erlaubt und den Kontakt durch das Randschichtmodell darstellt. Bild 1.3 skizziert die Funktionsweise dieses Algorithmus.

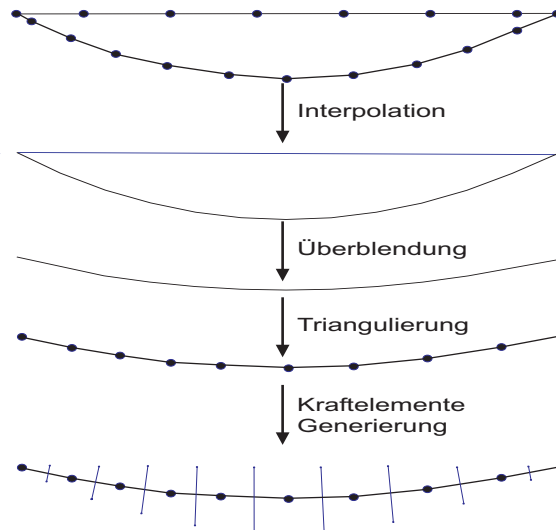


Bild 1.3: Prinzipielle Funktionsweise des Polygonal Contact Model

### 1.3 Aufgabenstellung

Ein wesentlicher Bestandteil des PCM ist die Approximation der Berührfläche im Kontaktgebiet. Im Rahmen dieser Arbeit soll ein hinsichtlich Effizienz, Robustheit und Flexibilität optimales Verfahren gefunden und implementiert werden, um diese Berührfläche aus den Schnittgebieten der unverformten Körperoberflächen abzuleiten. Weiter sollen bekannte Methoden zur möglichst gleichmäßigen Diskretisierung der Berührfläche gegenüber gestellt werden. Schließlich sollen die erarbeiteten Algorithmen in der Programmiersprache C implementiert und getestet werden.

# Kapitel 2

## Grundlagen der Flächendarstellung

Dieses Kapitel befasst sich mit Grundlagen zur Darstellung von Flächen im dreidimensionalen Raum. Zuerst werden implizite und parametrische Darstellung beschrieben und verglichen. Darauf folgt eine Beschreibung zusammengesetzter Feldfunktionen. Anschließend werden interpolierende und approximierende Flächen erläutert, gefolgt von einem Abschnitt über die Methode der „Variational Interpolation“.

### 2.1 Implizite und parametrische Darstellung

Während bei der mathematischen Beschreibung von Flächen in impliziter Form ein funktionaler Zusammenhang der zugrundeliegenden (globalen) Koordinaten besteht (z. B. in der Form  $f(x, y, z) = 0$ ), erfolgt die parametrische Darstellung in Abhängigkeit eines eigens eingeführten, zweidimensionalen Parameterraums, aus dem sich die Koordinaten jedes Flächenpunktes berechnen lassen (z. B. in der Form  $x = f_x(u, v)$ ,  $y = f_y(u, v)$  und  $z = f_z(u, v)$ ). Zuerst werden die Grundlagen der impliziten und der parametrischen Flächendarstellung beschrieben, gefolgt von einem Vergleich der beiden Flächentypen.

#### 2.1.1 Implizite Flächen

Das Konzept impliziter Flächen lässt sich sehr einfach durch eine zweidimensionale Analogie erklären: Ein Tropfen Öl auf feuchtem Asphalt wird mehrere, konzentrisch angeordnete, schillernde Farbringe erzeugen. Jeder einzelne Farbton wird eine geschlossene Kontur erzeugen. In Bild 2.1 wird dies am Beispiel des Farbtons Aquamarin verdeutlicht.

Diese Analogie lässt sich auf den dreidimensionalen Raum erweitern: Man stelle sich einen gefüllten Wassertank vor, in den ein Tropfen Farbe eingebracht wird. Dieser Tropfen wird sich durch Diffusion nach und nach im gesamten Volumen verteilen. Zu einem beliebigen Zeitpunkt, bevor der Tropfen gleichmäßig im gesamten Volumen verteilt ist, wird eine bestimmte Farbnuance eine geschlossene Fläche im Wassertank beschreiben.

Dementsprechend ist eine implizite Fläche eine Fläche mit einer bestimmten Eigenschaft. Im eben genannten Beispiel ist diese Eigenschaft die Farbe. Auf der impliziten Fläche ist der Wert der Eigenschaft konstant. Im Raum hingegen variiert er in Form eines stetigen Potentialfelds  $f(x, y, z)$ . Das Argument der Funktion  $f$  ist ein Punkt  $(x, y, z)$

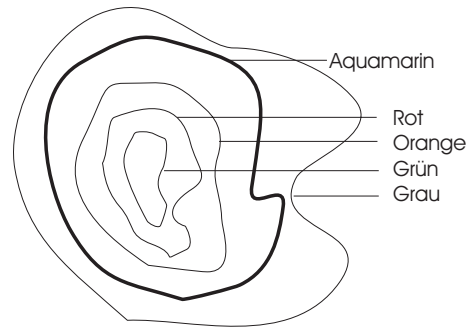


Bild 2.1: Analogie: Öltropfen auf nassem Asphalt

innerhalb des betrachteten Volumens. Flächen mit der Eigenschaft  $f(x, y, z) = \text{const}$  werden Isoflächen genannt. Sie sind vor allem dann von Interesse, wenn unterschiedliche Isowerte ausgewertet werden sollen. Implizite Flächen hingegen werden per Definition durch die Eigenschaft  $f(x, y, z) = 0$  beschrieben. Eine implizite Fläche ist also eine Isofläche mit dem Wert Null.

### 2.1.2 Parametrische Flächen

Bei parametrischen Flächen werden die Koordinaten der zugehörigen Punkte als Funktionen zweier Parameter ausgedrückt (Bild 2.2). Dabei finden folgende Formeln Anwendung:  $x = f_x(u, v)$ ,  $y = f_y(u, v)$  und  $z = f_z(u, v)$ .

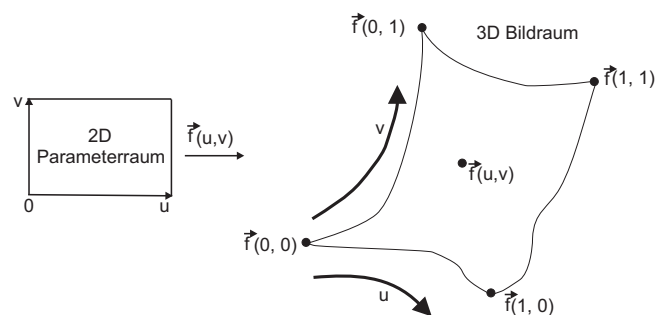


Bild 2.2: Prinzip parametrischer Flächendarstellung

Jeder 3D-Punkt auf der Fläche kann einem Parameterpaar  $(u, v)$  zugeordnet werden. Diese Art der Abbildung nennt sich Parametrisierung und findet beispielsweise bei der Texturierung (Texture-Mapping) Anwendung [13].

### 2.1.3 Vergleich parametrischer und impliziter Flächen

In Tabelle 2.1 werden die beiden Darstellungsformen am Beispiel einer Kugel mit dem Radius  $R$  gegenübergestellt.

Implizite Flächen haben gegenüber parametrischen folgende Vorteile:

parametrisch:	$x = R \cos v \cos u$ $y = R \cos v \sin u$ $z = R \sin v$
implizit:	$x^2 + y^2 + z^2 - R^2 = 0$

Tabelle 2.1: Kugel mit Radius  $R$  in impliziter und parametrischer Darstellung

1. Die implizite Darstellung ist in der Regel übersichtlicher.
2. Die implizite Darstellung beschreibt auch das Innere eines Objekts. Klassifikationen von Punkten (innerhalb oder ausserhalb des Objekts) sind dadurch deutlich leichter durchzuführen. Es muss lediglich das Vorzeichen des Funktionswertes  $f(x, y, z)$  ausgewertet werden.
3. Das Verschmelzen und Überblenden mehrerer Flächen ist bei impliziter Darstellung wesentlich einfacher.

Der letzte Punkt war entscheidend dafür, dass in dieser Arbeit mit impliziten Flächen gearbeitet wird.

## 2.2 Implizite Feldfunktionen

Der eine implizite Fläche umgebende Raum wird durch ein Skalarfeld repräsentiert. Für jeden Punkt  $P(x, y, z)$  in diesem Feld lässt sich der dazugehörige Funktionswert  $f(P) = f(x, y, z)$  berechnen. Wegen der Ähnlichkeit mit physikalischen Feldern werden diese Funktionen Feldfunktionen genannt.

Der Funktionswert  $f(P)$  gibt über das Vorzeichen Auskunft darüber, ob sich ein Punkt  $P$  innerhalb oder ausserhalb der Oberfläche befindet.

### 2.2.1 Klassifizierung

Gibt es eine zusammenhängende Menge von Punkten  $P$ , die alle den Funktionswert  $f(P) = 0$  haben (sich also auf der Oberfläche des Modells befinden), wurde eine die Oberfläche beschreibende Feldfunktion gefunden. Dabei kann die Funktion  $f$  jede Art von mathematischen Ausdrücken beinhalten:

1. Ein Ausdruck, der ausschließlich aus polynomialen Termen besteht, wird algebraisch genannt. Ein Beispiel hierfür ist die implizite Darstellung der Kugel in Tabelle 2.1.
2. Hat der Ausdruck auch nicht polynomiale Anteile (z. B. trigonometrische, exponentielle, logarithmische oder hyperbolische), wird er transzendental genannt.
3. Außerdem kann die Funktion  $f$  durch einen Algorithmus beschrieben werden. In diesem Fall können Eigenschaften nur durch numerische Auswertung gewonnen werden.

## 2.2.2 Kombination

Um komplexe Feldfunktionen umsetzen zu können, besteht die Möglichkeit, sie durch Linearkombination aus mehreren einzelnen zusammen zu setzen. Dabei wird der Funktionswert  $f(P)$  als gewichtete Summe von Feldfunktionen ausgedrückt:

$$f(P) = \sum g_i f_i(P) \quad (2.1)$$

Insbesondere wird es auf diese Weise möglich, zwischen zwei gegebenen Flächen stufenlos zu überblenden:

$$f(P) = \lambda f_1 + (1 - \lambda) f_2 \quad \lambda \in [0, 1] \quad (2.2)$$

Bild 2.3 zeigt zwei ebene Beispiele: Links Überblendungen zwischen Gerade und Bogen, rechts zwischen zwei Bögen.

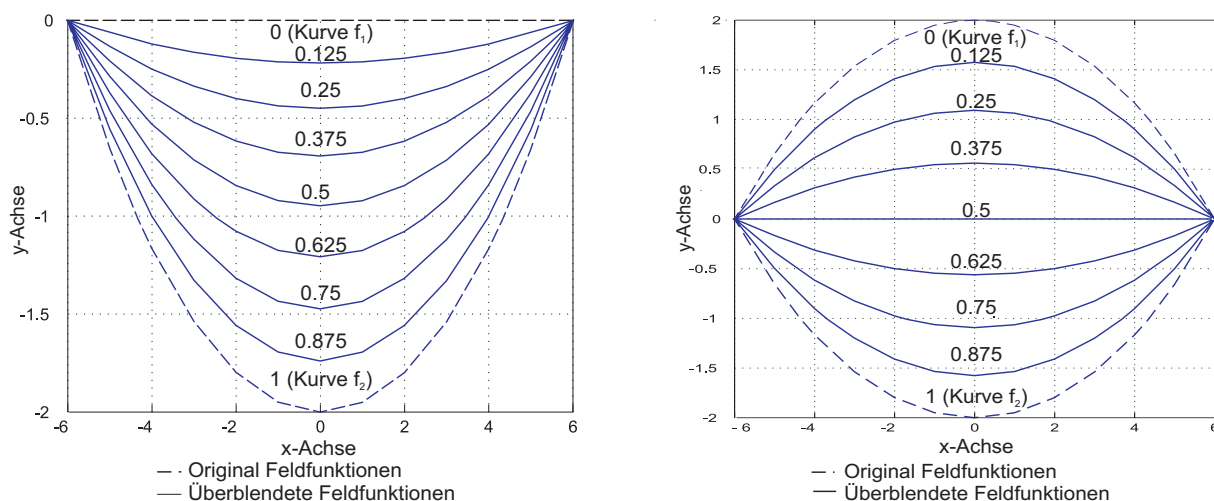


Bild 2.3: Beispiele überblendeter Feldfunktionen

Diese besondere Eigenschaft impliziter Flächen wird unter anderem zu Erzeugung von Morphing-Effekten ausgenutzt [19] und spielt in dieser Arbeit eine wichtige Rolle.

## 2.3 Freiformflächen

Die in der Industrie konstruierten Geometrien bestehen häufig aus komplexen Oberflächen, sog. Freiformflächen, deren Konstruktion mit verschiedensten Algorithmen erfolgt. Ausgegangen wird dabei von Punktmengen oder Kurvennetzen, die die Fläche beschreiben.

Durch eine Reihe vorgegebener Stützpunkte wird eine Fläche gelegt. Wenn diese Fläche genau durch die Stützpunkte verläuft, bezeichnet man sie als **interpolierende Freiformfläche**, während man eine möglichst „glatt“, aber nur näherungsweise durch die Stützpunkte verlaufende Fläche **approximierende Freiformfläche** nennt (Bild 2.4). Es gibt viele verschiedene Verfahren, die zur Flächengenerierung angewendet werden können, z.

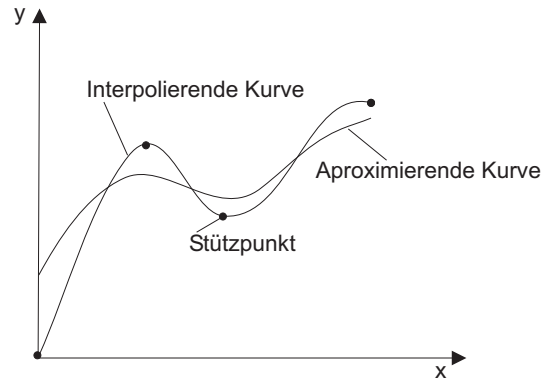


Bild 2.4: Interpolierende und approximierende Kurvendarstellung

B. Splinefunktionen, natürliche Polynome, Lagrange Polynome, Hermite Polynome oder Bernstein–Bezier Polynome.

Zur Erzeugung einer Freiformfläche muss man sich also zunächst darüber klar werden, auf welche Art und Weise interpoliert bzw. approximiert werden soll. Wird ein Verfahren gewählt, bei dem die Fläche zusammengesetzt wird, sind die Übergänge zwischen den Segmenten von großer Bedeutung. Hier ist auf die Erfüllung der geforderten Stetigkeitskriterien zu achten, die eng mit der mathematischen Darstellung der Fläche zusammen hängen. Soll die Fläche etwa glatt sein, müssen alle Ableitungen bis zur zweiten Ordnung stetig verlaufen.

In den folgenden Kapiteln werden *Thin-Plate-Spline* Flächen verwendet [19]. Wie ihr Name andeutet, wird hier die Krümmungsenergie der Fläche z. B. in der Darstellung  $z = f(x, y)$  betrachtet:

$$E = \int_{\Omega} (f_{xx}^2(x, y) + 2f_{xy}^2(x, y) + f_{yy}^2(x, y)) d\Omega \quad (2.3)$$

Die Fläche soll nun so angepasst werden, dass sie durch gegebene Stützpunkte verläuft, und möglichst glatt bleibt, indem sie  $E$  minimiert.

Thin-Plate-Spline Flächen zeichnen sich durch folgende Vorzüge aus:

1. Besondere Eignung für unregelmäßig verteilte Stützpunkte.
2. Geringe Neigung zum Oszillieren zwischen den Stützpunkten.
3. Interpolation, d. h. keine Abweichung von den Stützpunkten.

Das mathematische Fundament von Thin-Plate-Spline Flächen bildet die sog. *Variational Interpolation*. Auf sie wird im folgenden Abschnitt eingegangen.

## 2.4 Variational Interpolation

Variational Interpolation ist eine gebräuliche Methode, um eine interpolierende Funktion durch unregelmässig verteilte Stützpunkte (Scattered Data)  $f(\mathbf{c}_i) = h_i$ , für  $1 \leq i \leq k$  zu finden, die zusätzlich die Forderung erfüllt, dass ein Variationsfunktional minimal ist. Der mathematische Hintergrund ist hierbei die Variationsrechnung und die multivariante Interpolation anhand radialer Basisfunktionen [15].

### 2.4.1 Variationsrechnung

Eine reele Funktion  $y = f(x)$  besitzt genau dann einen **stationären Punkt**  $x_0$ , wenn  $f'(x_0) = 0$  gilt, d. h.  $f$  hat in diesem Punkt ein Minimum, ein Maximum oder einen horizontalen Wendepunkt (Bild 2.5).

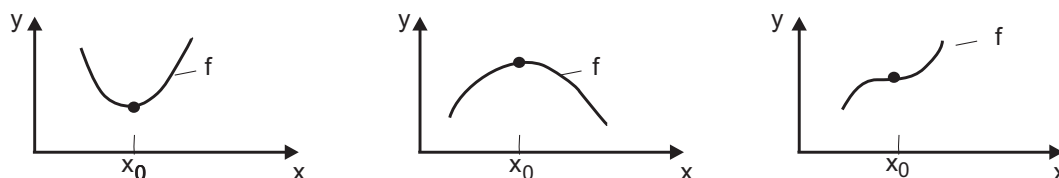


Bild 2.5: Stationäre Punkte einer Funktion  $y = f(x)$

Im Zusammenhang mit zahlreichen Problemen der mathematischen Physik genügt es nicht, nur Extrema zu ermitteln, sondern man muss sich mit Aufgaben beschäftigen, in denen ein Variationsintegral stationär wird.

Mit  $U$  bezeichnen wir die Menge aller Funktionen  $f$ , die der Interpolationsbedingung  $f(\mathbf{c}_i) = h_i$  genügen. Die Menge  $U$  besitzt folgende, leicht einzusehenden Eigenschaften:

$$\begin{aligned} f, g \in U &\Rightarrow (f + g) \in U, \\ f \in U, \quad t \in \mathbb{R} &\Rightarrow t \cdot f \in U. \end{aligned}$$

In der Mathematik spiegeln sich diese Eigenschaften im Begriff des linearen Vektorraums wider. Im Gegensatz zu den klassischen Vektoren stellen die Elemente der Menge  $U$  hier jedoch Funktionen (bzw. Vektoren aus Funktionen) einer oder mehrerer Veränderlicher dar. Jedem Element  $f_i \in U$  kann eine reelle Zahl  $\Pi(f_i)$  (z. B. die potenzielle Energie der entsprechenden Funktion) zugeordnet werden. Eine solche Zuordnung wird mathematisch durch den Begriff des (reelwertigen) Funktionals ausgedrückt:

$$f_i \in U \Rightarrow \Pi(f_i) \in \mathbb{R}. \quad (2.4)$$

Gesucht wird diejenige Funktion  $\bar{f}_i \in U$ , die das kleinste Funktional  $\Pi(\bar{f}_i)$  zur Folge hat:

$$\Pi(\bar{f}_i) \leq \Pi(f_i) \quad \forall f_i \in U \quad (2.5)$$

### 2.4.2 Interpolation mit radialen Basisfunktionen

Die multivariante Interpolation mit radialen Basisfunktionen wird seit einigen Jahren näher untersucht. Hierbei wird die interpolierende Funktion als Linearkombination von Basisfunktionen gebildet. In der Regel wird dabei die Basisfunktion als radial angenommen, d. h. ihr Wert an einer vorgegebenen Stelle  $\vec{x}$  hängt nur von der euklidischen Norm  $r = \|\vec{x}\|$  ab [20].

Die am häufigsten verwendeten radialen Basisfunktionen sind:

$$\begin{array}{lll} \phi(r) = e^{\beta r^2} & \beta > 0 & \text{Gaußglocken} \\ \phi(r) = r^\beta & \beta > 0 & \text{Potenziell} \\ \phi(r) = r^\beta \log(r) & \beta \in 2\mathbb{N} & \text{Biharmonisch} \\ \phi(r) = (r^2 + c^2)^{\beta/2} & \beta \notin 2\mathbb{N}, c > 0 & \text{(inverse) Multiquadrics} \end{array}$$

Die biharmonische Basisfunktion der Thin-Plate-Splines ist von großer Bedeutung, da die Interpolation die Forderung der Minimierung eines Variationsfunktionals erfüllt.

**Variational Interpolation** wird unter Verwendung von biharmonischen radialen Basisfunktionen<sup>1</sup>  $\phi(\vec{x}) = \|\vec{x}\|^3$  in folgender Weise definiert.

Liegen paarweise verschiedene Stützpunkte  $\vec{x}_1, \dots, \vec{x}_N \in \mathbb{R}^n$  vor, so lassen sich die Koeffizienten  $d_i$  der interpolierenden Funktion

$$f(\vec{x}) = \sum_{i=1}^N d_i \phi(\vec{x} - \vec{x}_i) \quad (2.6)$$

eindeutig bestimmen, sofern die Interpolationsmatrix

$$A = (\phi(\vec{x}_j - \vec{x}_k)) \quad 1 \leq j, k \leq N \quad (2.7)$$

invertierbar ist. Dies ist aber für die meisten radialen Basisfunktionen nicht der Fall, sodass dann üblicherweise ein multivariantes Polynom hinzugezogen wird, d. h. man setzt

$$f(\vec{x}) = \sum_{i=1}^N d_i \phi(\vec{x} - \vec{x}_i) + P(\vec{x}) \quad (2.8)$$

$$\begin{array}{ll} \vec{x}_i & : \text{ Stützpunkte} \\ d_i & : \text{ Gewichte} \\ \phi & : \text{ Radiale Basisfunktion} \\ P(\vec{x}) & : \text{ Multivariantes Polynom} \end{array}$$

Bei Interpolation mit Thin-Plate-Splines ist der Grad des Polynomes gleich eins.

Die Funktion (2.8) erfüllt die Forderung, dass das Variationsfunktional (2.3) minimal ist, d. h. kleiner als für alle anderen in Frage kommenden Funktionen, die die Interpolationsaufgabe lösen. Im zweidimensionalen Fall beschreibt das Funktional  $E$  die Verformungsenergie einer dünnen elastischen Platte (engl. thin plate). Analog liefert die Variational Interpolation auch im dreidimensionalen Raum eine möglichst glatte Interpolation.

<sup>1</sup>Eine andere Möglichkeit stellen Anisotropic Basis Functions dar [17].



