



Hochschule für Angewandte  
Wissenschaften Hamburg  
*Hamburg University of Applied Sciences*

Prüfungsexemplar

# **MASTER-THESIS**

Entwicklung eines Discrete-Element Modells zur Simulation von  
Partikelbewegungen und -kollisionen in einer Mehrphasenströmung

Bearbeitungszeitraum: 06.01.11 - 30.06.11

Student: B. Eng. Florian Uphoff  
Matr.-Nr.: 1985214  
Studiengang: Berechnung und Simulation im Maschinenbau

Erstprüfer: Prof. Dr.-Ing. Peter Wulf  
Zweitprüfer: Prof. Dr.-Ing. habil. Frank Ihlenburg

Hochschule: Hochschule für Angewandte Wissenschaften Hamburg  
*Fakultät Technik und Informatik*  
*Department Maschinenbau und Produktion*  
Berliner Tor 21  
20099 Hamburg



## Kopie der Aufgabenstellung



Prof. Dr.-Ing. Peter Wulf

Department Maschinenbau und Produktion

Hochschule für Angewandte Wissenschaften Hamburg

Hamburg University of Applied Sciences

Januar 2011

Aufgabenstellung für eine Masterthesis (M.Eng.)

### **Entwicklung eines *Discrete-Element-Modells* zur Simulation von Partikelbewegungen und -kollisionen in einer Mehrphasenströmung**

Bei der pneumatischen Schüttgutförderung werden pulverförmige Feststoffe durch eine Luftströmung in einem Leitungssystem gefördert. Für diese weit verbreitete Technologie werden als Auslegungsziele typischerweise ein kontinuierlicher Strom bei hohen Durchsätzen ohne Instabilitäten bei gleichzeitiger Reduktion des Energieverbrauchs und schonender Förderung angestrebt. Um diese Ziele optimieren zu können, soll im Projekt SimPneuTrans ein Mehrphasen-Simulationsmodell für die pneumatische Förderung von pulverförmigen trockenen Medien zu entwickelt und angewendet werden.

Dazu soll in einem mehrstufigen Verfahren der pneumatische Transport pulverförmiger Medien zunächst mit Hilfe von Verfahren der Discrete-Element-Method (DEM) mikromechanisch modelliert werden. Aufbauend auf dem mikromechanischen Modell sollen über eine Multiskalentransformation konstitutive Gleichungen für das Fließverhalten der pulverförmigen Schüttgüter entwickelt werden, die dann in kontinuumsmechanische Euler-Euler-Modelle von Mehrphasenströmungen in der Computational-Fluid-Dynamics (CFD) eingehen. Dies wird durch rheologische Untersuchungen an Schüttgütern begleitet.



Als Aufgabenstellung für eine Masterthesis soll ein DEM-Modell für die Berechnung und Visualisierung von Partikelbewegungen und -kollisionen entwickelt werden. Dazu sollen zunächst die theoretischen Grundlagen erarbeitet und dokumentiert werden. Anschließend ist das Simulationsmodell in einer frei wählbaren Programmiersprache zu entwickeln. Um die Arbeit in einem angemessenen Umfang zu halten, werden folgende Vereinfachungen vorgegeben: Das Partikelmodell soll für 2D-Simulationen entwickelt werden und den Schwerkrafteinfluss sowie eine begrenzte Anzahl kreisförmiger Partikel ( $N < 100$ ) mit gleichem Radius berücksichtigen. Als Kollisionsmodelle sind zunächst der zentrische sowie exzentrische teilelastische Stoß zu verwenden. Das Programm soll so aufgebaut sein, dass sich weitere Kollisionsmodelle einfach integrieren lassen. Weiterhin soll eine Schnittstelle implementiert werden, um Strömungsfelder einzulesen, die vorab mit einem CFD-Programm berechnet wurden. Für die Modellierung der Kraftübertragung der Strömung auf die Partikel ist zunächst ein einfaches Widerstandsgesetz vorzusehen. Abschließend soll das Modell anhand eines Berechnungsbeispiels getestet werden (z.B. das Schweben von Partikeln im vertikalen Luftstrom).