Gegeben seien  $N$  Stützpunkte  $\vec{x}_i = (x_i, y_i, z_i)$ , an denen die Werte der gesuchten Feldfunktion  $f(\vec{x}_i) = h_i$  gefordert werden. Dann gelten nach (2.8) die  $N$  Gleichungen:

$$h_i = \sum_{j=1}^N d_j \phi(\vec{x}_i - \vec{x}_j) + P(\vec{x}_i) \quad 1 \leq i \leq N \quad (2.9)$$

mit

$$P(\vec{x}_i) = p_0 + p_1 x_i + p_2 y_i + p_3 z_i \quad (2.10)$$

Die vier verbleibenden Freiheitsgrade bindet man mit den Forderungen [20]:

$$\sum_{i=1}^N d_i u_i = 0 \quad u_i \in (1, x_i, y_i, z_i) \quad (2.11)$$

Anhand den Gleichungen 2.9, 2.10 und 2.11 kann das folgenden lineare Gleichungssystem aufgebaut werden, in dem der Vektor der gesuchten Koeffizienten  $d_i$  und  $(p_0, p_1, p_2, p_3)$  zu ermitteln ist:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & \dots & 1 \\ 0 & 0 & 0 & 0 & c_1^x & c_2^x & \dots & c_k^x \\ 0 & 0 & 0 & 0 & c_1^y & c_2^y & \dots & c_k^y \\ 0 & 0 & 0 & 0 & c_1^z & c_2^z & \dots & c_k^z \\ 1 & c_1^x & c_1^y & c_1^z & \phi_{11} & \phi_{12} & \dots & \phi_{1k} \\ 1 & c_2^x & c_2^y & c_2^z & \phi_{21} & \phi_{22} & \dots & \phi_{2k} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & c_k^x & c_k^y & c_k^z & \phi_{k1} & \phi_{k2} & \dots & \phi_{kk} \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \\ d_1 \\ d_2 \\ \vdots \\ d_k \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ h_1 \\ h_2 \\ \vdots \\ h_k \end{bmatrix} \quad (2.12)$$

Die Koeffizientenmatrix in (2.12) ist symmetrisch und semipositiv definit (siehe Anhang A.1). Deshalb gibt es eine eindeutige Lösung, sofern die Elemente auf der rechten Seite nicht alle null sind.

# Kapitel 3

## Approximation der Kontaktfläche

PCM [10] basiert auf einer Abwandlung des klassischen Randschichtmodells. Anstelle der für komplexe Oberflächen nicht näher definierten Wahl einer Tangentialebene, in die die Profile der beiden Körper projiziert werden, wird die elastische Randschicht entlang einer angenommenen Kontaktfläche zur Wirkung gebracht.

Aus der relativen Lage und Orientierung zweier betrachteter Körper ermittelt eine Kollisionsprüfung die Schnittlinien zwischen ihren durch Dreiecksflächen repräsentierten (unverformten) Oberflächen. Paarweise Abstandsberechnung zwischen den Endpunkten liefert dann einen (oder bei Mehrfachkontakt mehrere) Schnittpolygonzug, der die sich überschneidenden Gebiete von den aussenliegenden trennt. Im nächsten Schritt werden die „aktiven“ Polygone ermittelt. Darunter werden diejenigen Dreiecke verstanden, durch die das Schnittpolygon verläuft, sowie alle, die im Inneren des Schnittgebiets liegen.

In diesem Kapitel wird beschrieben, wie aus dem Schnittpolygonzug und den aktiven Dreiecken eine angenommene Kontaktfläche in impliziter Darstellung approximiert werden kann. Dies geschieht in zwei Schritten:

1. Interpolation der beiden aktiven Schnittflächen mittels Variational Interpolation
2. Überblendung zwischen den interpolierten Schnittflächen

Im Fall von Mehrfachkontakt werden die Schnitte sequentiell abgearbeitet, d. h. jedes Kontaktgebiet wird separat approximiert und trianguliert.

### 3.1 Interpolation der unverformten Schnittfläche

Die Feldfunktionen der beiden unverformten Schnittflächen im Kontaktgebiet werden durch Variational Interpolation gewonnen. Die dafür erforderlichen Stützpunkte werden aus den aktiven Dreiecken und dem Schnittpolygonzug abgeleitet.

### 3.1.1 Interpolationsbedingungen

Im Abschnitt 2.4.2 wurde gezeigt, dass das Gleichungssystem (2.12) eine eindeutige nicht-triviale Lösung hat, wenn im Vektor auf der rechten Seite nicht alle Elemente null sind. Um die impliziten Flächen mit Variational Interpolation erstellen zu können, werden daher nicht nur Punkte auf der Fläche

$$\begin{aligned} f(\vec{x}_v) &= 0 && \text{für die Eckpunkte der aktiven Dreiecke} \\ f(\vec{x}_p) &= 0 && \text{für die Eckpunkte des Schnittpolygonzugs} \end{aligned}$$

sondern auch Hilfspunkte außerhalb der Fläche benötigt.

$$\begin{aligned} f(\vec{x}_{v+}) &= +h && \text{für äußere Hilfspunkte} \\ f(\vec{x}_{v-}) &= -h && \text{für innere Hilfspunkte} \end{aligned}$$

### 3.1.2 Konstruktion der Hilfspunkte

Die Hilfspunkte liegen einen bestimmten Abstand  $\delta$  entfernt von den Eckpunkten der aktiven Dreiecke in einer Richtung, die aus den Normalenrichtungen der zum Eckpunkt gehörigen Dreiecke gemittelt wird (Bild 3.1).

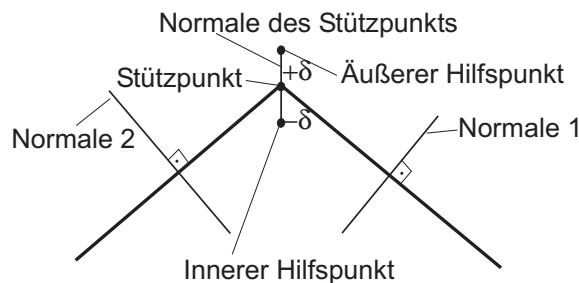


Bild 3.1: Konstruktion der Hilfspunkte

### 3.1.3 Einfluß der Hilfspunkte auf die Interpolation

In der Literatur findet man unterschiedliche Varianten der Variational Interpolation hinsichtlich der Wahl der Hilfspunkte. Um eine geeignete Auswahl für die Anwendung im PCM treffen zu können, wurden vor der Implementierung mit Matlab Untersuchungen an einfachen, zweidimensionalen Beispielen durchgeführt.

Die Bilder 3.2 und 3.3 zeigen repräsentative Ergebnisse dieser Tests. Eine Gerade (Bild 3.2) und ein Bogen (Bild 3.3) wurden mit einem einzigen inneren Hilfspunkt (links), allen inneren normalen Hilfspunkten (Mitte) und allen inneren und äußeren normalen Hilfspunkten (rechts) interpoliert. Nach diesen Ergebnissen ist zunächst kein Vorteil der aufwändigeren Varianten erkennbar.

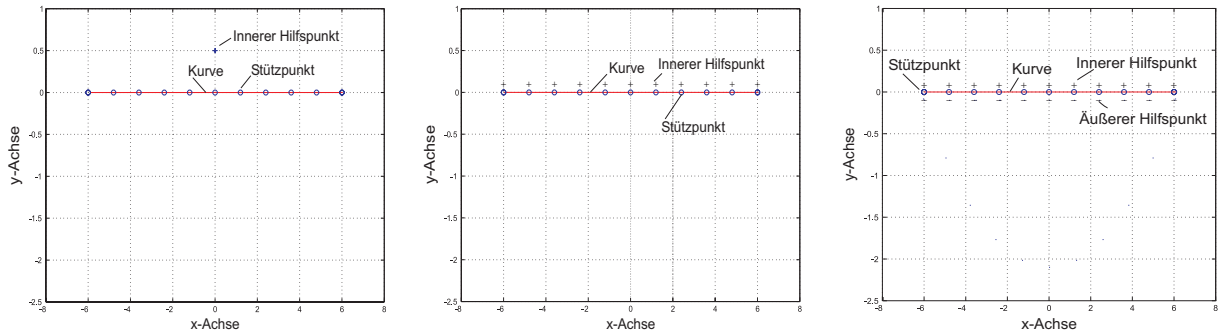


Bild 3.2: Variational Interpolation einer Gerade mit einem inneren Hilfspunkt (links), inneren normalen Hilfspunkten (Mitte) und inneren und äußeren normalen Hilfspunkten im Abstand  $\delta = 0.1$

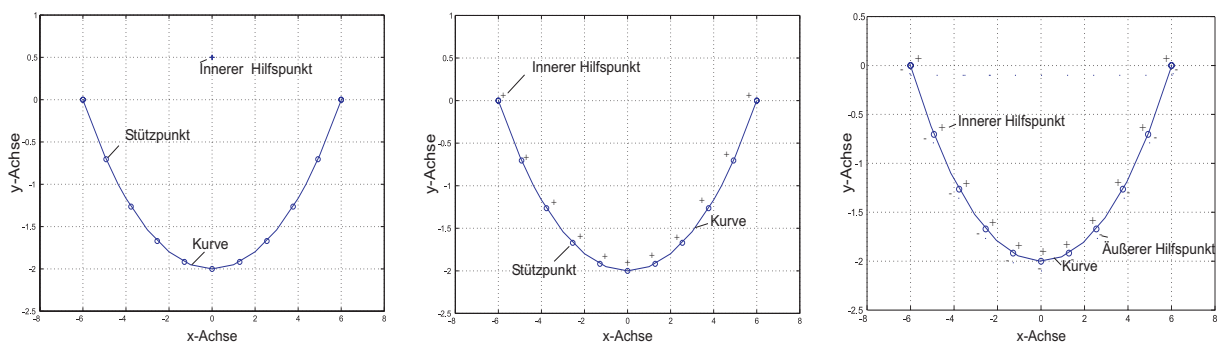


Bild 3.3: Variational Interpolation eines Bogens mit einem inneren Hilfspunkt (links), inneren normalen Hilfspunkten (Mitte) und inneren und äußeren normalen Hilfspunkten im Abstand  $\delta = 0.1$

## 3.2 Überblendung zwischen den Kontaktflächen

### 3.2.1 Approximation der Kontaktfläche

Die angenommene Berührfläche wird in zwei Schritten erzeugt:

1. Man erstellt jeweils die implizite Feldfunktion für die beiden aktiven Gebiete der Oberflächen mittels Variational Interpolation.

$$f_1(x, y, z), \quad f_2(x, y, z) \quad (3.1)$$

2. Die implizite Feldfunktion für die Kontaktfläche wird aus den beiden Funktionen linear kombiniert.

$$f(x, y, z) = \lambda f_1(x, y, z) + (1 - \lambda) f_2(x, y, z) \quad (3.2)$$

Der Faktor  $\lambda \in [0, 1]$  wird dabei im Sinne einer Reihenschaltung aus den Elastizitätsmoduli der beiden Körper gebildet.

### 3.2.2 Einfluss der Hilfspunkte auf die Überblendung

Wie im letzten Abschnitt wurden auch hier die drei Varianten für die Hilfspunkte (einzeln innerer Hilfspunkt, innere normale Hilfspunkte und innere und äußere normale Hilfspunkte) anhand zweidimensionaler Testkurven untersucht. Es zeigte sich ein erheblicher Einfluss auf den Verlauf der kombinierten Kurve, der für die Implementierung die Wahl auf die Version mit inneren und äußeren Hilfspunkten fallen ließ.

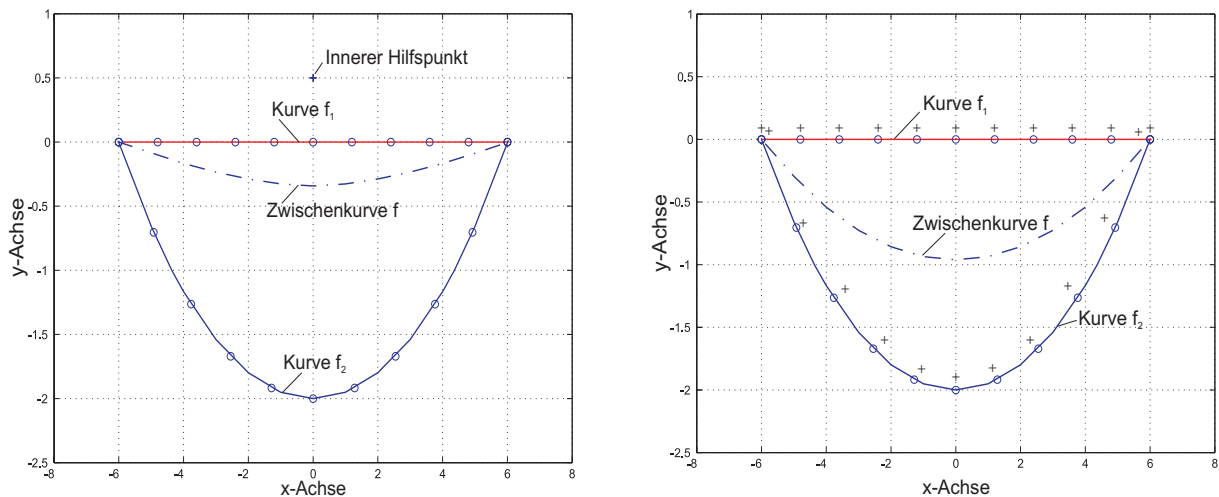


Bild 3.4: Überblendung zwischen einer Geraden und einem Bogen mit Kombinationsparameter  $\lambda = 0,5$ , links mit einem inneren Hilfspunkt, rechts mit äußeren und inneren normalen Hilfspunkten

Im Bild 3.4 wurde die Zwischenkurve einer Geraden und eines Bogens mit einem einzigen inneren Hilfspunkt (links) sowie allen inneren normalen Hilfspunkten (rechts) kombiniert. Der Parameter der Linearkombination ist  $\lambda = 0,5$ , d. h. die Zwischenkurve sollte in der Mitte zwischen den beiden Kurven liegen. Die Ergebnisse zeigen, dass nur der zweite Fall diese Forderung in guter Näherung erfüllt.

Weiter wurde der Einfluss des Abstands  $\delta$  zwischen den Stütz- und Hilfspunkten auf die Überblendung untersucht. Es wurde wieder die Zwischenkurve einer Geraden und eines Bogens (Bild 3.5) sowie zweier Bögen (Bild 3.6) getestet. Links wurden jeweils nur innere, rechts innere und äußere normale Hilfspunkte verwendet.

Die Ergebnisse zeigen, dass die Qualität der Zwischenkurven von der Wahl der Hilfspunkte abhängt. In Bild 3.5 ergeben sich nur kleine Abweichungen zwischen den überblendeten Kurven. Aber Bild 3.6 lässt in beiden Fällen aufschlussreiche Zusammenhänge erkennen. Wird nur mit inneren Hilfspunkten gearbeitet, hängt die Qualität der Zwischenkurve deutlich stärker vom Abstand  $\delta$  ab. Und in beiden Fällen liegt die Zwischenkurve für kleinere Werte von  $\delta$  wie gewünscht näher an der Symmetrieachse. Auch in der späteren 3D-Implementierung lieferte  $\delta = 0,01$  gute Ergebnisse.

Die Untersuchungen des Einflusses der Wahl der Hilfspunkte haben gezeigt, dass die Überblendung die besten Ergebnisse liefert, wenn die Kurven mit inneren und äußeren normalen Hilfspunkten in möglichst geringem Abstand interpoliert werden. Bild 3.7 zeigt

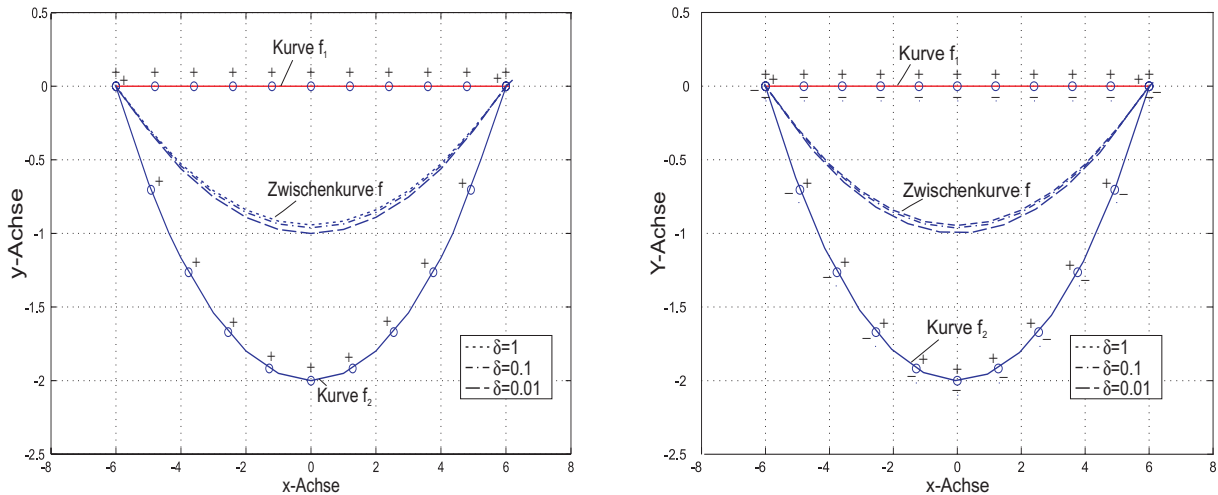


Bild 3.5: Überblendung zwischen einer Geraden und einem Bogen mit Kombinationsparameter  $\lambda = 0.5$ , links mit inneren normalen Hilfspunkten, rechts mit äusseren und inneren normalen Hilfspunkten, mit den Abständen  $\delta = 1, 0.1, 0.01$  zwischen Stütz- und Hilfspunkten

Zwischenkurven, die mit verschiedenen Kombinationsparametern  $\lambda$  übergeblendet wurden. Man sieht, dass sich die Überblendung zumindest zur Approximation von 2D-Kontaktflächen gut eignet. Auch im Dreidimensionalen bewährte sich das Verfahren; die Berührfläche in Bild 3.8 wurde anhand der im Rahmen dieser Arbeit implementierten Routinen berechnet.

### 3.3 Alternative Approximation der Konktakfläche

In [19] wird ein anderes Verfahren vorgeschlagen, um zwischen Flächen in impliziter Darstellung überzublendern. Auch diese Methode wurde anhand einfacher 2D-Beispiele untersucht.

#### 3.3.1 Interpolation

Auch hier wird von zwei Mengen von Stütz- und Hilfspunkten ausgegangen, die jeweils zur Interpolation einer impliziten Fläche dienen. Man erstellt jedoch nicht für jede Fläche eine eigene 3D-Feldfunktion, sondern eine einzige gemeinsame 4D-Feldfunktion, indem man die beiden 3D-Punktmenen um eine Dimension erweitert und kombiniert:

$$(x, y, z) \rightarrow (x, y, z, \lambda)$$

Es wird eine vierte Koordinate  $\lambda$  eingeführt, die z. B. für jeden Punkt der ersten Fläche null und für die Punkte der zweiten eins gesetzt werden kann. Dann wird die Variational Interpolation formal angewendet und liefert eine vierdimensionale Feldfunktion

$$f(x, y, z, \lambda) = 0 \tag{3.3}$$

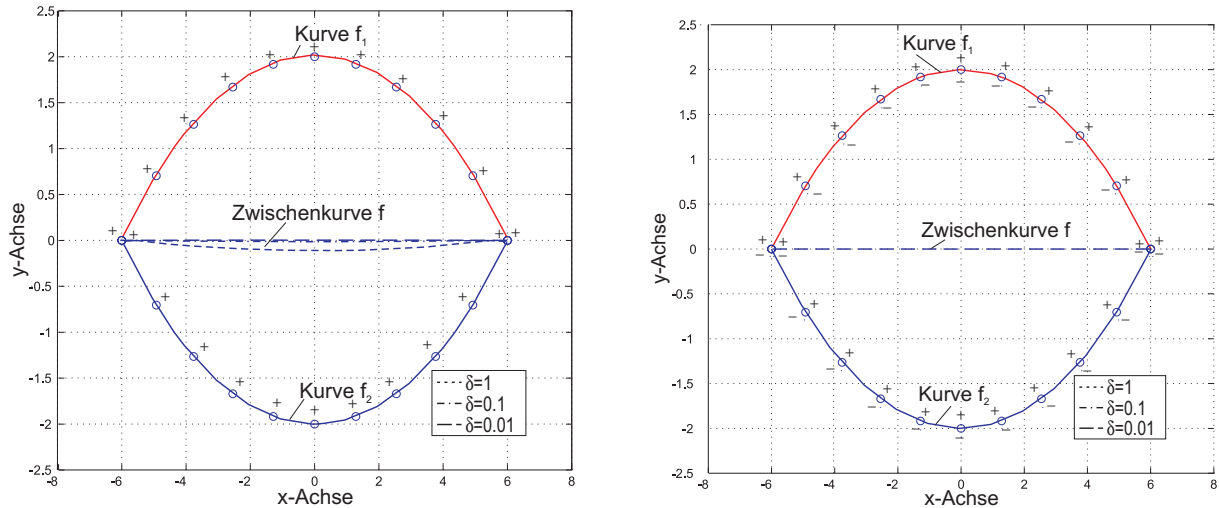


Bild 3.6: Überblendung zwischen zwei Bögen mit Kombinationsparameter  $\lambda = 0.5$ , links mit inneren normalen Hilfspunkten, rechts mit inneren und äußeren normalen Hilfspunkten, mit den Abständen  $\delta = 1, 0.1, 0.01$  zwischen Stütz- und Hilfspunkten

### 3.3.2 Überblendung

Setzt man nun in der Feldfunktion (3.3)  $\lambda = 0$ , erhält man eine 3D-Feldfunktion, deren Stützstellen auf der ersten Fläche liegen. Analog liefert  $\lambda = 1$  eine Variational Interpolation der zweiten Fläche. Ähnlich wie bei der Linearkombination erhält man die Feldfunktion für eine Zwischenfläche, indem man für  $\lambda$  einen Wert zwischen null und eins setzt, der aus dem Verhältnis der Elastizitäten der beiden Körper gebildet wird.

### 3.3.3 Praxistests

Auch die alternative Methode wurde mit Matlab umgesetzt und näher untersucht. Es zeigte sich zunächst, dass die Interpolation des Bogens bei Verwendung eines einzigen inneren Hilfspunkts nicht funktionierte (Bild 3.9 links), während die beiden anderen Varianten gute Ergebnisse lieferten (Bild 3.9 Mitte, rechts).

Doch in einem anderen Fall erwies sich das Verfahren als nicht brauchbar: Bild 3.10 zeigt den Versuch, die Zwischenkurve zweier Halbkreise zu berechnen. Hierbei traten Singularitäten auf, die dazu führten, dass die alternative Methode nicht weiter verfolgt wurde.

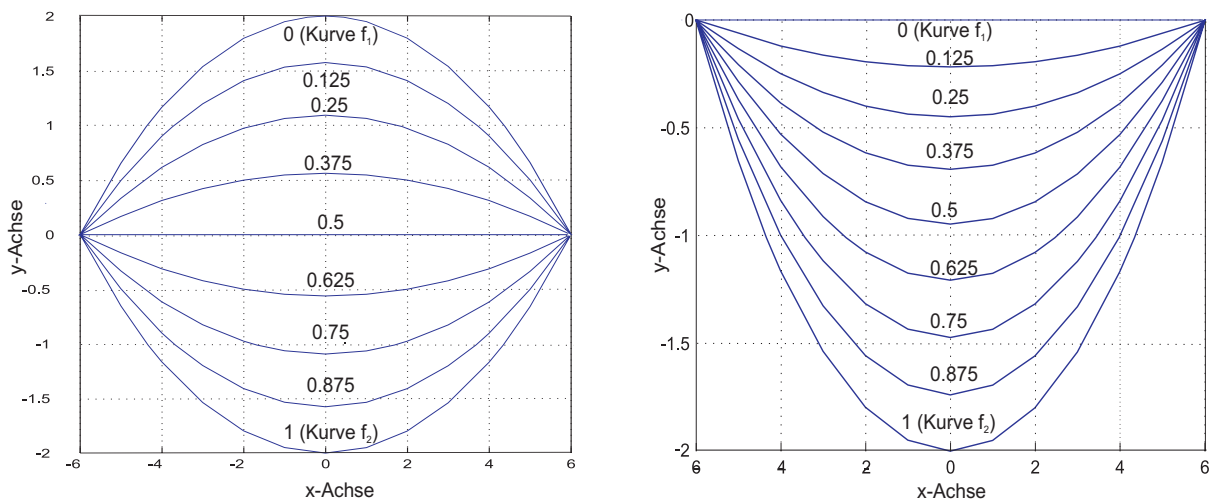


Bild 3.7: Mit verschiedenen Kombinationsparametern  $\lambda$  berechnete Zwischenkurven. Der Abstand zwischen Stütz- und Hilfspunkten beträgt  $\delta = 0.01$

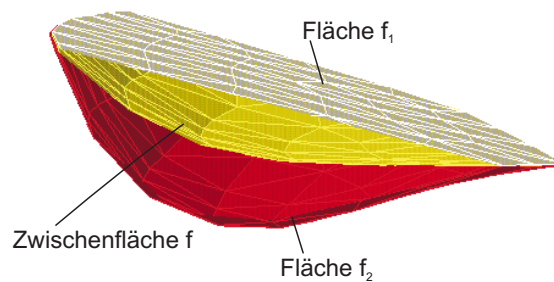


Bild 3.8: 3D-Beispiel einer approximierten Kontaktfläche

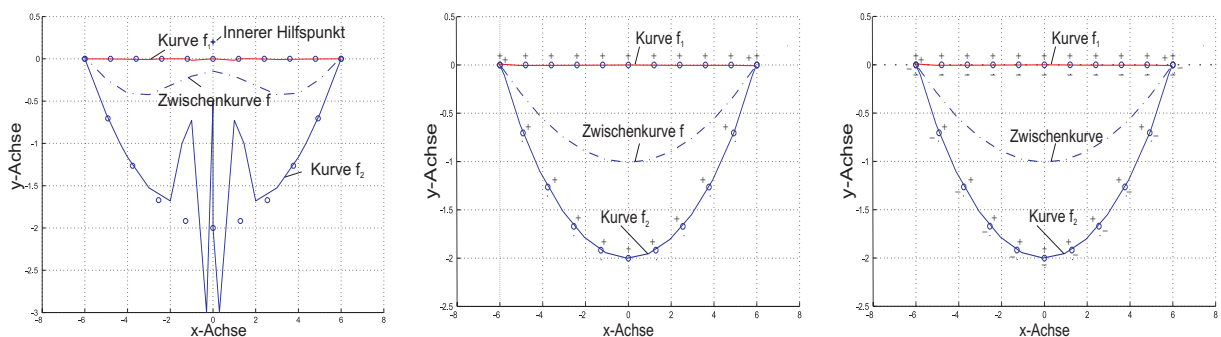


Bild 3.9: Alternative Überblendung zwischen einer Geraden und einem Bogen mit Kombinationsparameter  $\lambda = 0.5$ , mit einem inneren Hilfspunkt (links), inneren normalen Hilfspunkten (Mitte) und inneren und äußeren normalen Hilfspunkten im Abstand  $\delta = 0.1$



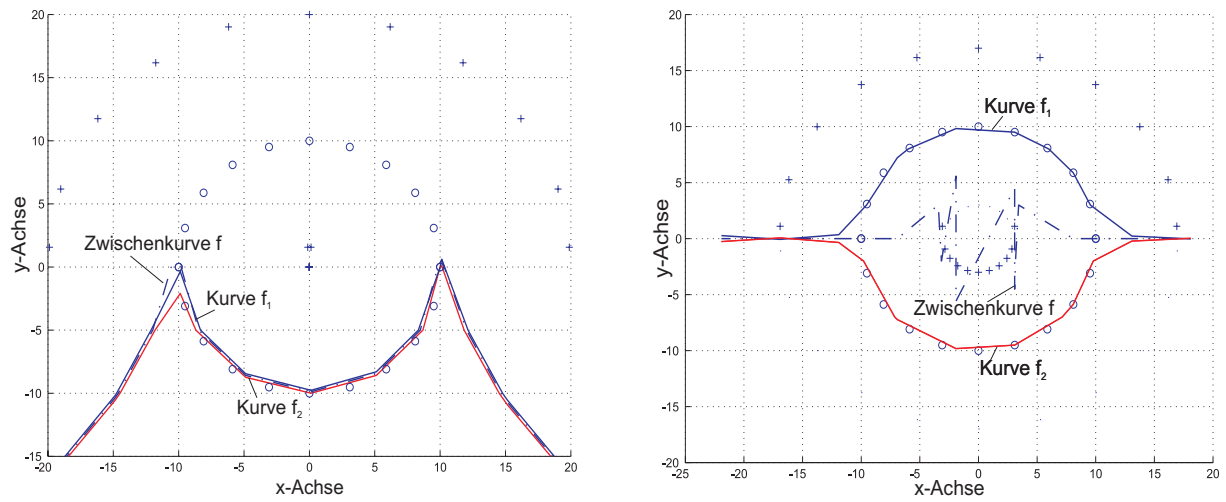


Bild 3.10: Alternative Überblendung zwischen zwei Halbkreisen mit Kombinationsparameter  $\lambda = 0.5$ , mit inneren normalen Hilfspunkten (links) und inneren und äußeren normalen Hilfspunkten (rechts) im Abstand  $\delta = 7$

# Kapitel 4

## Diskretisierung der Kontaktfläche

Nach Anwendung der Verfahren des vorigen Kapitels liegt die angenommene Kontaktfläche in impliziter Form vor und wird durch den Schnittpolygonzug begrenzt. In diesem Kapitel wird beschrieben, wie diese Fläche möglichst gleichmäßig trianguliert, d. h. in Dreiecke unterteilt werden kann. Schliesslich wird erläutert, wie aus den Dreiecken *Kontaktelemente* konstruiert werden, die als diskrete Elemente für das Randschichtmodell dienen.

Wird mit offenen Polygonflächen gearbeitet, können auch offene Schnittpolygonzüge auftreten. Für solche Fälle ist das hier vorgestellte Verfahren nicht anwendbar, weil für die im Abschnitt 4.2 beschriebene Triangulierung das Abbruchkriterium fehlt. Dagegen werden Flächen mit mehreren Berandungen, die z. B. bei ringförmigen Kontaktgebieten vorkommen, korrekt behandelt.

### 4.1 Triangulierung im Überblick

Für die Triangulierung impliziter Flächen gibt es eine Vielzahl von Algorithmen, die hier nicht alle dargestellt werden sollen. Es werden lediglich die wichtigsten Vertreter der verschiedenen Ansätze sowie einige wichtige Optimierungen der Grundalgorithmen analysiert.

Die Abschnitte 4.1.1 und 4.1.2 behandeln *volumenbasierte* Algorithmen, die Abschnitte 4.1.3 und 4.2 den *flächenbasierten* Marching Triangles Algorithmus.

Bei der volumenbasierten Triangulierung wird das die 3D-Fläche umgebende Volumen in Zellen aufgeteilt. Zwischen den Schnittpunkten der Zellenkanten mit der 3D-Fläche werden Polygone erzeugt und die Fläche somit approximiert. Der Vorteil hierbei ist, dass das zu verarbeitende Volumen algorithmisch relativ einfach abgearbeitet werden kann.

Bei der flächenbasierten Triangulierung wird ausgehend von einem initialen Polygon auf der Fläche (d. h. dass alle Eckpunkte des Polygons sich auf der Fläche befinden) ein weiteres Polygon auf derselben generiert und so weiter. Der Vorteil hierbei ist, dass die erzeugten Polygone wesentlich gleichmässiger sind, und dass nicht das gesamte die Fläche umgebende Volumen abgearbeitet werden muss.

Die 2D-Beispiele in Bild 4.1 deuten bereits darauf hin, dass die Polygone bei der flächenbasierten Triangulierung gleichmässiger sind. Ausserdem werden anstelle von 22 Linien

nur 16 Linien generiert, also ca. 30 % weniger, womit im weiteren Verlauf des PCM-Algorithmus 30 % Rechenleistung eingespart werden können [13].

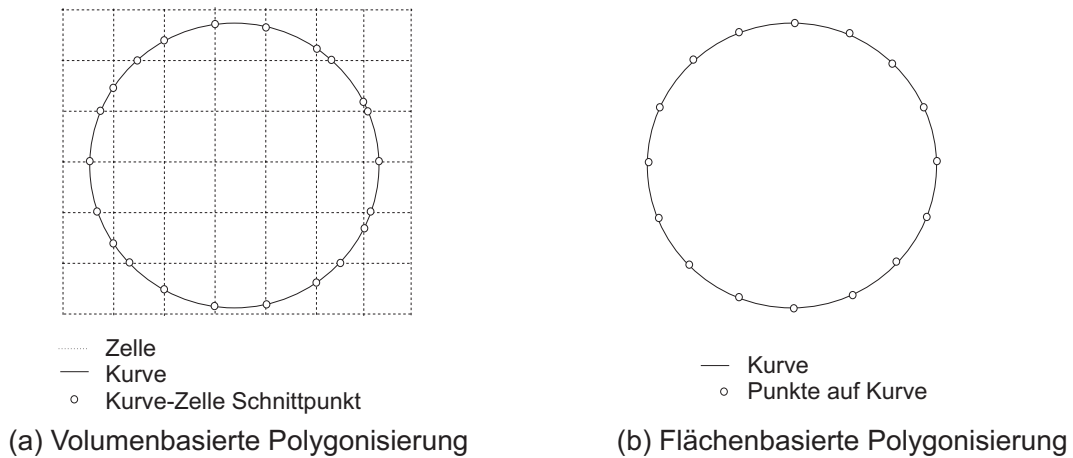


Bild 4.1: Prinzip volumen- und flächenbasierter Polygonisierung

### 4.1.1 Marching Cubes

Der Marching Cubes Algorithmus ist wohl der bekannteste volumenbasierte Algorithmus zur Triangulierung impliziter Flächen. Das die Fläche umgebende Volumen wird in gleichmässige Datenwürfel (Zellen) zerlegt [14]. Nun „wandert“ ein Würfel (bzw. ein Quader) von einer Zelle zur nächsten, bis alle Zellen bearbeitet wurden (Bild 4.2).

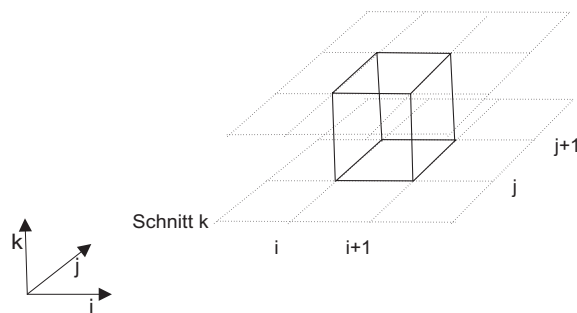


Bild 4.2: Marching Cubes: Prinzipdarstellung einer Zelle

Beim Wandern des Würfels durch das Volumen wird für jede Zelle festgestellt, ob sich die gesuchte implizite Fläche mit den Kanten des Würfels schneidet. Dies wird festgestellt, indem die Eckpunkte des Würfels als „innerhalb“ bzw. „ausserhalb“ der impliziten Fläche (also  $f(\vec{x}) < 0$  bzw.  $f(\vec{x}) > 0$ ) klassifiziert werden. Liefern die Eckpunkte einer Kante unterschiedliche Vorzeichen, schneidet sie die Fläche. Der genaue Schnittpunkt der Kante wird durch Interpolation der Funktionswerte der Eckpunkte ermittelt. Zwischen diesen Schnittpunkten werden schliesslich Dreiecke generiert und die implizite Fläche somit stückweise approximiert.

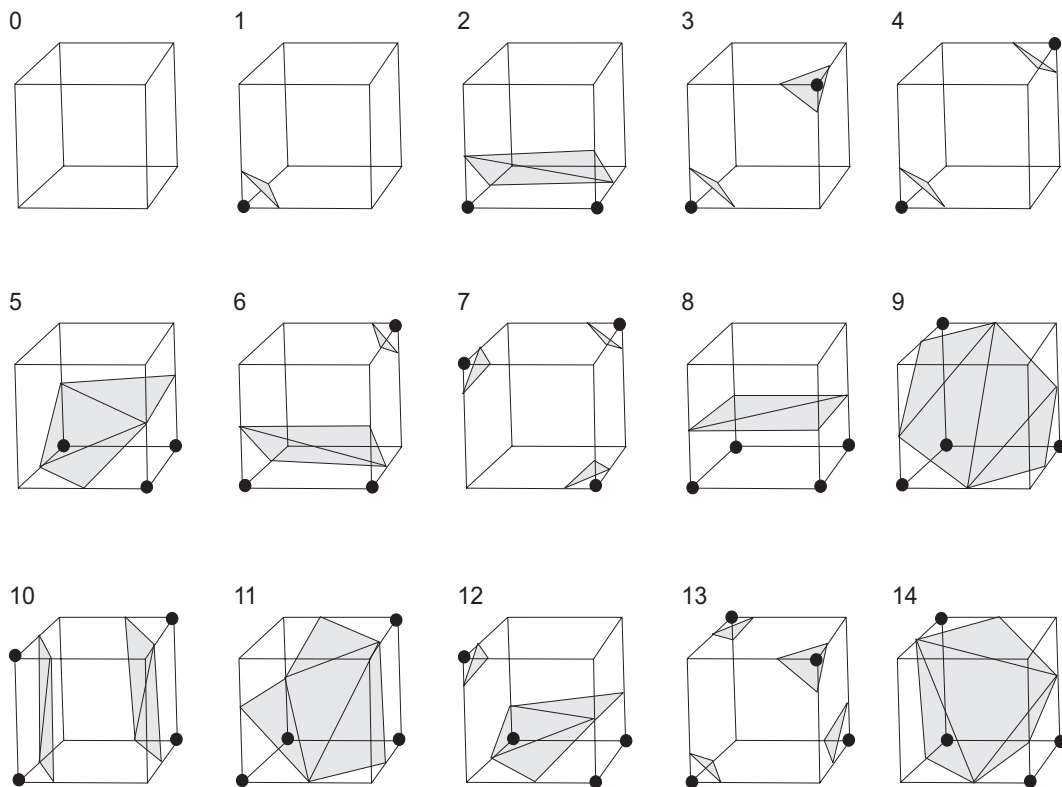


Bild 4.3: Relevante Klassifikationen der Würfel beim Marching Cubes Algorithmus

Da ein Würfel acht Ecken hat, ergeben sich für ihn theoretisch 256 mögliche Klassifikationen. Durch Ausnutzung von Symmetrien (so ist es z. B. unerheblich, ob sich alle Punkte innerhalb oder ausserhalb befinden, da das Ergebnis bis auf eine einfache Negation identisch ist) verbleiben noch 15 zu betrachtende Fälle, die in Bild 4.3 dargestellt werden. Allerdings sind in bestimmten Fällen mehrere Triangulierungen möglich. Beispielsweise lassen sich die Klassifikationen 3, 4, 6, 7, 10, 12 und 13 aus Bild 4.3 sinngemäß einer der beiden Triangulierungen aus Bild 4.4 zuordnen. Diese Mehrdeutigkeiten müssen durch Betrachtung der benachbarten Zellen und deren Triangulierung gelöst werden.

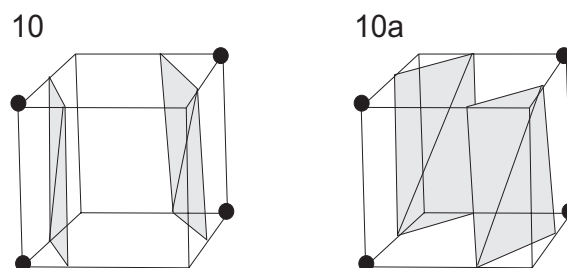


Bild 4.4: Mehrdeutigkeit bei der Triangulierung mit dem Marching Cubes Algorithmus

Der Marching Cubes Algorithmus ist relativ alt. Dadurch kann man mit Sicherheit eine passende Implementation finden, die direkt einsetzbar ist. Allerdings birgt er den Nachteil, dass das gesamte Volumen schrittweise abgearbeitet wird. Somit werden auch Zellen vom Typ 0 bearbeitet. Bei einer schlechten Wahl des betrachteten Bounding-Volumens (Bild

4.5 links) oder einer ungünstig geformten Fläche (Bild 4.5 rechts) werden so wesentlich mehr Zellen ausgewertet als eigentlich notwendig.

Desweiteren werden beim Marching Cubes Algorithmus Dreiecke verschiedenster Größe erzeugt, was zu nicht effektiv darstellbaren Dreiecksflächen führen kann. Die verschiedenen Größen der Dreiecke ergeben sich durch die unterschiedlichen Abstände der Schnittpunkte der impliziten Fläche mit den Kanten der Würfel und sind der Grund dafür, dass die meisten durch den Marching Cubes Algorithmus erzeugten Dreiecksflächen mit entsprechendem Aufwand simplifiziert werden müssen, bevor sie verwendet werden können (vgl. [18]).

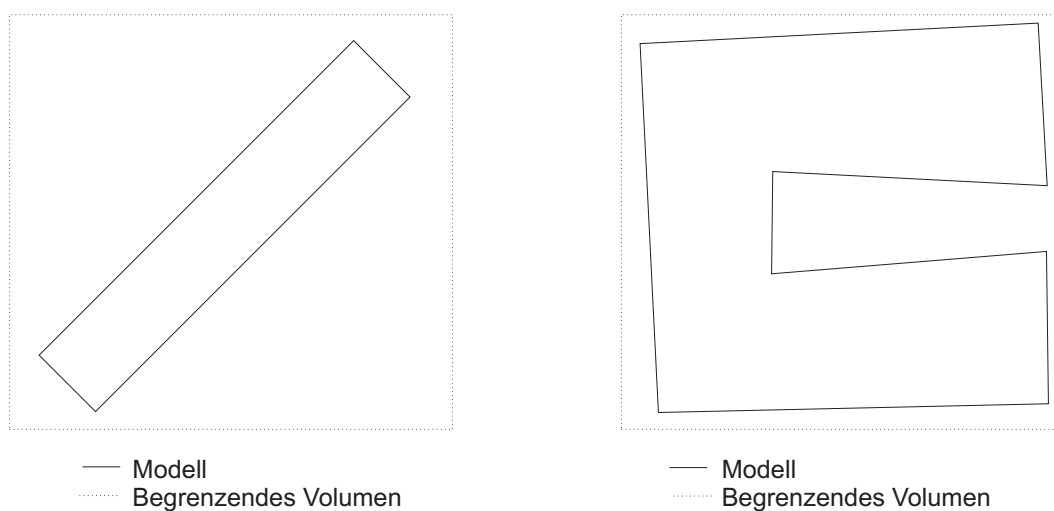


Bild 4.5: Ungünstige Bouding–Volumes für den Marching Cubes Algorithmus

### 4.1.2 Marching Tetrahedra

Eine Optimierung des Marching Cubes Algorithmus stellt der Marching Tetrahedra Algorithmus [18] dar. Hierbei wandert nicht ein Würfel durch die Zellen des Volumens, sondern ein Tetraeder. Der Vorteil hierbei ist, dass es nur drei verschiedenen zu betrachtende Fälle gibt (Bild 4.6). Ausserdem werden maximal zwei Dreiecke pro Tetraeder generiert und es treten keine Mehrdeutigkeiten auf.

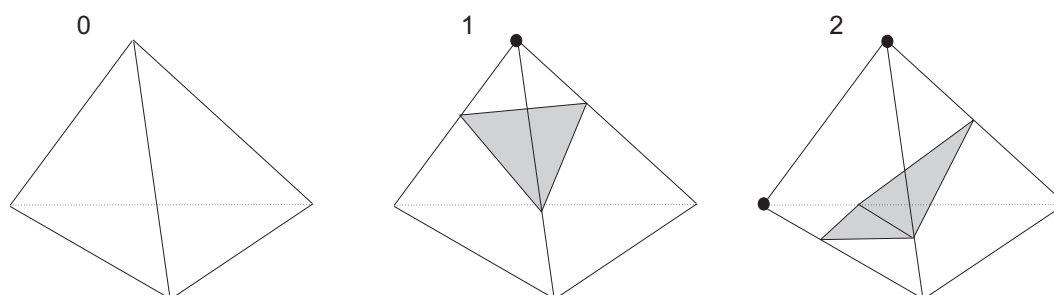


Bild 4.6: Klassifikationen der Tetraeder beim Marching Tetrahedra Algorithmus

Auch beim Marching Tetrahedra Algorithmus werden alle Zellen durchlaufen, unabhängig vom Typ der Zelle. Wenn die Fläche nicht durch die Zelle verläuft, wird die Zelle unnötigerweise bearbeitet. Um gleichmäßig große Dreiecke zu erhalten, muss auch hier ein zusätzlicher Bearbeitungsschritt durchgeführt werden [18].

### 4.1.3 Marching Triangles

Bei den bisher beschriebenen Ansätzen zur Triangulierung impliziter Flächen wurde das Volumen anfangs immer in Zellen aufgeteilt. Einen gänzlich anderen Ansatz verfolgt Hilton mit seinem Marching Triangles Algorithmus [9].

Die 3D-Fläche wird durch eine Liste aller aussen liegenden Kanten der Dreiecke repräsentiert. Ausgehend von einem initialen Teil der impliziten Fläche<sup>1</sup> wird eine Liste mit allen aussen liegenden Kanten generiert. Der Algorithmus basiert auf einfachem Durchlaufen dieser Liste. Neue, durch den Algorithmus generierte, Kanten werden am Ende der Liste angefügt, bearbeitete Kanten werden aus ihr gelöscht. Damit enthält das erste Element der Liste immer die als nächstes zu bearbeitende Kante. Die Kante, die gerade bearbeitet wird, wird aktive Kante genannt. Der Algorithmus wird so lange ausgeführt, bis alle Elemente der Liste bearbeitet wurden.

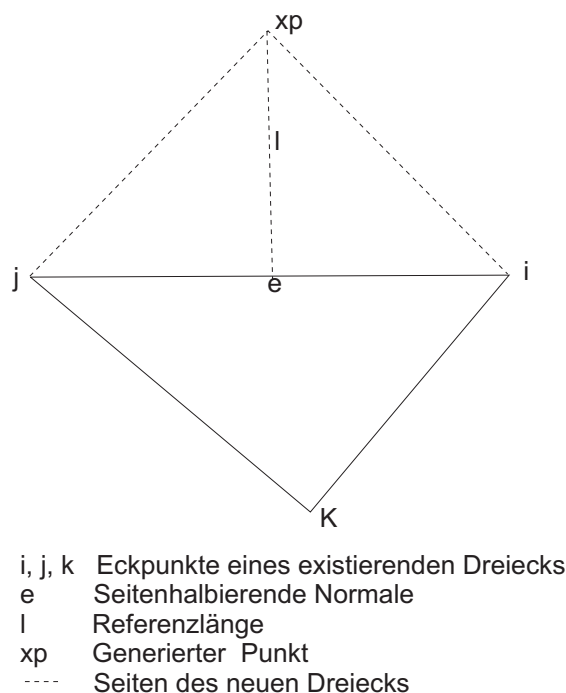


Bild 4.7: Generierung eines Dreiecks beim Marching Triangles Algorithmus

Die Bearbeitung eines Elements der Liste sieht folgende Schritte vor:

1. Es wird ausgehend von der aktiven Kante  $(i, j)$  ein neuer Punkt  $\vec{x}_p$  generiert. Hierzu wird der Mittelpunkt der aktiven Kante in Richtung der nach aussen weisenden, in

<sup>1</sup>Dabei kann es sich um ein einzelnes Dreieck oder um ein vorab generiertes Flächenteil handeln.

der Ebene des aktiven Dreiecks  $(i, j, k)$  liegenden Senkrechten der aktiven Kante um die Referenzlänge  $l$  verschoben (Bild 4.7).

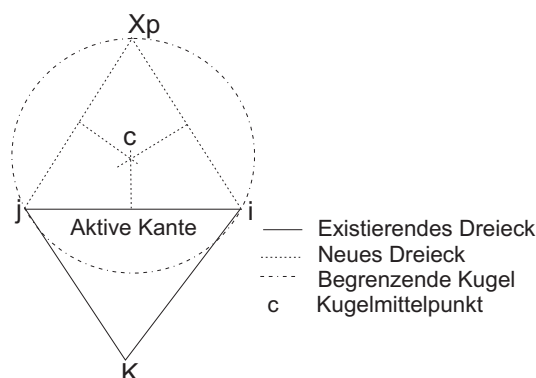


Bild 4.8: Geometrie des 3D-Delaunay Test

2. Der Punkt  $\vec{x}_p$  wird auf die implizite Fläche projiziert. Hierzu wird der Punkt entlang des Gradienten der Feldfunktion  $\nabla f(\vec{x}_p)$  um den Wert der Feldfunktion  $f(\vec{x}_p)$  verschoben.
3. Das neu generierte Dreieck wird dem *3D-Delaunay Test* unterzogen. Dieser Test gilt als bestanden, wenn sich kein Teil der bereits existierenden Dreiecke<sup>2</sup> innerhalb der Umkugel des zu testenden Dreiecks befindet (Bild 4.8).
4. Wenn der 3D-Delaunay Test bestanden wurde:
  - (a) Das neue Dreieck wird den bereits existierenden Dreiecken hinzugefügt.
  - (b) Die neuen Kanten des Dreiecks werden der Kantenliste hinzugefügt.
  - (c) Die aktive Kante wird aus der Kantenliste entfernt.
5. Wenn der 3D-Delaunay Test nicht bestanden wurde, wird geprüft, ob ein neu generiertes Dreieck mit  $\vec{x}_p = j_{left}$  bzw. mit  $\vec{x}_p = i_{right}$  den Test besteht. Wird der Test bestanden, werden die beteiligten Kanten aus der Liste entfernt. Dieser Schritt erlaubt das lokale Verbinden von bereits existierenden Teilen (Bild 4.9(a)).
6. Wenn auch diese 3D-Delaunay Tests nicht bestanden werden, wird geprüft, ob ein neu generiertes Dreieck mit  $\vec{x}_p = i$  bzw. mit  $\vec{x}_p = j$  den Test besteht. Dieser Schritt erlaubt das Schliessen von eventuell noch vorhandenen Lücken der Triangulierung und ist bei den meisten Flächen notwendig (Bild 4.9(b)).
7. Der Algorithmus endet, sobald die Liste der noch zu bearbeitenden Kanten leer ist.

<sup>2</sup>Hilton gibt keine Auskunft darüber, was er unter „Teil der Dreiecke“ versteht.

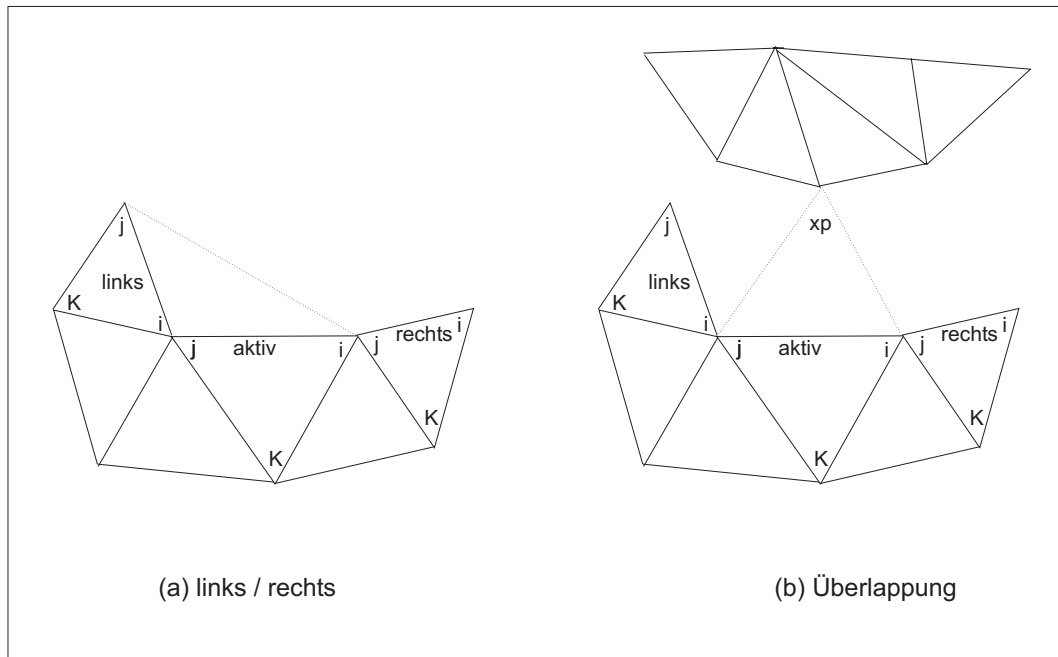


Bild 4.9: Generierung von Dreiecken zwischen bereits vorhandenen Teilen der Triangulierung

## 4.2 Triangulierung der Kontaktfläche

In der Veröffentlichung des Marching Triangles Algorithmus von Hilton fehlen einige Details, beispielsweise eine Beschreibung der Tests auf Überlappung mit bestehenden Teilen. Bei der Triangulierung mancher Funktionen kann es vorkommen, dass die Fläche nicht ganz geschlossen werden kann [13].

Für die Implementierung wurde daher eine weiterentwickelte, besser dokumentierte Version des Marching Triangles Algorithmus von Hartmann [7] gewählt und an die Besonderheiten zur Anwendung im PCM angepasst. Sie kann wie folgt charakterisiert werden:

1. Es wird ein kohärentes Netz selbst für sich schneidende Flächen gefunden. Die Kontaktfläche wird durch den Schnittpolygonzug eingegrenzt.
2. Der Algorithmus besteht im Groben aus vier Schritten und benötigt nur eine einfache Datenstruktur.

### 4.2.1 Grundidee

**Schritt 0:** Aus den Schnittpolygonzügen werden die initialen *Frontpolygone* generiert.

Das aktuelle Frontpolygon ist der Teil der Fläche, für den gerade die Triangulierung durchgeführt wird. Es wird repräsentiert durch die Punkte, die den Kantenzug des Polygons bilden.



**Schritt 1:** Für jeden Punkt des aktuellen Frontpolygons, der noch trianguliert werden muss, bestimmt man den *Frontwinkel*.

Der Frontwinkel in einem Punkt des Frontpolygons ist der Winkel, der durch die beiden angrenzenden Kanten aufgespannt wird.

**Schritt 2a:** Durch eine paarweise Abstandsmessung zwischen den Punkten des aktuellen Frontpolygons (mit Ausnahme der unmittelbaren Nachbarpunkte) wird festgestellt, ob sich die Triangulierung entsprechend Bild 4.15 schliessen lässt, sodass das aktuelle Frontpolygon in zwei kleinere unterteilt wird.

**Schritt 2b:** Die Entfernung der Punkte des aktuellen Frontpolygons  $\Pi_0$  zu den Punkten aller weiteren Frontpolygone  $\Pi_k$  mit  $k > 0$ . Wenn es Punkte  $p_{0i} \in \Pi_0$  und  $p_{mj} \in \Pi_m$  mit  $\|p_{0i} - p_{mj}\| < \delta_t$  gibt, dann vereinige  $\Pi_0$  mit  $\Pi_k$  zu einem neuen und größeren aktuellen Frontpolygon  $\Pi_0$ . Lösche das Frontpolygon  $\Pi_k$ .

**Schritt 3:** Der Eckpunkt des aktuellen Frontpolygons mit minimalem Frontwinkel wird mit in etwa gleichseitigen Dreiecken umgeben. Die neu entstandenen Eckpunkte werden anstelle dieses Punktes in das aktuelle Frontpolygon eingefügt.

**Schritt 4:** Die Schritte 1 bis 3 werden wiederholt, bis das aktuelle Frontpolygon nur noch aus drei Punkten besteht, die ein neues Dreieck bilden. Dann wird mit den restlichen Frontpolygone fortgefahren. Gibt es keine Frontpolygone mehr, wird der Algorithmus beendet.

## 4.2.2 Bestimmung von Flächenpunkten

Der Marching Triangles Algorithmus von Hartmann ist im Prinzip für verschiedene Darstellungen der zu triangulierenden Fläche anwendbar. Nur bei der Bestimmung eines einem gegebenen Punkt  $\vec{q}$  nahe gelegenen Flächenpunkts  $\vec{p}$  muss er speziell auf die mathematische Repräsentation angepasst werden. Diese Operation wurde in der Prozedur `pcm_vipoasp` für implizite Flächen implementiert und erfordert folgende Voraussetzungen:

1. eine implizite Darstellung der Fläche in der Form  $f(\vec{x}) = 0$ ;
2. der Gradient  $\nabla f(\vec{x})$  existiert und ist für jeden Punkt der Betrachtung nicht  $\vec{0}$ ;
3. ein Startpunkt  $\vec{q}$  in der Umgebung der Fläche.

Der Algorithmus berechnet aus diese Eingaben einen Punkt  $\vec{p}$ , der in der Nähe des Startpunkts  $\vec{q}$  liegt, sowie die zugehörige normierte Flächennormale  $\vec{n}$  und zwei normierte, zueinander senkrechte Tangentialvektoren  $\vec{t}_1, \vec{t}_2$ .

1. Bestimmung des Punkts  $\vec{p}$  mit dem Newton-Verfahren

$$\vec{u}_0 := \vec{q};$$

**repeat**

$$\vec{u}_{k+1} := \vec{u}_k - \frac{f(\vec{u}_k)}{\|\nabla f(\vec{u}_k)\|^2} \nabla f(\vec{u}_k) \quad (\text{Herleitung siehe Anhang A.3})$$

**until**  $\|\vec{u}_{k+1} - \vec{u}_k\| < \varepsilon$

$$\vec{p} := \vec{u}_{k+1}$$

2. Bestimmung der Flächennormalen im Punkt  $\vec{p}$

$$\vec{n} := \frac{\nabla f(\vec{p})}{\|\nabla f(\vec{p})\|}$$

3. Bestimmung der Tangentialvektoren im Punkt  $\vec{p}$

$$(n_x, n_y, n_z) := \vec{n}$$

**if**  $n_x > 0.5$  **or**  $n_y > 0.5$

$$\vec{t}_1 := \frac{(n_y, -n_x, 0)}{\|n_y, -n_x, 0\|}$$

**else**

$$\vec{t}_1 := \frac{(-n_z, 0, n_x)}{\|-n_z, 0, n_x\|}$$

$$\vec{t}_2 := \vec{n} \times \vec{t}_1$$

### 4.2.3 Implementierung

#### 4.2.3.1 Datenstruktur

Die Datenstruktur ist relativ einfach. Für die Konstruktion der Dreiecke benötigt man eine Referenzlänge  $\delta_t > 0$ , die die gewünschte Länge der Dreiecksseiten vorgibt. Die Eckpunkte werden fortlaufend durchnummeriert. Jeder einzelne Punkt speichert folgende Informationen:

1. Seine Koordinaten  $(x, y, z)$ ;
2. seine Flächennormale  $\vec{n}$  und die beiden Tangentialvektoren  $\vec{t}_1$  und  $\vec{t}_2$ ;
3. den Frontwinkel `angle`, falls der Punkt Eckpunkt eines Frontpolygons ist;
4. die boolesche Variable `angle_changed`, die angibt, ob der aktuelle Frontwinkel geändert wurde und neu berechnet werden muss.

Die Dreiecke werden ebenfalls fortlaufend nummeriert. Jedes Dreieck speichert die Indizes seiner Eckpunkte. Die Frontpolygone werden durch Indexlisten ihrer Eckpunkte repräsentiert.

#### 4.2.3.2 Algorithmus

**Schritt 0:** Der PCM Algorithmus liefert die Schnittpolygonzüge der beiden unverformten Oberflächen. Sie dienen als Begrenzung der Triangulierung. Um Frontpolygone mit gleichmäßiger Kantenlänge zu erhalten, werden sie zunächst äquidistant unterteilt (Bild 4.11). Das zu bearbeitende Gebiet erkennt man am Umlaufsinn der Schnittpolygonzüge: Die Kontaktfläche liegt stets rechts zur Laufrichtung (Bild 4.12)

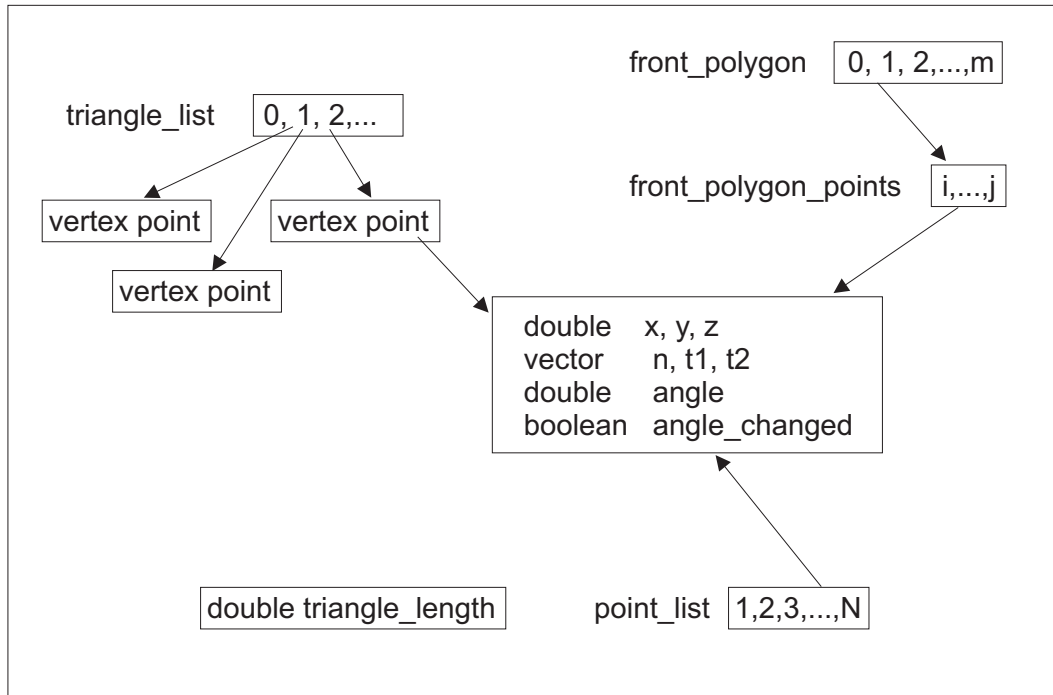


Bild 4.10: Datenstruktur der Marching Triangles Implementierung

**Schritt 1:** Wenn ein Punkt  $p_{0i}$  des aktuellen Frontpolygons  $\Pi_0 = (p_{01}, p_{02}, \dots, p_{0N_0})$  gerade eingefügt wurde oder ein Nachbar von  $p_{0i}$  ein neuer Punkt ist, dann muss der aktuelle Frontwinkel  $\omega$  im Punkt  $p_{0i}$  neu berechnet werden (Bild 4.13).

if  $i = 1$

$\nu_1 := p_{0N_0}$

else

$\nu_1 := p_{0,i-1}$

if  $i = N_0$

$\nu_2 := p_{01}$

else

$\nu_2 := p_{0,i+1}$

$(\xi_1, \eta_1, \zeta_1)$  und  $(\xi_2, \eta_2, \zeta_2)$  sind die Koordinaten von  $\nu_1$  bzw.  $\nu_2$  im lokalen Orthonormalsystem  $(\vec{t}_1, \vec{t}_2, \vec{n})$  im Punkt  $p_{0i}$ .

$\omega_1 :=$  Polarwinkel von  $(\xi_1, \eta_1) \in [0, 2\pi]$

$\omega_2 :=$  Polarwinkel von  $(\xi_2, \eta_2) \in [0, 2\pi]$

(siehe Anhang A.5)

if  $\omega_2 \geq \omega_1$

$\omega := \omega_2 - \omega_1$

else

$\omega := \omega_2 - \omega_1 + 2\pi$

**Schritt 2:** Um ein Überlappen eines neuen Dreiecks mit den bereits vorhandenen zu verhindern, wird überprüft:

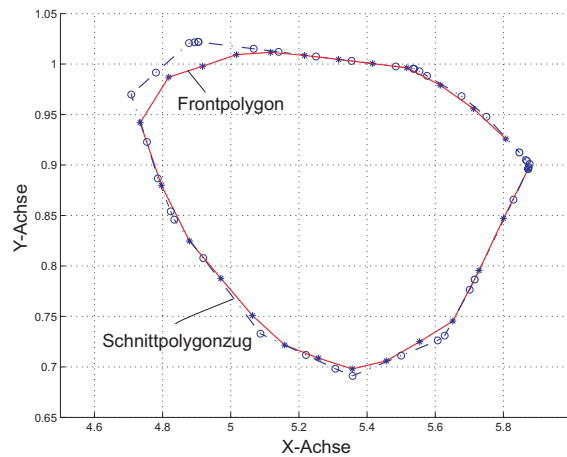


Bild 4.11: Übergang vom Schnittpolygonzug zum Frontpolygon

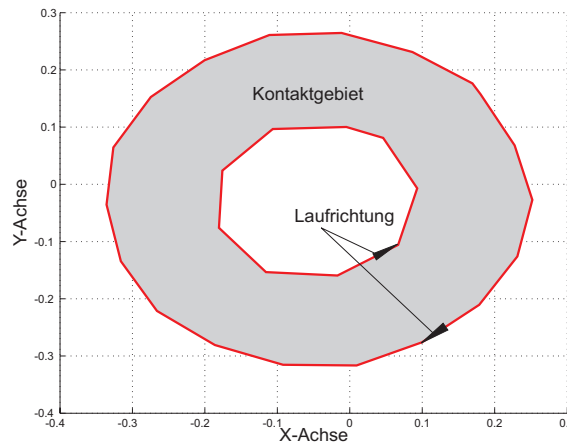


Bild 4.12: Laufrichtung der Schnittpolygonzüge und Kontaktgebiet

- die Entfernung von Paaren von Punkten des aktuellen Frontpolygons  $\Pi_0$ . Wenn es zwei Punkte  $p_{0i}$  und  $p_{0j}$  gibt, die weder Nachbarn noch Nachbarn von Nachbarn sind und  $\|p_{0i} - p_{0j}\| < \delta_t$  ist, dann wird das aktuelle Frontpolygon aufgeteilt in ein neues aktuelles Frontpolygon  $(p_{01}, \dots, p_{0i}, p_{0j}, \dots, p_{0N_0})$  mit  $i + N_0 - (j - 1)$  Punkten und ein weiteres Frontpolygon  $(p_{0i}, \dots, p_{0j})$  mit  $j - i + 1$  Punkten (Bild 4.14). Die Punkte  $p_{0i}$  und  $p_{0j}$  dürfen nicht in spätere Entfernungsprüfungen einbezogen werden.
- die Entfernung der Punkte des aktuellen Frontpolygons  $\Pi_0$  zu den Punkten aller weiteren Frontpolygone  $\Pi_k$  mit  $k > 0$ . Wenn es Punkte  $p_{0i} \in \Pi_0$  und  $p_{mj} \in \Pi_m$  mit  $\|p_{0i} - p_{mj}\| < \delta_t$  gibt, dann werden die Polygone  $\Pi_0 = (p_{01}, \dots, p_{0N_0})$  und  $\Pi_m = (p_{m1}, \dots, p_{mN_m})$  zu einem neuen aktuellen Frontpolygon  $\Pi_0 = (p_{01}, \dots, p_{0i}, p_{mj}, \dots, p_{mN_m}, p_{m1}, \dots, p_{mj}, p_{0i}, \dots, p_{0N_0})$  mit  $N_0 + N_m + 2$  Punkten vereinigt (Bild 4.15).

Die Punkte  $p_{0i}$  und  $p_{mj}$  erscheinen zweimal. Bevor Weiteres unternommen wird, sollte man den Frontwinkel dieser Punkte bestimmen, wenn sie das erste Mal im Polygon  $\Pi_0$  erscheinen, und erst denjenigen mit dem kleineren Winkel mit Drei-

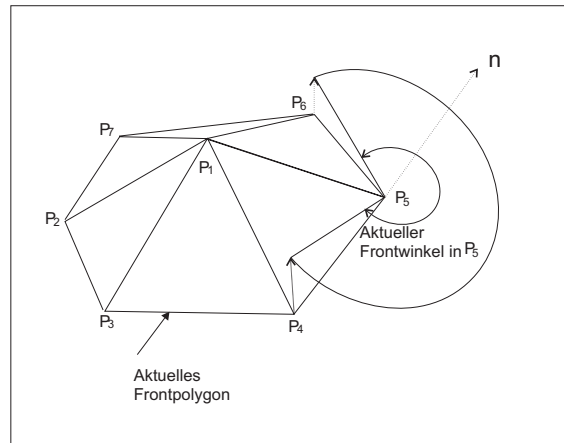


Bild 4.13: Berechnung des aktuellen Frontwinkels

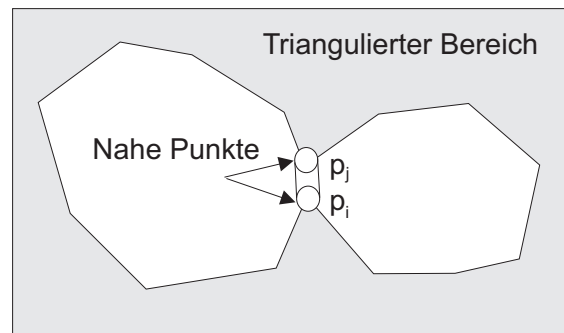


Bild 4.14: Teilen des aktuellen Frontpolygons

ecken umgeben (s. Schritt 3), und anschliessend den anderen Punkt. Nach dieser Operation werden diese beiden Punkte im aktuellen Frontpolygon gelöscht, und ihre boolesche Variable `angle_changed` wird wieder auf `true` gesetzt, um eine Neuberechnung ihrer Frontwinkel zu sichern.

**Schritt 3:** Ist  $p_{om}$  der Punkt des aktuellen Frontpolygons  $\Pi_0$  mit minimalem Frontwinkel  $\omega$ , dann vervollständige die Triangulierung folgendermaßen:

1. Bestimme die Nachbarn  $\nu_1, \nu_2$  von  $p_{om}$  (s. Schritt 1);
2. Bestimme die Anzahl der Dreiecke  $n_t$ , die generiert werden sollen:

$$n_t := \text{int}(3\omega/\pi) + 1$$

$$\Delta\omega := \omega/n_t$$

Korrektur für extreme Fälle (Bild 4.16):

$$\text{if } \Delta\omega < 0.8 \text{ and } n_t > 1$$

$$\Delta\omega := \omega/(n_t - 1)$$

$$n_t := n_t - 1$$

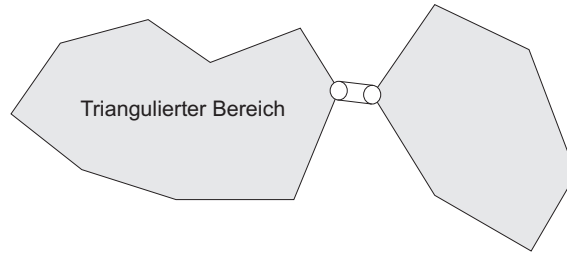


Bild 4.15: Verbinden des aktuellen Frontpolygons mit einem anderen

if  $\Delta\omega > 0.8$  and  $n_t = 1$  and  $\|\nu_1 - \nu_2\| > 1.2\delta_t$

$$\Delta\omega := \omega/2$$

$$n_t := 2$$

if  $\omega < 3$  and (  $\|\nu_1 - p_{0m}\| < 0.5\delta_t$  and  $\|\nu_1 - \nu_2\| > 1.2\delta_t$  or  $\|\nu_2 - p_{0m}\| < 0.5\delta_t$  )

$$n_t := 1$$

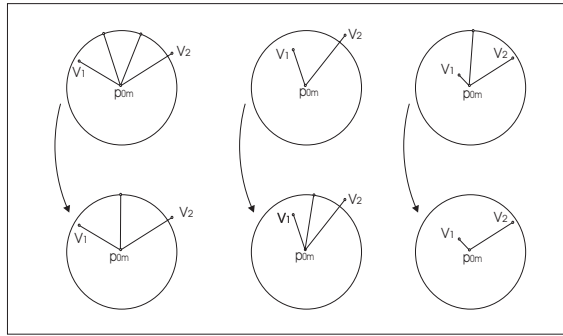


Bild 4.16: Korrekturen für extreme Fälle

### 3. Generiere neue Dreiecke:

Ist  $n_t = 1$ , erzeugt man ein neues Dreieck  $(\nu_1, \nu_2, p_{0m})$ . Sonst sind  $q_0$  und  $q_{n_t}$  die orthogonalen Projektionen von  $\nu_1$  und  $\nu_2$  in die Tangentialebene des Punktes  $q_{0m}$ , und die Punkte  $q_i$  ergeben sich aus einer Rotation des Punktes  $p_{0m} + \delta_t(q_0 - p_{0m}/\|q_0 - p_{0m}\|)$  um den Winkel  $i\Delta\omega$  um die Normale in  $p_{0m}$  (für  $i = 1, \dots, n_t - 1$ ) (Bild 4.17). Durch Anwendung der Prozedur `pcm_vipoasp` auf die Punkte  $q_i$  erhält man die neuen Punkte  $p_{N+i}$ , wobei  $N$  die Anzahl aller bereits existierenden Punkte und  $n_t$  die der neuen Dreiecke ist:

$$(\nu_1, p_{N+1}, p_{0m}), (p_{N+1}, p_{N+2}, p_{0m}), \dots, (p_{N+n_t-1}, \nu_2, p_{0m})$$

### 4. Erneure das aktuelle Frontpolygon:

Lösche den Punkt  $p_{0m}$ . Wenn  $n_t > 1$  ist, füge die neuen Punkte  $p_{N+1}, \dots, p_{N+n_t-1}$  ein. Durch setzen der boolschen Variablen `angle_changed` für die Punkte  $\nu_1, \nu_2, p_{N+1}, \dots, p_{N+n_t-1}$  wird die Neuberechnung ihrer Frontwinkel vorgemerkt.

Bild 4.18 zeigt das Ergebnis einer Triangulierung mit dem im Rahmen dieser Arbeit implementierten Algorithmus für die in Bild 4.19 dargestellte Kontaktsituation.

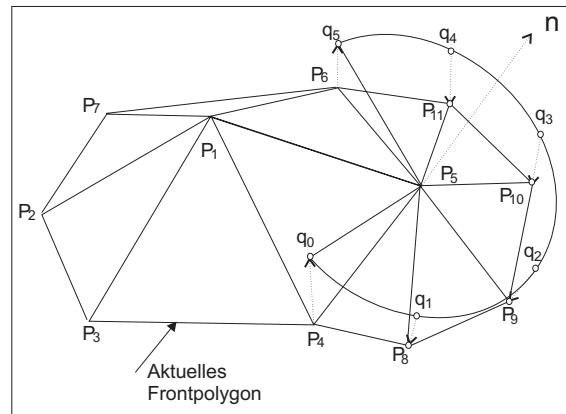


Bild 4.17: Generieren neuer Dreiecke

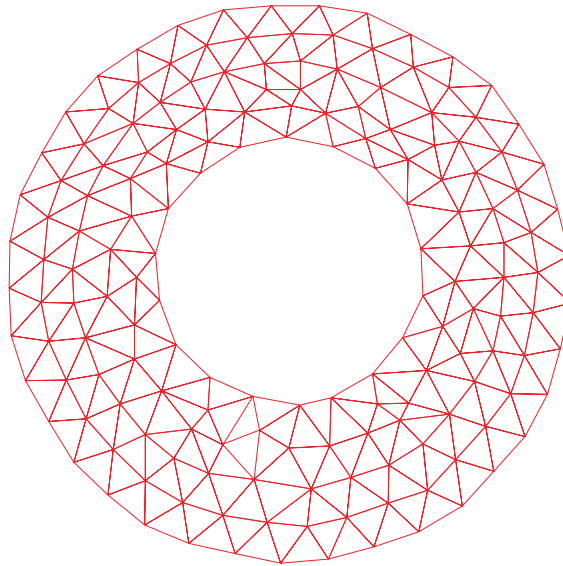


Bild 4.18: Triangulierung der Kontaktfläche zwischen Kugel und Kegel

Bild 4.20 veranschaulicht den implementierten Algorithmus durch einen Programmablaufplan.

#### 4.2.4 Bewertung

Der modifizierte Marching Triangles Algorithmus hat gegenüber den zuvor beschriebenen Verfahren den Vorteil, dass die Feldfunktion  $f(\vec{x})$  nur dann ausgewertet wird, wenn ein Punkt auf der Fläche generiert werden soll. Ein Abarbeiten des gesamten Volumens – und somit auch von Zellen, die komplett innerhalb oder ausserhalb der Fläche liegen – ist nicht notwendig, und die Punkte der Triangulierung befinden sich exakt auf der Fläche und werden nicht – wie z. B. beim Marching Cubes Algorithmus – durch Interpolation gewonnen.

Im Zusammenspiel mit dem PCM erweist sich als besonders vorteilhaft, dass die Größe

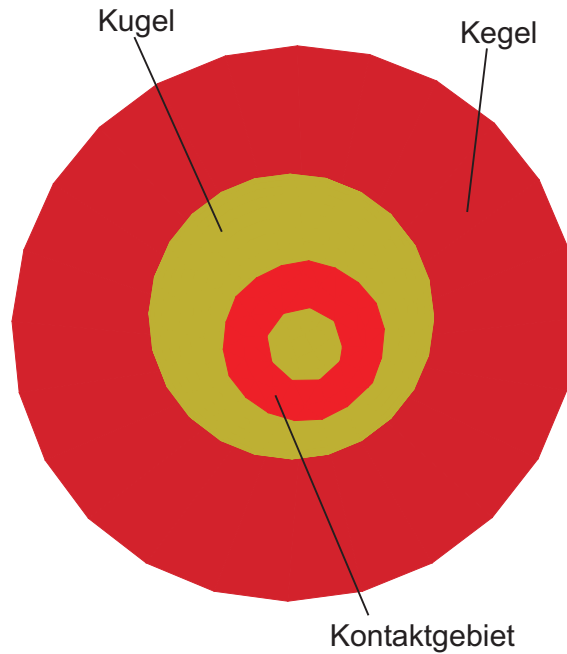


Bild 4.19: MKS-Modell (Kugel in Kegel)

der generierten Dreiecke sehr gleichmäßig ist und die Schnittpolygonzüge sehr einfach in Frontpolygone übergeführt werden können. Weiter können folgende Punkte zur Bewertung herangezogen werden:

### Vorteile

1. kohärentes Netz von Dreiecken
2. gleichmäßige Verteilung der Dreiecke über die Fläche
3. für beliebige implizite Flächen anwendbar
4. hohe Effizienz
5. einfache Datenstruktur

### Nachteile

1. nur für Flächen in impliziter Darstellung anwendbar
2. die Referenzlänge  $\delta_t$  muss für eine gute Annäherung möglichst klein sein
3. nicht anwendbar bei offenen Schnittpolygonzügen
4. kleine Änderungen der relativen Lage und Orientierung der Oberflächen können zu einer wesentlich veränderten Triangulierung führen



### 4.3 Konstruktion der Kontaktelemente

Im PCM wird zur Berechnung der Kontaktkraft eine diskretisierte Darstellung des Randschichtmodells (siehe Kapitel 1) verwendet. Die hierbei generierten *Kontaktelemente* entsprechen prinzipiell der gemäß Kapitel 3 approximierten und nach Kapitel 4 triangulierten Berührfläche.

Als Position eines Kontaktelements dient sein *Schwerpunkt*, der aus den Koordinaten seiner Eckpunkte gemittelt wird:

$$\vec{x}_{CE} = (\vec{x}_{p_1} + \vec{x}_{p_2} + \vec{x}_{p_3})/3 \quad (4.1)$$

An diesem Ort wird der durch das Element generierte Kontaktkraftvektor in die berührenden Körper eingepreßt. Außerdem wird für diese Stelle die Relativgeschwindigkeit bestimmt, die in die Kraftanteile aus normaler Dämpfung und tangentialer Reibung eingehen.

Die *Normale* des Kontaktelements wird ebenfalls aus seinen Eckpunkten berechnet:

$$\vec{n}_{CE} = \frac{\vec{r}_{p_1 p_2} \times \vec{r}_{p_1 p_3}}{\|\vec{r}_{p_1 p_2} \times \vec{r}_{p_1 p_3}\|} \quad (4.2)$$

Durch sie wird das Kontaktproblem in eine normale und eine tangentiale Komponente unterteilt, sodass sie die Richtung für Relativkinematik und Wirkung der entsprechenden Kraftgesetze festlegt.

Als wichtigste Eingangsgröße zur Bestimmung der Kontaktnormalkraft wird weiter die *Einsenkung*

$$u_{nCE} = \|\vec{r}_{q_1 q_2}\| \quad (4.3)$$

als Abstand der beiden Schnittpunkte der Normalen mit den unverformten Körperoberflächen bestimmt. Zur Berechnung der Schnittpunkte wird jeweils das Newton-Verfahren auf die nach Abschnitt 3.1 interpolierte Fläche  $f_i(x, y, z) = 0$  in impliziter Darstellung entlang der Normalen angewendet:

$\delta_0 := 0.0$

$\vec{x}_0 := \vec{x}_{CE}$

**repeat**

$$\delta_{k+1} := \delta_k - \frac{f(\vec{x}_k)}{\vec{n}_{CE} \cdot \nabla f(\vec{x}_k)}$$

$$\vec{x}_{k+1} := \vec{x}_{CE} + \delta_{k+1} \vec{n}_{CE}$$

**until**  $\|f(\vec{x}_{k+1})\| < \varepsilon$

$\vec{q}_i := \vec{x}_{k+1}$

Schließlich wird der *Flächeninhalt*

$$A_{CE} = \frac{1}{2} \|\vec{r}_{p_1 p_2} \times \vec{r}_{p_1 p_3}\| \quad (4.4)$$

des Kontaktelements berechnet, sodass sich seine Normalkraft nach (1.4) bei Annahme konstanter Spannung ergibt:

$$F_{nCE} = \frac{K}{b} A_{CE} u_{nCE} \quad (4.5)$$

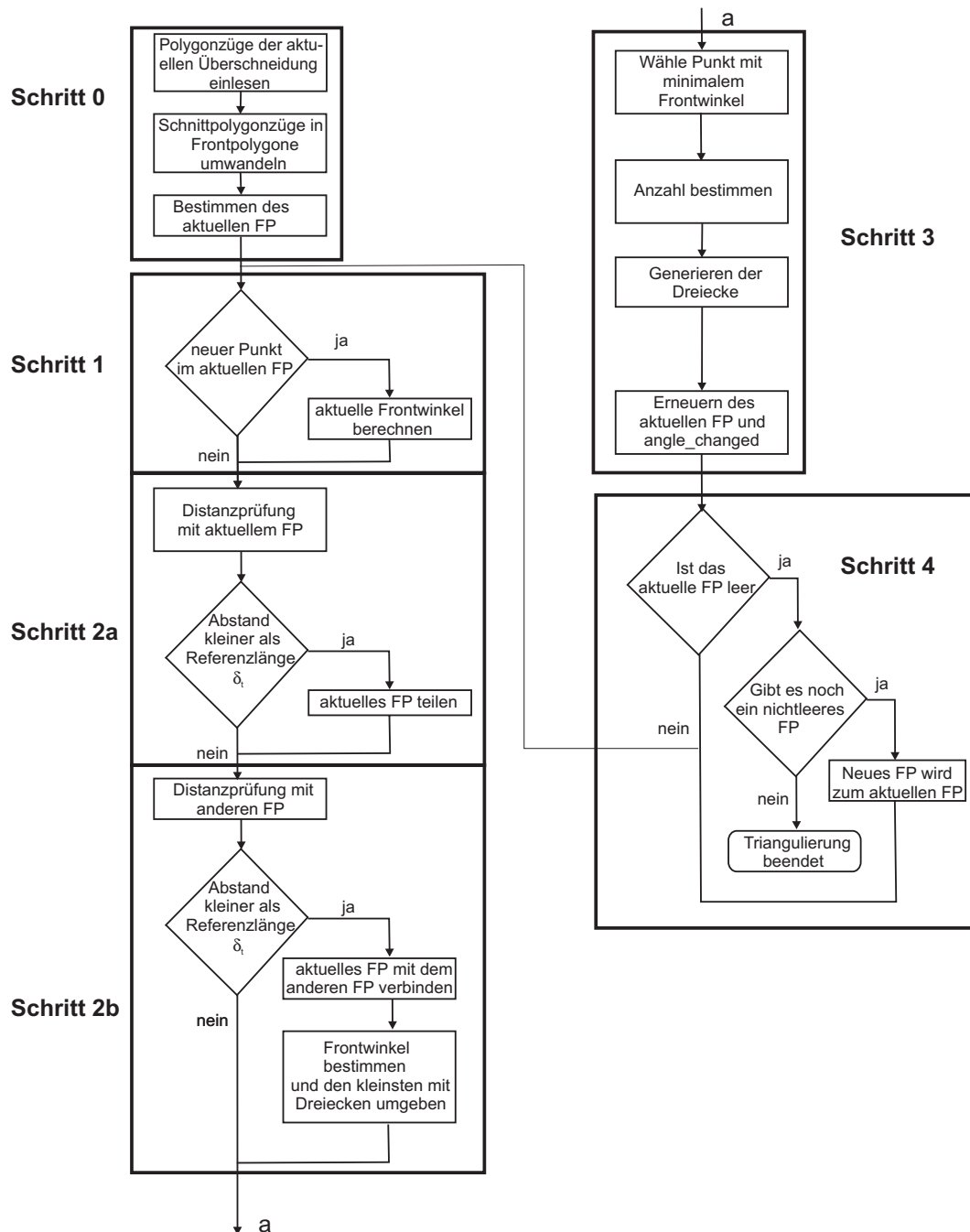


Bild 4.20: Programmablaufplan des modifizierten Marching Triangles Algorithmus

# Kapitel 5

## Anwendung

Zur Verifikation des in den vorigen Kapiteln beschriebenen Verfahrens zur Generierung von Kontaktelementen wurde das in Bild 5.1 dargestellte MKS-Modell einer Zahnradgetriebestufe mit gerader Evolventenverzahnung herangezogen. Modellaufbau und Simulation wurden mit dem Programm *SIMPACK* durchgeführt, wobei das PCM als User Routine eingebunden wurde.

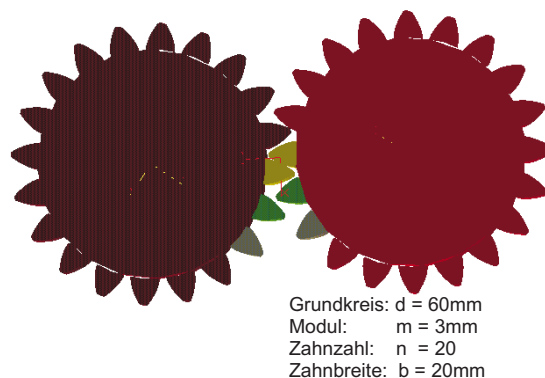


Bild 5.1: MKS-Modell eines geraden Zahnradgetriebes

Das antreibende Rad wird rheonom mit konstanter Winkelgeschwindigkeit von  $\omega = 1.5 \text{ rad/s}$  bewegt. Parallel zum Lager des Abtriebs wurde ein linearer Drehdämpfer mit der Dämpfungskonstante  $d = 5.0 \text{ Nms/rad}$  modelliert. Mit dem Übersetzungsverhältnis  $i = 1$  ergäbe sich somit bei ideal gleichförmigem Lauf ein konstantes An- und Abtriebsmoment von  $7.5 \text{ Nm}$ .

Die Oberfläche der Zahnflanken wurden als Dreiecksfläche generiert (Bild 5.2) und mit der erforderlichen Einbauposition und -lage jeweils 40-fach an den Radkörpern angebracht. Zwischen den korrespondierenden Zahnpaarungen wurden identische PCM-Kontakte definiert, bei denen die angenommene Kontaktfläche wie gezeigt durch Variational Interpolation approximiert und durch den modifizierten Marching Triangles Algorithmus trianguliert wurde. Im Randschichtmodell wurde für die Normalkraftberechnung in den Kontaktelementen  $F_{CE} = 1.154 \cdot 10^{12} \text{ N/m}^3 \cdot A_{CE} \cdot u_{CE}$  angenommen, was einer weichen Randschicht ( $E = 2.1 \text{ MPa}$ ,  $\nu = 0.2$ ) von  $2 \text{ mm}$  Dicke entspricht und Materialdämpfung und Reibung vernachlässigt.

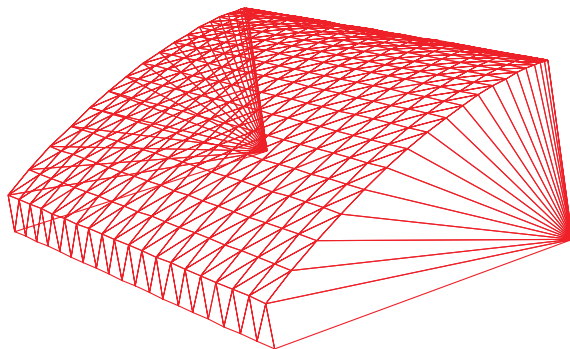


Bild 5.2: Geometrie einer Zahnflanke als Polygonfläche

Bei den ersten Simulationen zeigten sich einige Schwächen des neuen Verfahrens. Zunächst führte eine zu „weich“ angenommene Kontaktsteifigkeit zu so großen Eindringungen, dass sich das Kontaktgebiet über die Ränder der Flankenflächen hinaus ausbreitete und sich so ein offener Schnittpolygonzug ergab. Weil dann die Frontpolygone der Triangulierung nicht mehr geschlossen waren, fehlte im Algorithmus das Abbruchkriterium. Diese Schwierigkeit lässt sich vermeiden, indem man die Kontaktflächen ausreichend groß modelliert oder mit geschlossenen Flächen arbeitet, wie sie in der Wirklichkeit ausschliesslich vorkommen.

Ein schwerwiegenderes Problem stellt die Tatsache dar, dass das PCM im Zusammenspiel mit der neuen Kontaktelementgenerierung praktisch keine Anwendung moderner Integrationsverfahren mit Korrekturiteration zur Lösung der Bewegungsgleichungen erlaubt. Vermutlich führt das „Umkippen“ der Triangulierung bei minimalen Änderungen des Systemzustands dazu, dass keine sinnvolle Berechnung der Jakobimatrix mehr möglich ist und die Schrittweitensteuerung zu extrem kleinen Zeitschritten gezwungen wird. Aus diesem Grund mussten sämtliche Simulationen mit dem expliziten Eulerverfahren integriert werden.

Schließlich konnte auch die Effizienz von Approximation und Triangulierung der Kontaktfläche nicht überzeugen. In einem Funktionsaufruf der Integration benötigte die Berechnung für das recht einfache Modell der Verzahnung Zeiten in der Größenordnung einer Sekunde. Bisher wurde nicht untersucht, wie sich die Rechenzeit auf die Module des Algorithmus aufteilt.

Die Bilder 5.3 bis 5.6 zeigen Ergebnisplots für die ersten drei Eingriffe der Verzahnung. Bild 5.3 stellt Winkellage und -geschwindigkeit der Räder dar, in Bild 5.4 sieht man die zugehörigen Beschleunigungen und Drehmomente. Die Verläufe der abgeleiteten Größen lassen die leichte Ungleichförmigkeit der Verzahnung erkennen, wobei die beiden größeren Überschwinger neuen Zahneingriffen zugeordnet werden können. Die unterschiedlichen Drehmomente zwischen An- und Abtrieb lassen sich durch die Dynamik des Abtriebsrads (Trägheitsmoment  $10^{-3} \text{kgm}^2$ ) erklären. Die Spitzen in der Gesamtkraft der Kontakte zwischen den Zähnen (Bild 5.5) lassen auf Zustände schliessen, die zu fehlerhaftem Verhalten des Algorithmus führen. Die drei kleineren Aufschwingungen beim Abwälzen jeder Zahnpaarung hängen damit zusammen, dass an diesen Stellen neue Dreiecksreihen auf den Flanken aktiv werden, sodass die Geometrie der approximierten Kontaktfläche „umklappt“. Schließlich zeigt Bild 5.6 die Anzahl der Linien der Schnittpolygonzüge und der

generierten Kontaktelemente.

Für die bisher gezeigten Ergebnisse wurden die Flanken durch Dreiecksflächen mit  $n_D = 200$  Dreiecken repräsentiert. Als Referenzlänge der Triangulierung wurde mit  $\delta_t = 2mm$  gearbeitet, und die Schrittweite der Zeitintegration betrug  $\Delta t = 0.05ms$ . Die  $0.5s$  Echtzeit entsprechende Simulation dauerte ca. 36000 s CPU-Zeit.

Um den Einfluss der Feinheit der die Zahnflankenoberflächen beschreibenden Dreiecksflächen zu untersuchen, wurden zwei weitere Rechnungen mit  $n_D = 200$  und  $n_D = 400$  durchgeführt ( $\delta_t = 1mm, \Delta t = 0.2ms$ ). Im Vergleich der Winkelbeschleunigung des Abtriebsrads (Bild 5.7) und der Anzahl der Kontaktelemente (Bild 5.8) ist offensichtlich, dass feiner diskretisierte Flanken die Stabilität des Verfahrens erheblich verbessern. Die Störungen rühren anscheinend vom „Umspringen“ der angenommenen Kontaktfläche her, wenn die Flanken aufeinander abrollen und eine neuen Reihe von Dreiecken vom passiven zum aktiven Zustand wechselt. Damit erweist sich ein wesentliches Motiv der Kontaktflächenapproximierung über Variational Interpolation, nämlich die weitgehende Unabhängigkeit der Größe der Kontaktelemente von der Triangulierung der verwendeten Oberflächen, als nicht erfüllt. Allerdings stellt die vorliegende Anordnung einen disbezüglich besonders ungünstigen Fall dar; bauen sich die aktiven Flächen dagegen „kontinuierlich“ Dreieck für Dreieck auf und ab, treten die Störungen in den Hintergrund.

Einen weiteren Vergleich, der Aufschluss über den Einfluss der Referenzlänge  $\delta_t$  der Triangulierung gibt, zeigen die Bilder 5.10 und 5.9. Bei  $n_D = 200$  und  $\Delta t = 0.2ms$  lässt sich keine signifikante Verbesserung erkennen, wenn  $\delta_t$  von  $4mm$  auf  $2mm$  verkleinert wird.

Schließlich wurde eine weitere Vergleichsrechnung mit unterschiedlichen Schrittweiten  $\Delta t = 0.05ms$  und  $\Delta t = 0.2ms$  des Eulerverfahrens durchgeführt ( $n_D = 200, \delta_t = 2mm$ ). Der Verläufe der Winkelgeschwindigkeit (Bild 5.11) und Beschleunigung (Bild 5.12) des Abtriebsrads zeigen deutlich, dass die Schrittweite die Ergebnisse sehr stark beeinflusst und mit  $0.2ms$  bereits keine sinnvollen Resultate mehr erzielt werden können. Offenbar tritt beim vorliegenden, recht steifen Differentialgleichungssystem die Stabilitätsgrenze des Eulerverfahrens in Erscheinung.

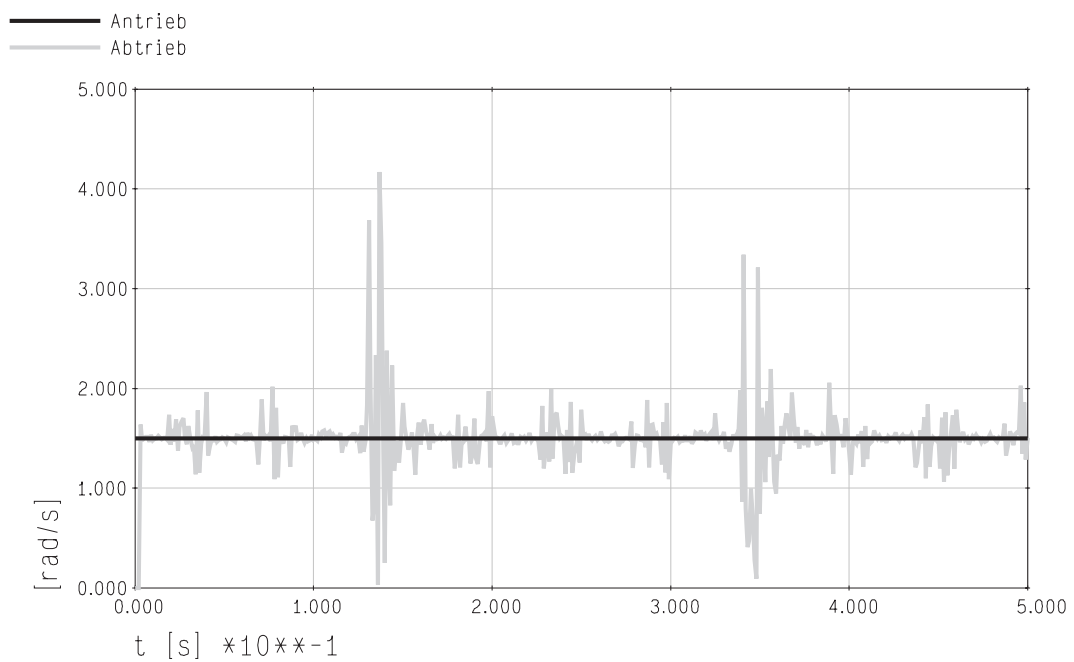
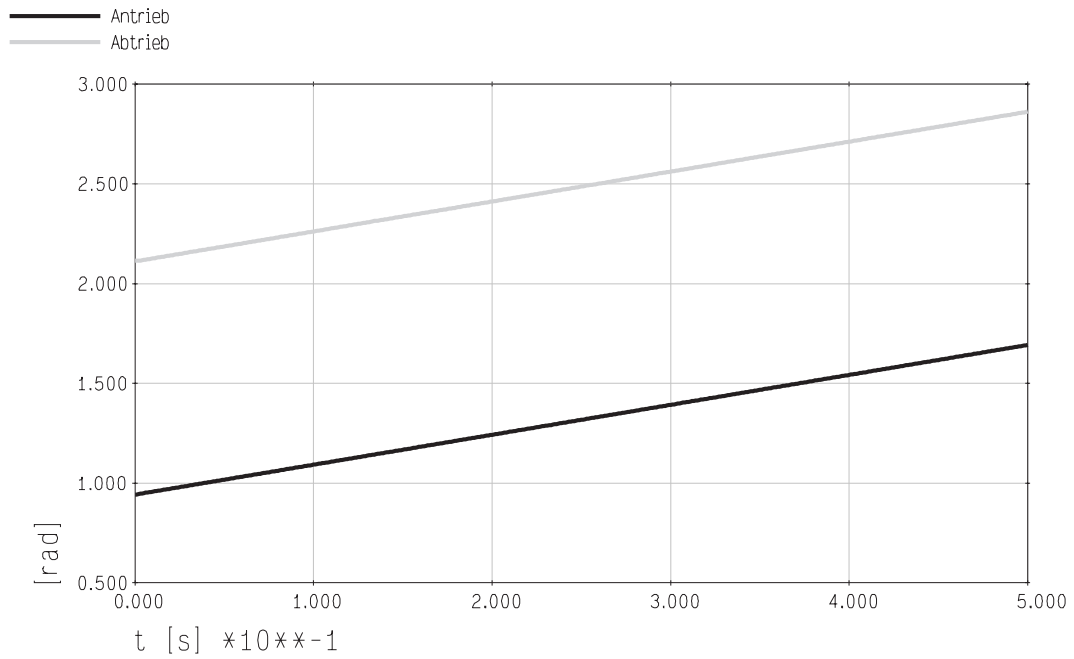


Bild 5.3: Winkellage und -geschwindigkeit von Antriebs- und Abtriebsrad.

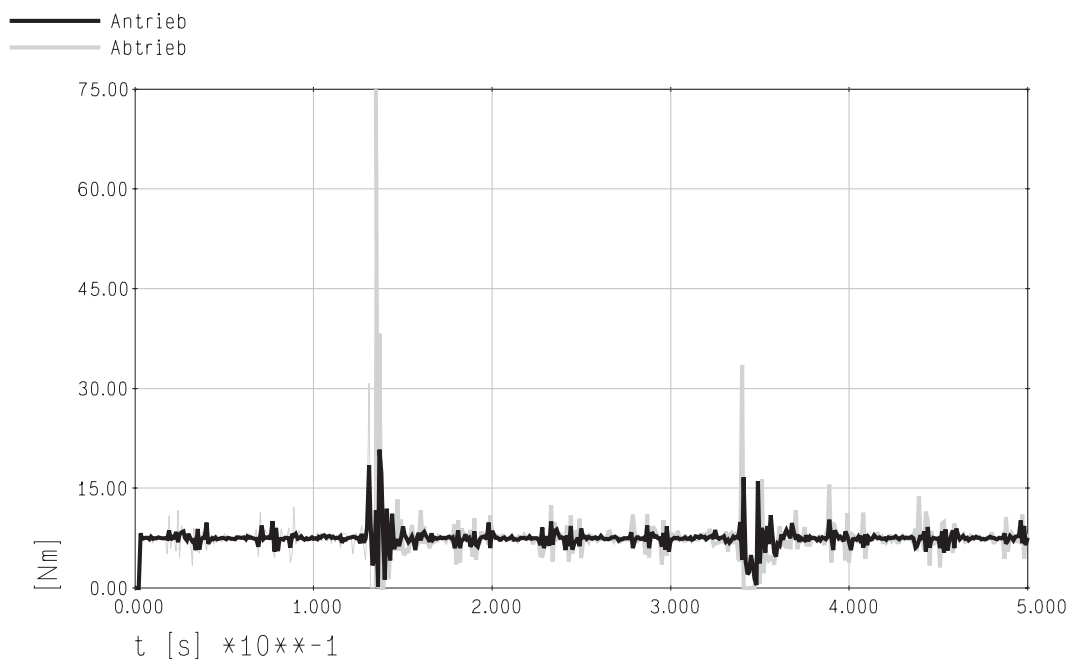
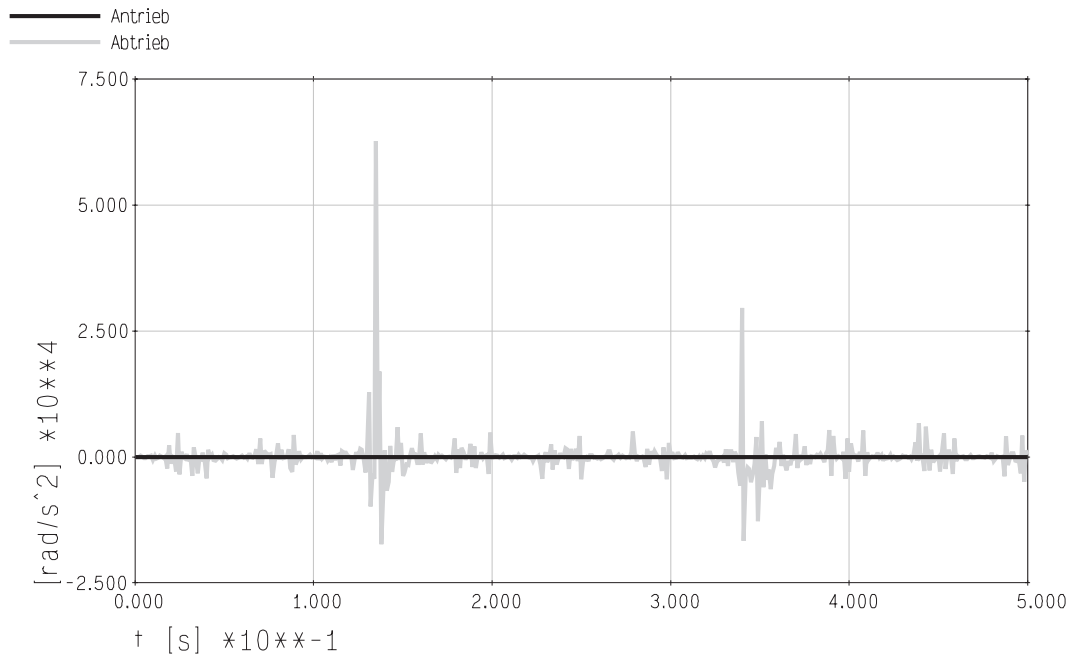


Bild 5.4: Winkelbeschleunigung und Drehmoment von Antriebs- und Abtriebsrad.



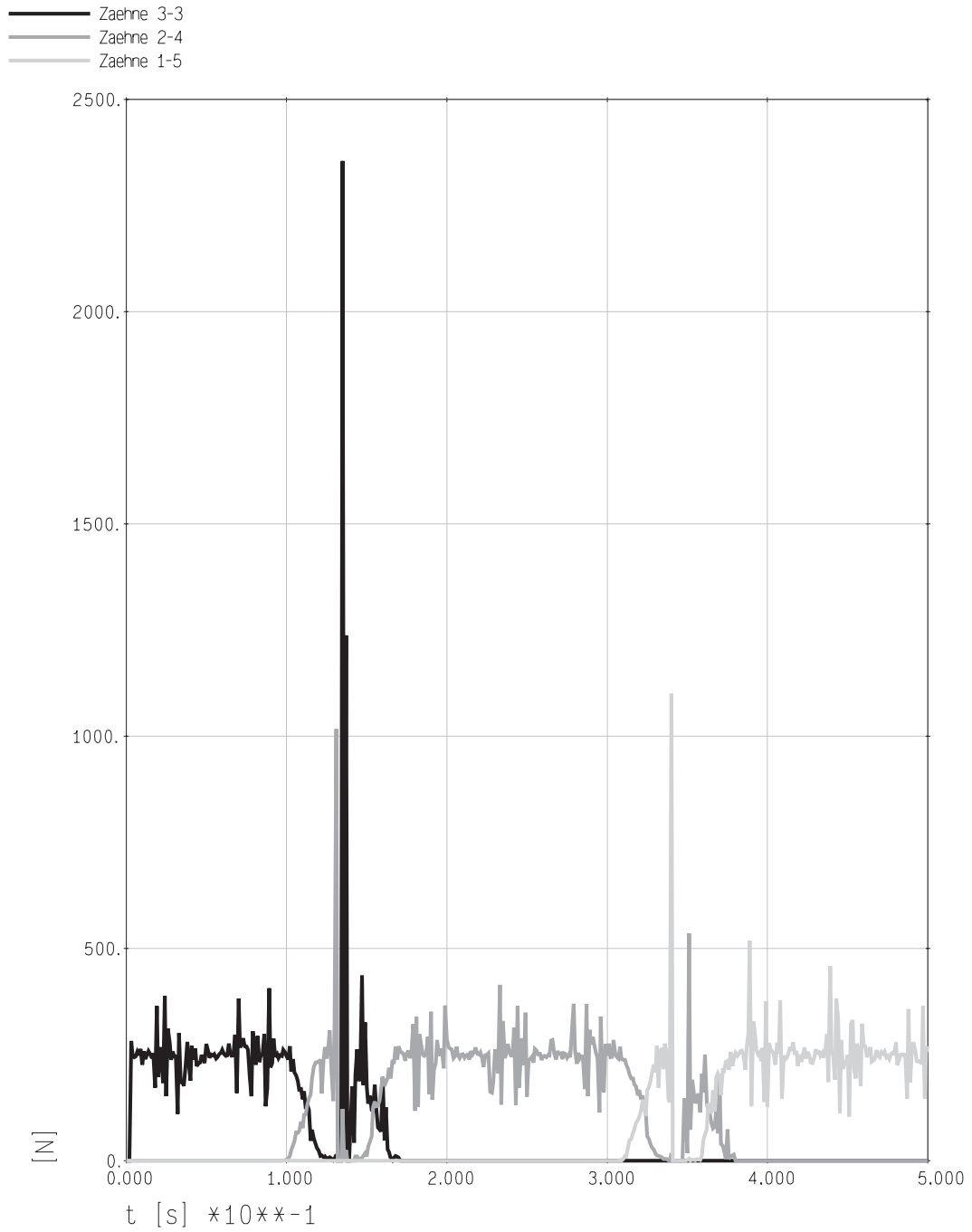


Bild 5.5: Betrag der Kontaktkräfte.

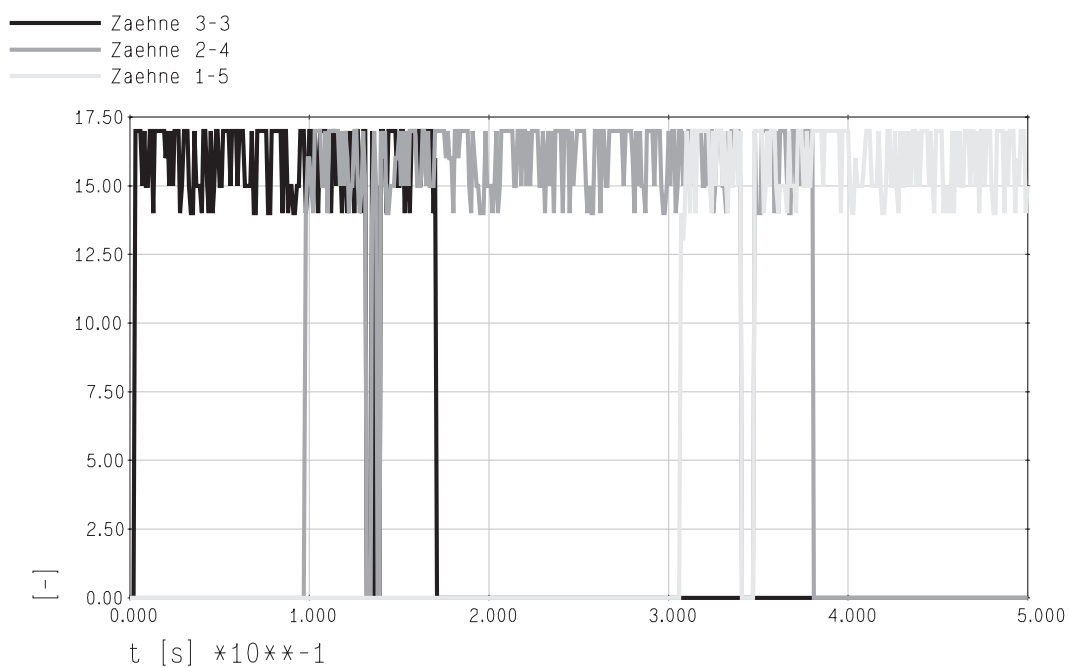
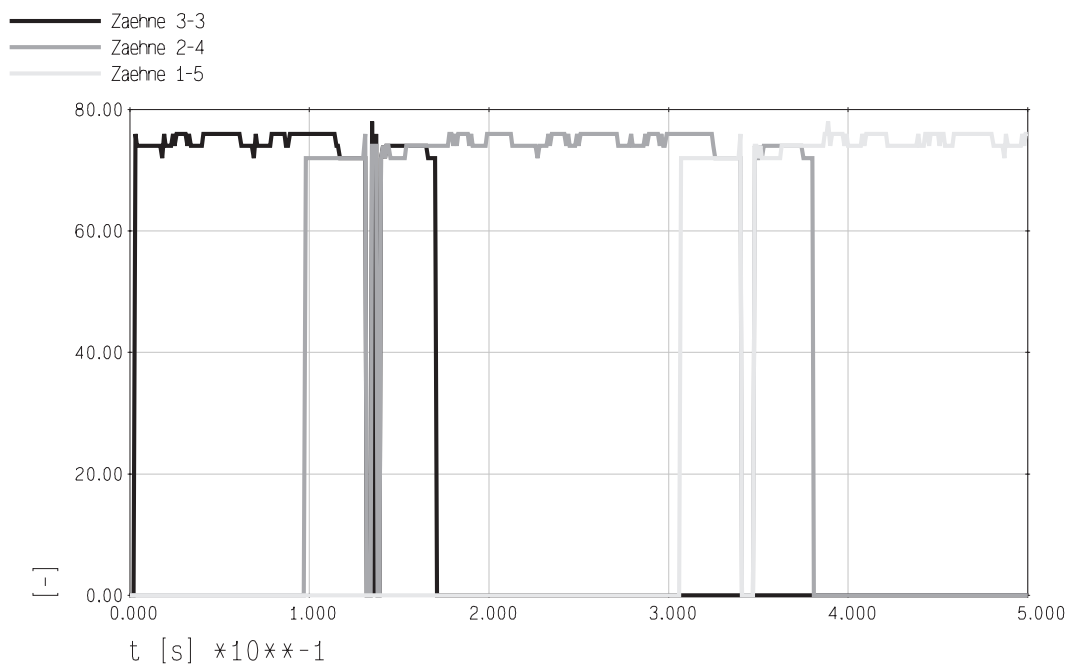


Bild 5.6: Anzahl der Linien des Schnittpolygonzugs und der Kontaktelemente

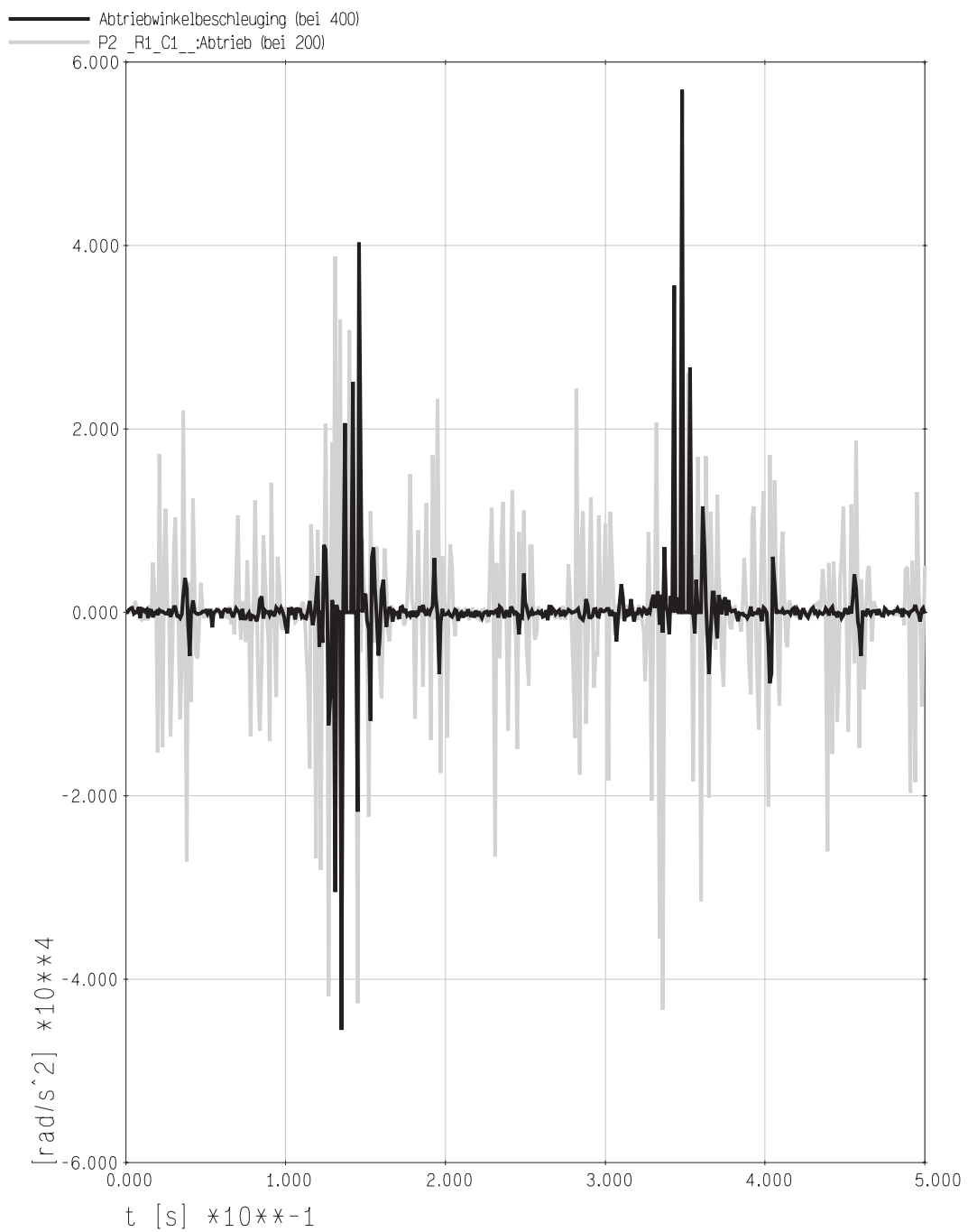


Bild 5.7: Vergleich der Winkelbeschleunigung des Abtriebsrads bei Verwendung von Zahnflanken mit  $n_D = 200$  und  $n_D = 400$  Dreiecken ( $\delta_t = 1\text{mm}$ ,  $\Delta t = 0.2\text{ms}$ ).

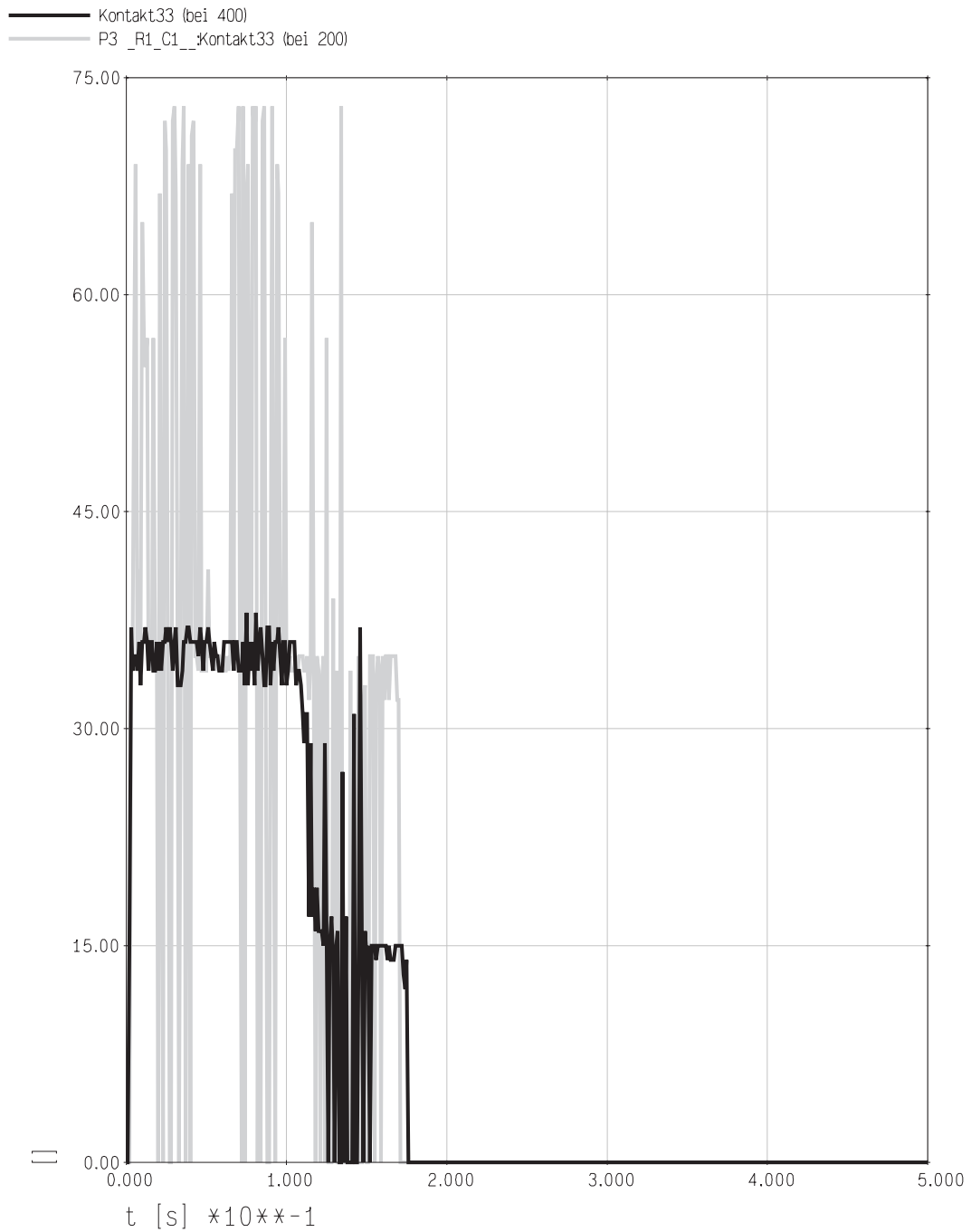


Bild 5.8: Vergleich der Anzahl der Kontaktelemente bei Verwendung von Zahnflanken mit  $n_D = 200$  und  $n_D = 400$  Dreiecken ( $\delta_t = 1mm$ ,  $\Delta t = 0.2ms$ ).

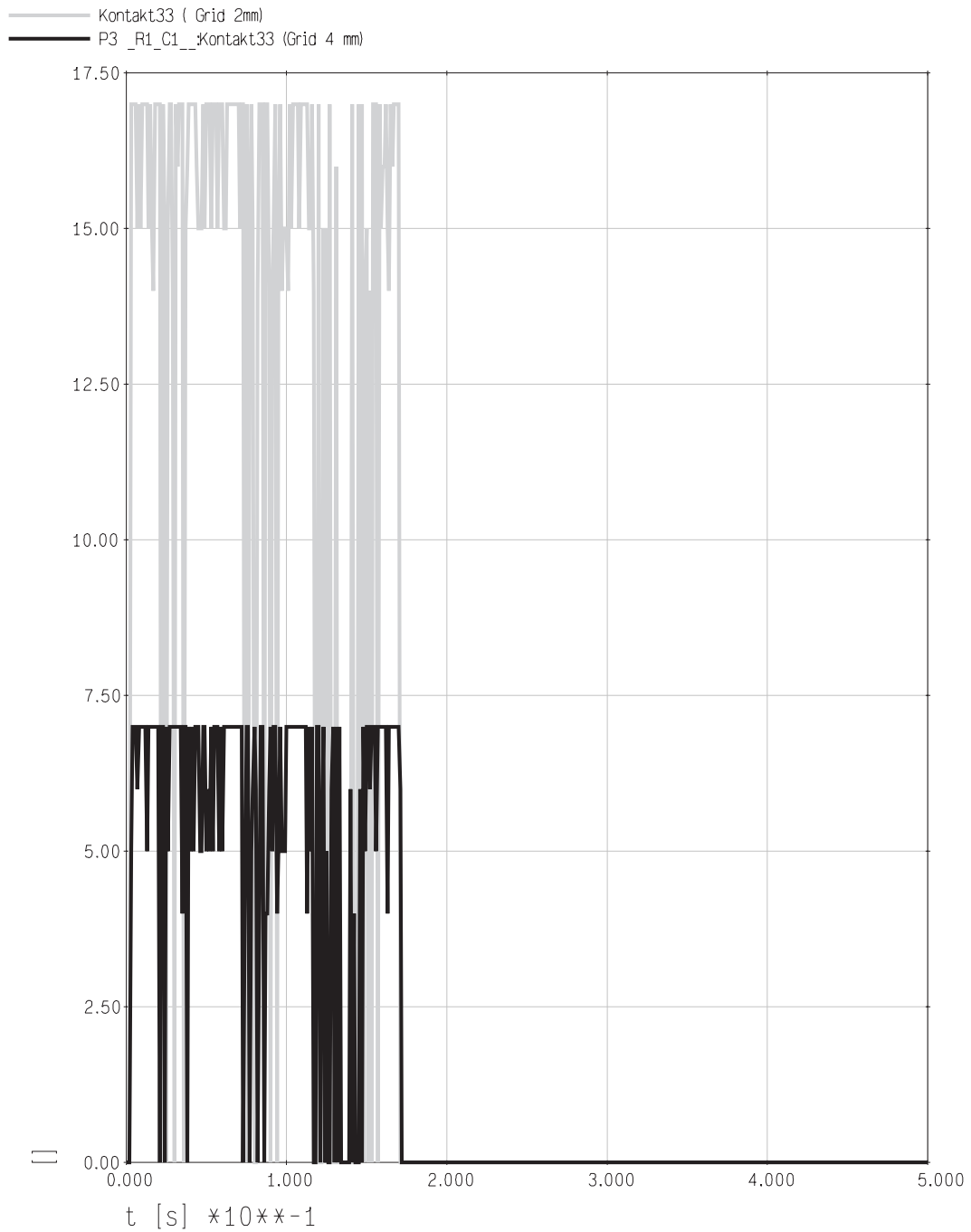


Bild 5.9: Vergleich der Anzahl der Kontaktelemente bei Triangulierung mit  $\delta_t = 2mm$  und  $\delta_t = 4mm$  ( $n_D = 200$ ,  $\Delta t = 0.2ms$ ).

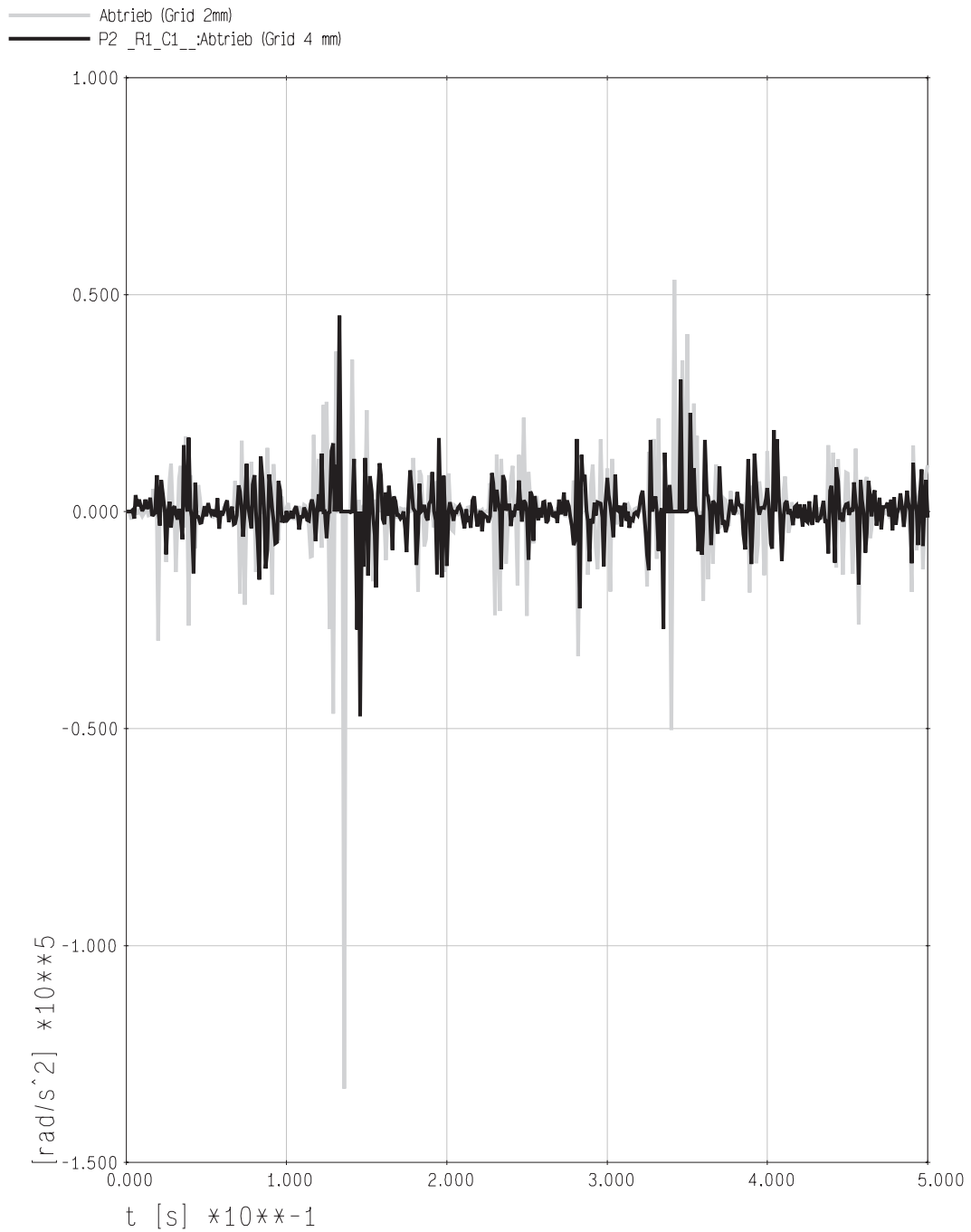


Bild 5.10: Vergleich der Winkelbeschleunigung des Abtriebsrads bei Triangulierung mit  $\delta_t = 2\text{mm}$  und  $\delta_t = 4\text{mm}$  ( $n_D = 200$ ,  $\Delta t = 0.2\text{ms}$ ).

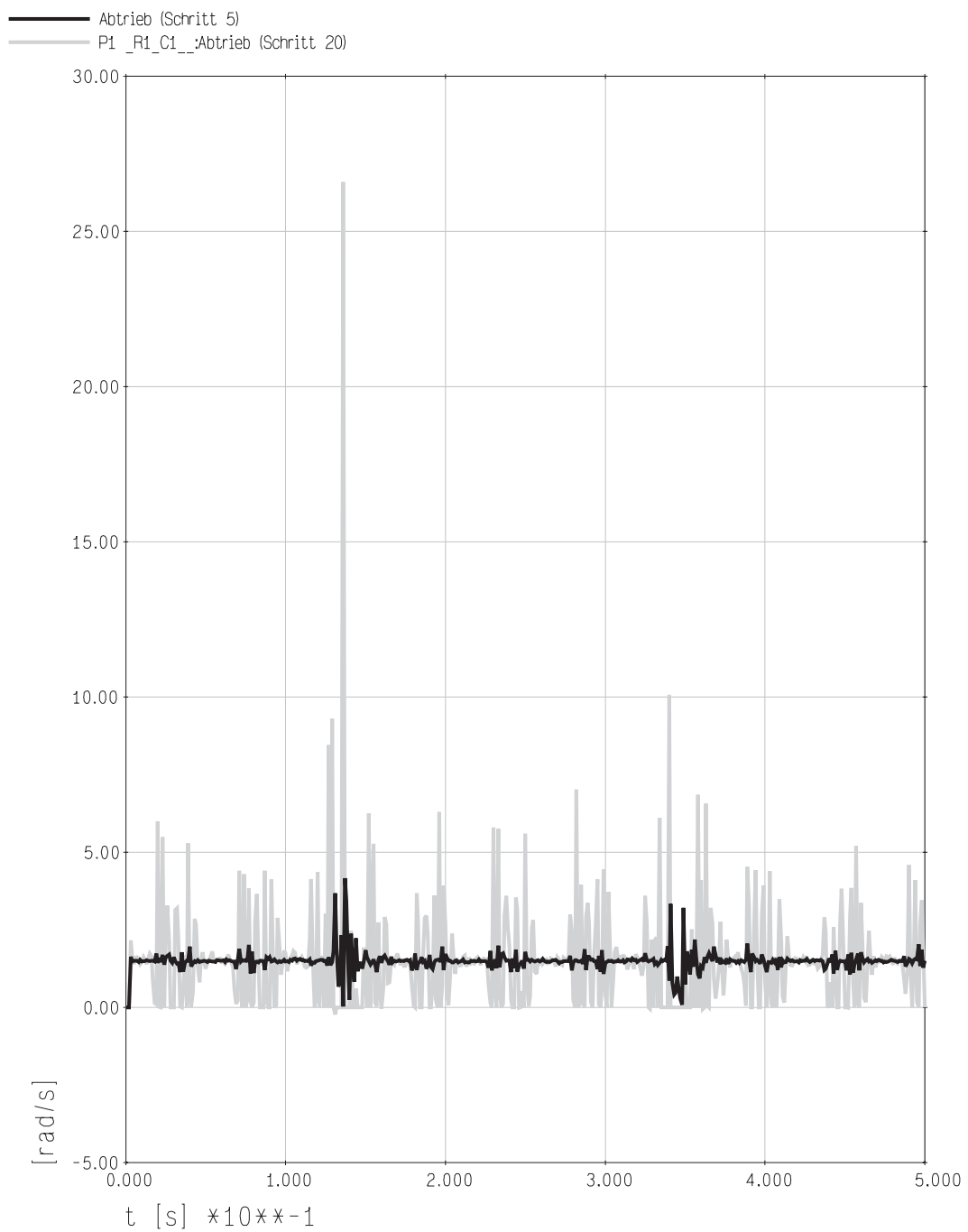


Bild 5.11: Vergleich der Winkelgeschwindigkeit des Abtriebsrads bei Zeitschrittweiten von  $\Delta t = 0.05ms$  und  $\Delta t = 0.2ms$  ( $n_D = 200, \delta_t = 2mm$ ).

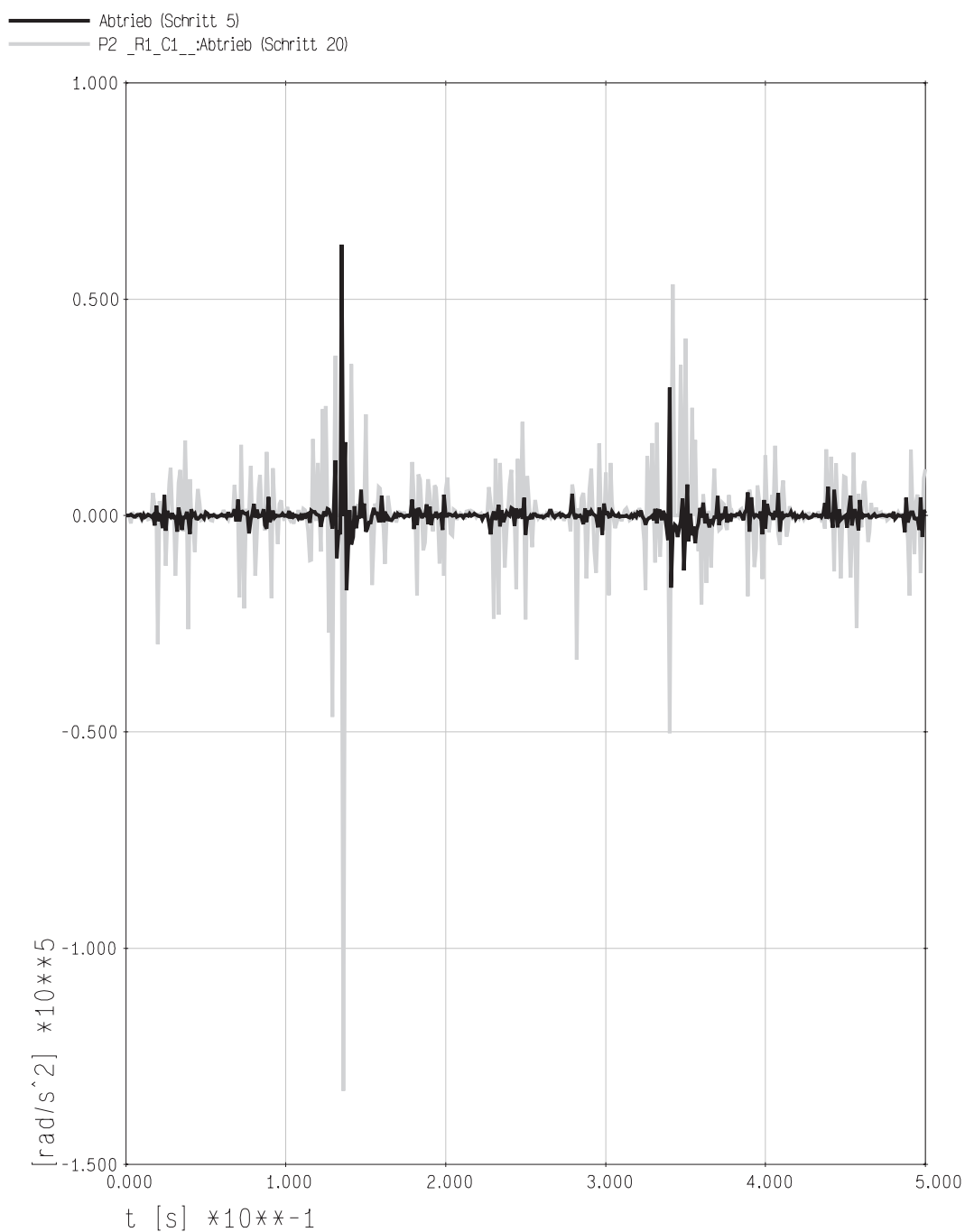


Bild 5.12: Vergleich der Winkelbeschleunigung des Abtriebsrads bei Zeitschrittweiten von  $\Delta t = 0.05ms$  und  $\Delta t = 0.2ms$  ( $n_D = 200, \delta_t = 2mm$ ).





# Kapitel 6

## Zusammenfassung

Die Arbeit beschreibt Methoden und Implementierung zur Approximation und Diskretisierung der Berührfläche zweier Körper im Rahmen der Mehrkörperdynamik. Ausgegangen wird dabei vom Schnittvolumen der unverformten Oberflächen, dargestellt durch Dreiecksflächen.

In Kapitel 3 wird gezeigt, wie anhand der im Schnitt befindlichen „aktiven“ Dreiecke und dem Schnittpolygonzug eine Näherung für die Berührfläche konstruiert werden kann. Dazu werden die beiden aktiven Flächengebiete zunächst durch Funktionen in impliziter Darstellung interpoliert. Dann wird die Berührfläche unter Berücksichtigung des Verhältnisses der Elastizitätsmoduli der beiden Körper durch Linearkombination gebildet.

Kapitel 4 stellt ein Verfahren vor, mit dem die in impliziter Darstellung vorliegende Berührfläche durch Dreiecke diskretisiert werden kann. Dazu wird eine modifizierte Version des flächenbasierten Marching Triangles Algorithmus erarbeitet, die auf die Anwendung im Kontaktmodell zugeschnitten und implementiert wurde. Außerdem wird gezeigt, wie aus der Diskretisierung Kontaktelemente gebildet werden, die eine Kontaktmodellierung durch das Randschichtmodell ermöglichen.

In Kapitel 5 werden Ergebnisse und Erfahrungen aus der Simulation eines Anwendungsbeispiels präsentiert. Anhand von Parameterstudien wird der Einfluss der Feinheit der Dreiecksflächen der Körper und der Diskretisierung der Berührfläche sowie der Zeitschrittweite der Zeitintegration untersucht.

Auf Grund der aktuellen Rechenleistung muss zunächst auf eine allgemeingültigen Anwendung von PCM mit Variational Interpolation verzichtet werden, jedoch kann in Zukunft bei gesteigerter Rechenleistung nochmals auf die Methode zurückgegriffen werden.



# Anhang A

## Numerische Grundlagen

### A.1 Matrizen

#### A.1.1 Symmetrische Matrix

DEFINITION. Eine Matrix  $A \in \mathbb{R}^{n \times n}$  heisst **symmetrisch**, wenn gilt:

$$A = A^T$$

SATZ. Sei  $A$  eine symmetrische  $n \times n$  Matrix über  $\mathbb{R}$ . Dann gilt:

1. Alle Eigenwerte von  $A$  sind reell.
2. Für jeden Eigenwert stimmen algebraische und geometrische Vielfachheit überein.
3.  $A$  ist diagonalisierbar.
4. Es gibt eine Basis des  $\mathbb{R}^n$ , die aus paarweisen Eigenvektoren von  $A$  besteht.

#### A.1.2 Positiv/negativ definite Matrix

DEFINITION. Sei  $A$  eine symmetrische  $n \times n$  Matrix über  $\mathbb{R}$ .  $A$  heisst **positiv definit** bzw. **negativ definit**, wenn

$$\vec{x}^T A \vec{x} > 0 \quad \text{bzw.} \quad \vec{x}^T A \vec{x} < 0$$

für alle  $\vec{x} \in \mathbb{R}^n$  mit  $\vec{x} \neq \vec{0}$ .

Bezeichnet „ $\cdot$ “ das Skalarprodukt auf  $\mathbb{R}^n$ , so ist:

$$\vec{x}^T A \vec{x} = \vec{x} \cdot (A \vec{x})$$

Gilt  $\vec{x}^T A \vec{x} \leq 0$  bzw.  $\vec{x}^T A \vec{x} \geq 0$ , so heisst die symmetrische Matrix **positiv semi definit** bzw. **negativ semidefinit**. Gibt es Vektoren  $\vec{x}, \vec{y} \in \mathbb{R}^n$ , sodass  $\vec{x}^T A \vec{x} \leq 0$  und  $\vec{y}^T A \vec{y} \geq 0$ ,

so heisst die Matrix **indefinit**. Der folgende Satz liefert ein Kriterium, ob eine Matrix  $A$  positiv definit ist:

SATZ. Eine symmetrische Matrix  $A \in \mathbb{R}^n$  ist genau dann positiv definit, wenn die Determinanten der  $n$  Hauptmatrizen  $H_i$ ,  $1 \leq i \leq n$ , d. h. von

$$H_1 = a_{11} \text{ , } H_2 = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \text{ , } \dots \text{ , } H_k = \begin{pmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & \ddots & \vdots \\ a_{k1} & \cdots & a_{kk} \end{pmatrix} \text{ , } H_n = A$$

alle positiv sind.

Der nächste Satz stellt eine Verbindung zwischen der Eigenschaften „positiv (negativ) (semi) definit bzw. indefinit“ und den Eigenwerten her:

SATZ. Eine symmetrische Matrix  $A \in \mathbb{R}^{n \times n}$  ist genau dann positiv definit (bzw. negativ definit), wenn alle ihre Eigenwerte positiv (bzw. negativ) sind. Sie ist genau dann positiv semidefinit (bzw. negativ semidefinit), wenn alle ihre Eigenwerte nicht negativ (bzw. nicht positiv) sind. Sie ist genau indefinit, wenn sie positive und negative Eigenwerte besitzt.

## A.2 Lösung symmetrischer, semipositiver Linearsysteme

Das Gleichungssystem (2.12) ist symmetrisch und semipositiv definit und kann Koeffizientenmatrizen mit sehr großer Dimension besitzen. Da die Zahl der Elemente dieser Matrizen quadratisch mit der Zahl der Unbekannten wächst, erreicht man sehr schnell die Speichergrenzen des Hauptspeichers. Es stellt sich somit die Frage nach angepassten Abspeicherungsmethoden für auftretenden Matrizen. Ist die Koeffizientenmatrix symmetrisch und semipositiv definit, reicht es aus, wenn man alle Elemente abspeichert, die auf der Hauptdiagonalen stehen oder sich rechts von ihr befinden. Man spricht dann von der oberen Dreiecksmatrix [?].

Für lineare Gleichungssysteme gibt es eine Reihe von Lösungsmethoden. Sogar manche nichtlinearen Probleme werden deshalb mit verschiedenen Verfahren (z. B. durch die Taylor-Reihenentwicklung) linearisiert, d. h. durch ein lineares Gleichungssystem angenähert, um Lösbarkeit zu erreichen.

Für die numerische Lösung von linearen Gleichungssystemen unterscheidet man direkte Verfahren, Iterationsverfahren und Verfahren für Systeme mit speziellen Eigenschaften. Ein sehr einfaches direktes Verfahren ist das LU-Verfahren. LU-Zerlegung bedeutet, dass mit dem Eliminationsverfahren nach Gauss die Systemmatrix  $A$  in ein Produkt  $A = LU$  einer unteren Dreiecksmatrix  $L$  und einer oberen Dreiecksmatrix  $U$  zerlegt wird. Ist nun eine LU-Zerlegung gegeben, dann kann man das lineare Gleichungssystem  $A\mathbf{x} = \mathbf{b}$  auch als  $LU\mathbf{x} = \mathbf{b}$  schreiben. Damit lässt sich das untere Dreieckssystem  $L\mathbf{y} = \mathbf{b}$  durch Vorwärtseinsetzen und das obere Dreieckssystem  $U\mathbf{x} = \mathbf{y}$  durch Rückwärtseinsetzen lösen.

### A.2.1 LU-Zerlegung mit Elimination nach Gauß

Ziel des Gaußschen Eliminationsverfahrens ist es, ein lineares Gleichungssystem in ein äquivalentes dreiecksförmiges System zu überführen. Die quadratische Systemmatrix wird dabei in eine dreiecksförmige Matrix transformiert, ohne dass sich dadurch die Lösungsmenge ändert. Das Verfahren beruht auf der Tatsache, dass man ein Gleichungssystem mit gleicher Lösung erhält, wenn man das Vielfache einer Zeile zu einer anderen Zeile addiert [2].

#### Berechnung der Obermatrix:

Durch systematische Addition der Vielfachen von Zeilen kann man erreichen, dass alle Koeffizienten  $a_{ij}$  mit  $j < i$ , unterhalb der Hauptdiagonalen zu Null werden. Addiert man das  $f$ -fache der ersten Zeile zur zweiten mit  $f = -a_{21}/a_{11}$ <sup>1</sup>, dann erhält man als neue zweite Zeile [3]:

$$\begin{aligned} \left( a_{21} + \frac{-a_{21}}{a_{11}} a_{11} \right) x_1 + \left( a_{22} + \frac{-a_{21}}{a_{11}} a_{12} \right) x_2 + \cdots + \left( a_{2n} + \frac{-a_{21}}{a_{11}} a_{1n} \right) x_n \\ = b_2 + \frac{-a_{21}}{a_{11}} b_1 \end{aligned}$$

Wenn man die Koeffizienten der neuen Zeile mit  $a'_{ij}$  bezeichnet und die neue rechte Seite mit  $b'_2$ , dann erhält man:

$$0x_1 + a'_{22}x_2 + \cdots + a'_{2N}x_N = b'_2$$

Analog verfährt man mit den Zeilen 3 ... n, bis man alle Koeffizienten unter  $a_{11}$  Null gesetzt hat. Ebenso verfährt man mit der zweiten Spalte für die Koeffizienten unterhalb  $a_{22}$  usw., bis man schließlich ein System in Dreiecksform erhält:

$$U\vec{x} := \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a'_{22} & \cdots & a'_{2n} \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & a'_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b'_2 \\ \vdots \\ b'_n \end{bmatrix}$$

#### Pivotisierung:

Wir nehmen im obenliegendem Verfahren an, dass das Gleichungssystem eine eindeutige Lösung besitzt und dass die Pivotelemente  $a_{ii}$  auf der Hauptdiagonalen verschieden von Null sind. Theoretisch reicht es aus, jede von Null verschiedene Zahl als Pivotelement für das Eliminationsverfahren zu wählen. Praktisch jedoch sind kleine Pivotelemente problematisch, denn sie führen zu großen Multiplikatoren (da durch das Pivotelement dividiert wird). Genaue Lösungen zu bekommen bedeutet aber, unnötig große Zahlen zu vermeiden. Aus diesem Grund strebt man an, die Multiplikatoren nicht größer als 1 werden zu lassen. Dies ist möglich, wenn man die größte Zahl auf oder unterhalb der

<sup>1</sup> $f$  heißt Multiplikator und  $a_{11}$  heißt Pivotelement.

Diagonalen als Pivotelement wählt. Diese Strategie nennt man teilweise oder partielle Pivitisierung. Sie ermöglicht es, die numerische Stabilität des Gauss-Eliminationverfahrens zu verbessern. Eine partielle Pivitisierung erfordert Zeilenvertauschungen. Der Name teilweise Pivitisierung kommt daher, dass das größte Element nur aus der gegenwärtigen Spalte ausgewählt wird. Wählt man das Pivoelement als größtes Element in der verbleibenden reduzierten Untermatrix, so spricht man von vollständiger Pivitisierung. Diese verwendet bei Bedarf auch Spaltenvertauschungen, die als Unternumerierung gedeutet werden können [6].

### A.2.2 Rücksubstitution

Von dreiecksförmigen linearen Gleichungssystemen wissen wir, daß sie sich durch sukzessives Einsetzen leicht lösen lassen, und dass reguläre lineare Abbildungen die Lösungsmenge nicht verändern.

Aus der letzten Zeile der Obermatrix kann man  $x_n$  berechnen:

$$x_n = \frac{b'_n}{a'_{nn}} \quad (\text{A.1})$$

Mit bekanntem  $x_n$  erhält man  $x_{n-1}$  durch Einsetzen in die Zeile  $n - 1$  :

$$x_{n-1} = \frac{b'_{n-1} - a'_{(n-1)n}x_n}{a'_{(n-1)(n-1)}} \quad (\text{A.2})$$

Allgemein gilt:

$$x_i = \frac{b'_i - \sum_{j=i+1}^n a'_{ij}x_j}{a'_{ii}} \quad (\text{A.3})$$

## A.3 Newton–Verfahren

Die im Abschnitt (4.2.2) verwendete Gleichung (A.12), die einen einem gegebenen Punkt  $\vec{q}$  nahe gelegenen Flächenpunkt  $\vec{p}$  bestimmt, wird hier kurz hergeleitet:

Eine implizite Funktion  $f(\vec{u}) = 0$  wird diskretisiert dargestellt:

$$f(\vec{u}_{k+1}) = f(\vec{u}_k + \Delta\vec{u}) \quad (\text{A.4})$$

Dabei nimmt man an:

$$\Delta\vec{u} = \nabla f(\vec{u}_k) \cdot t \quad (\text{A.5})$$

Weiter wird eine äquivalente implizite Hilfsfunktion mit einer einzigen Variante  $t$  wird definiert:

$$g(t) = f(\vec{u}_k + \nabla f(\vec{u}_k) \cdot t) \quad (\text{A.6})$$

Die Nullstelle  $g(t_0) = 0$  wird wie folgt bestimmt:

$$g(t) \approx g(0) + g'_{-t}(0) \cdot t \quad (\text{A.7})$$

$$\Rightarrow t = -g(0)/g'_{-t}(0) \quad (\text{A.8})$$

Dann wird  $\vec{u}_{k+1}$  bestimmt:

$$\vec{u}_{k+1} = \vec{u}_k - \frac{g(0)}{g'_{-t}(0)} \nabla f(\vec{u}_k) \quad (\text{A.9})$$

mit

$$g(0) = f(\vec{u}_k) \quad (\text{A.10})$$

$$g'_{-t}(t) = (\nabla f(\vec{u}_k + \nabla f(\vec{u}_k \cdot t)))^T \nabla f(\vec{u}_k)$$

$$g'_{-t}(0) = \|\nabla f(\vec{u}_k)\|^2 \quad (\text{A.11})$$

Schließlich wird folgende Gleichung hergeleitet, die zur Suche einer Nullstellen einer impliziten Funktion durch das Newton-Verfahren benutzt werden kann:

$$\vec{u}_{k+1} = \vec{u}_k - \frac{f(\vec{u}_k)}{\|\nabla f(\vec{u}_k)\|^2} \nabla f(\vec{u}_k) \quad (\text{A.12})$$

## A.4 Koordinatentransformation

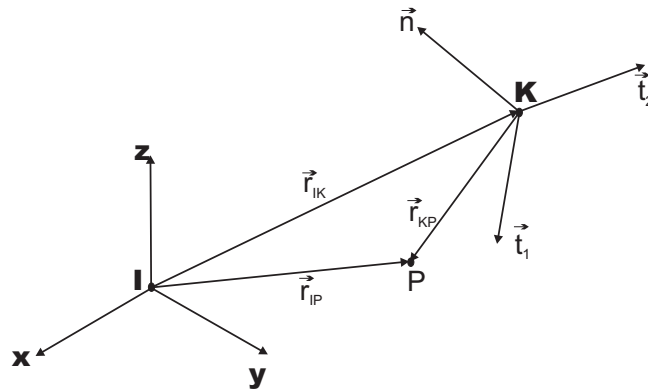


Bild A.1: Koordinatentransformation eines Punkts  $P$  zwischen einem Inertialsystem ( $I$ ) und einem lokalen Orthonormalsystem ( $K$ )

Für die kartesischen Koordinaten eines Punkts  $P$  im Inertialsystem ( $I$ ) und einem lokalen Orthonormalsystem ( $K$ ) =  $\vec{t}_1, \vec{t}_2, \vec{n}$  (Bild A.1) gilt die Transformationsformel:

$${}_K\vec{r}_{KP} = A_{KI}({}_I\vec{r}_{IP} - {}_I\vec{r}_{IK}) \quad (\text{A.13})$$



Die Transformationsmatrix  $A_{KI}$  entspricht der Inversen der Ausgangstransformationsmatrix  $A_{IK}$ . Ohne weitere Herleitung sei auf die wichtige Eigenschaft verwiesen, dass eine inverse Transformationsmatrix ihrer Transponierten entspricht:

$$A_{KI} = A_{IK}^{-1} = A_{IK}^T \quad (\text{A.14})$$

Die Ausgangsmatrix  $A_{IK}$  lässt sich durch spaltenweise Belegung mit den Einheitsvektoren der lokalen Basis gewinnen:

$$A_{IK} = ({}_I\vec{e}_{Kx} \quad {}_I\vec{e}_{Ky} \quad {}_I\vec{e}_{Kz}) = ({}_I\vec{t}_1 \quad {}_I\vec{t}_2 \quad {}_I\vec{n}) \quad (\text{A.15})$$

Damit ergibt sich die Transformationsmatrix  $A_{KI}$  zu:

$$A_{KI} = \begin{pmatrix} {}_I\vec{t}_{1x} & {}_I\vec{t}_{1y} & {}_I\vec{t}_{1z} \\ {}_I\vec{t}_{2x} & {}_I\vec{t}_{2y} & {}_I\vec{t}_{2z} \\ {}_I\vec{n}_x & {}_I\vec{n}_y & {}_I\vec{n}_z \end{pmatrix} \quad (\text{A.16})$$

## A.5 Polarwinkel

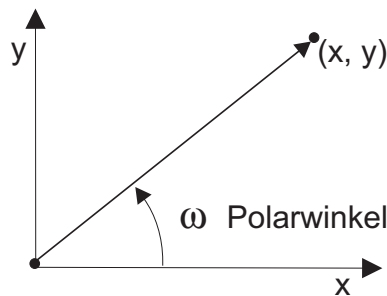


Bild A.2: Die Bedeutung des Polarwinkels

Der Polarwinkel  $\omega$  eines Punktes  $(x, y) \in \mathbb{R}^2$  (Bild A.2) kann anhand der Formel

$$\omega = \arctan\left(\frac{x}{y}\right) + \frac{\pi}{2} \text{sign}(x)(1 - \text{sign}(y)) \quad (\text{A.17})$$

berechnet werden. Für negative Werte kann man das Ergebnis durch Addieren von  $2\pi$  in den Bereich  $[0, 2\pi]$  umrechnen.

# Anhang B

## Routinen

Die im Rahmen dieser Arbeit implementierten C-Funktionen wurden auf vier Module bzw. Dateien verteilt. Es folgt eine Übersicht mit kurzen Beschreibungen.

### B.1 `pcm_crcevip.o.c`

`pcm_crcevip.o` ist die Schnittstelle zur restlichen PCM-Implementierung und ruft alle nötigen Funktionen auf.

`pcm_viposctel` erzeugt Kontaktelemente aus der zuvor generierten Triangulierung.

### B.2 `pcm_vipofun.c`

Dieses Modul enthält Funktionen zur Approximation der Kontaktfläche durch Variational Interpolation und Überblendung (siehe Kapitel 3).

`pcm_vipofundat` ist die Main-Funktion und ruft alle weiter benötigten Funktionen auf.

`pcm_vipocnsh` berechnet Positionen und Funktionswerte der Stützpunkte (Eckpunkte der aktiven Dreiecke) und Hilfspunkte (siehe Abschnitt 3.1.2) zur anschließenden Variational Interpolation.

`pcm_vipolsys` generiert und löst das lineare Gleichungssystem (2.12) der Variational Interpolation (siehe Anhang A.2).

### B.3 `pcm_vipotri.c`

Dieses Modul enthält Funktionen zur Triangulierung der zuvor approximierten Kontaktfläche.

**pcm\_vipotri** ist die Main-Funktion und ruft alle weiter benötigten Funktionen auf.

**pcm\_vipodsp** sucht einen nahe gelegenen Punkt auf einer impliziten Fläche und bestimmt die zugehörigen Tangential- und Normalvektoren (siehe Abschnitt 4.2.2).

**pcm\_vipoafp** bestimmt den Frontwinkel zu einem Punkt des aktuellen Frontpolygons (siehe Abschnitt 4.2.3.2).

**pcm\_vipoodc** führt eine Distanzprüfung durch, um ein Überlappen neuer Dreiecke zu verhindern (siehe Abschnitt 4.2.3.2).

**pcm\_vipotrie** Generiert neue Dreiecke um einen Punkt des aktuellen Frontpolygons (siehe Abschnitt 4.2.3.2).

## B.4 pcm\_vipomath.c

Dieses Modul enthält numerische Hilfsfunktionen.

**pcm\_vipotrans** transformiert die Koordinaten eines Punkts vom Intertialsystem ein lokales Orthonormalsystem (siehe Anhang A.4).

**pcm\_trans** transformiert die Koordinaten der Hilfspunkte zwischen den Referenzsystemen der Körperoberflächen.

**pcm\_viporbf** stellt die für "Variational Interpolation" benötigte radiale Basisfunktion dar.

**pcm\_gviporbf** bestimmt die Gradient der für "Variational Interpolation" benötigten radiale Basisfunktion .

**pcm\_vipofun** stellt die variational interpolierte implizite Funktion dar.

**pcm\_gvipofun** bestimmt die Gradient der variational interpolierten impliziten Funktion dar. .

**pcm\_lsyst** löst ein lineares Gleichungssystem durch LU-Zerlegung (siehe Anhang A.2).

**sign** bestimmt das Vorzeichen einer Gleitkommazahl.

**pcm\_vipolenpp** bestimmt den Abstand zweier Punkte.

**pcm\_viponormpp** bestimmt die Verbindungsrichtung zweier Punkte.

# Literaturverzeichnis

- [1] BECKER, J., DREYER, H., HAACKE, W., NABERT, R., *Numerische Mathematik für Ingenieure*. 2. Auflage, Teubner Stuttgart, 1985.
- [2] BIRAN, A., BREINER, M., *Matlab für Ingenieure. Systematische und praktische Einführung*. 2. Auflage, Addison-Wesley, 1997.
- [3] BÖTTCHER, A., KNEISSL, F., *Informatik für Ingenieure, Grundlagen und Programmierung in C*. Oldenbourg, 1999.
- [4] EBERHARD, P., *Kontaktuntersuchungen durch hybride Mehrkörpersystem / Finite Elemente Simulationen*. Habilitationsschrift, Shaker Verlag Aachen, 2000.
- [5] GLOCKER, C., *Dynamik von Starrkörpersystemen mit Reibung und Stößen*. Dissertation, Fakultät für Maschinenwesen, Technische Universität München, 1995.
- [6] GRAMLICH, G., WERNER, W., *Numerische Mathematik mit Matlab. Eine Einführung für Naturwissenschaftler und Ingenieure*. dpunkt.verlag, Heidelberg, 2000.
- [7] HARTMANN, E., *A marching method for the triangulation of surfaces*. Department of Mathematics, University of Technology, Darmstadt, 1999.
- [8] HILLS, D.A., NOWELL, D., SACKFIELD, A., *Mechanics of elastic contacts*. Butterworth-Heinemann Ltd., 1993.
- [9] HILTON, A., ILLINGWORTH, J., *Marching Triangles: Delaunay Implicit Surface Triangulation*. Vision, Speech and Signal Processing Group, Department of Electronic and Electrical Engineering, University of Surrey, Guildford, 1998.
- [10] HIPPMANN, G., *A compliant contact algorithm for multibody dynamics*. ECCOMAS Multibody Dynamics, Lissabon, 2003.
- [11] JOHNSON, K.L., *Contact mechanics*. Cambridge University Press, 1985.
- [12] KALKER, J.J., VAN RANDEN, Y., *A minimum principle for frictionless elastic contact with application to non-hertzian half-space contact problems*. In *Journal of engineering mathematics*. 6(2):193–206, 1972.
- [13] KLETA, H.G., *Polygonisierung von impliziten Feldfunktionen*. Diplomarbeit, Faculty of Media Information Science, Fachhochschule Wedel, 2002.
- [14] LORENSEN, W.E., CLINE, H.E., *Marching Cubes: A high resolution 3D surface construction algorithm*. ACM SIGGRAPH, Volume 21, Number 4, 1987.

- [15] MORSE, B.S., *Interpolating Implicit Surfaces From Scattered Surface Data Using Compactly Supported Radial Basis Functions*. Department of Computer Science, Brigham Young University, 2001.
- [16] PFEIFFER, F., GLOCKER, C., *Multibody dynamics with unilateral contacts*. Wiley&Sons New York, 1996.
- [17] QUYNH DINH, H., TURK, G., *Reconstructing Surfaces Using Anisotropic Basis Functions*. Georgia Institute of Technology, College of Computing, Graphics, Visualization, and Usability Center, 2000.
- [18] TREECE, G.M., PRAGER, R.W., GEE, A.H., *Regularised marching tetrahedra: improved iso-surface extraction*. Engineering Department, Cambridge University, 1998.
- [19] TURK, GREG, O'BRIEN, JAMES F., *Shape Transformation Using Variational Implicit Functions*. College of Computing, Georgia Institute of Technology, Atlanta, 1999.
- [20] WENDLAND, H., *Konstruktion und Untersuchung radialer Basisfunktionen mit kompaktem Träger*. Dissertation, Mathematisch-Naturwissenschaftlicher Fachbereiche der Georg-August-Universität, Göttingen, 1996.