**Einzelaufgaben:**

- Einarbeitung in die Grundlagen der Discrete-Element-Method
- Modellierung der Kinematik und Kinetik der Partikel
- Effiziente Erkennung von Partikelkollisionen
- Erstellung von Kollisionsmodellen für den zentrischen/exzentrischen teilelastischen Stoß
- Realisierung der Impuls- und Drehimpulsübertragung
- Modellierung von Randbedingungen (z.B. Einlässe, Auslässe und teilelastische Wände)
- Implementierung von expliziten Zeitdiskretisierungsverfahren verschiedener Ordnung
- Visuelle Ausgabe der Partikelbewegungen und -kollisionen (ggf. unter Nutzung eines Visualisierungsprogramms wie ParaView)
- Implementierung einer allgemein nutzbaren Schnittstelle zu einem CFD-Code
- Strukturierte schriftliche Darstellung der Vorgehensweise und der Erkenntnisse unter Beachtung der formalen Anforderungen an das wissenschaftliche Schreiben

## Vorwort

Die vorliegende Arbeit wurde im Zeitraum vom 6.01.2011 bis zum 30.06.2011 an der Hochschule für Angewandte Wissenschaften Hamburg im Rahmen des Studiengangs „Berechnung und Simulation im Maschinenbau“ erstellt. Die Themenstellung für diese Arbeit wurde sich selbst überlegt und mit Herrn Prof. Dr.-Ing. Peter Wulf so abgestimmt, dass sie für eine Master-Thesis angemessen ist. Die Idee für dieses Thema stammt aus der Mitarbeit am Projekt SimPneuTrans, das zurzeit an der Hochschule für Angewandte Wissenschaften Hamburg bearbeitet wird. Innerhalb des Projektes SimPneuTrans soll die Discrete Element Method zur Modellierung des Materialverhaltens feiner Schüttgüter angewendet werden. Zwei Ziele werden durch diese Themenstellung verfolgt. Zum einen soll durch die eigenständige Implementierung eines Discrete-Element Modells in einem Programm ein tiefer Einblick in die Theorie und Funktionsweise dieser Methode erhalten werden. Zum anderen kann das entwickelte Programm für Berechnungen innerhalb des Projektes genutzt werden und aufgrund des bekannten Programmcodes einfach an neu auftretende Problemstellungen angepasst werden.

Ich danke denjenigen, die mich bei meiner Arbeit unterstützt haben. Dies sind zum einen die Mitarbeiter der Forschungsplattform M.Sc. Nils Altfeld und M.Eng. Patrick Diffo, die immer gute Ansprechpartner waren und bei auftretenden Fragestellungen und Problemen mit gutem Rat zur Seite standen. Zum anderen möchte ich mich bei Herrn Prof. Dr.-Ing. habil. Frank Ihlenburg bedanken, dass er die Aufgabe des Zweitprüfers für diese Arbeit übernommen hat. Zuletzt möchte ich mich noch besonders bei Herrn Prof. Dr.-Ing. Peter Wulf dafür bedanken, dass er mir ermöglicht hat, diese sehr interessante Themenstellung zu bearbeiten und bei Problemen und Fragen immer ein offenes Ohr für mich hatte.

Hamburg, Juni 2011

B.Eng. Florian Uphoff



# Inhaltsverzeichnis

Kopie der Aufgabenstellung.....	I
Vorwort.....	III
Abbildungsverzeichnis.....	VII
Tabellenverzeichnis.....	IX
Symbolübersicht.....	X
<b>1 Einleitung.....</b>	<b>1</b>
1.1 Ausformulierung der Aufgabenstellung.....	1
1.2 Das Projekt SimPneuTrans.....	2
1.3 Auswahl der verwendeten Software.....	3
<b>2 Hinführung zum Thema.....</b>	<b>4</b>
2.1 Die Discrete Element Method.....	4
2.2 Kontaktmodelle.....	5
2.2.1 Soft-Particle Modell.....	5
2.2.2 Hard-Particle Modell.....	7
2.2.3 FEM-Particle Modell.....	8
2.3 Einsatzgebiete der Discrete Element Method.....	9
<b>3 Theorie der Discrete Element Method.....</b>	<b>11</b>
3.1 Herleitung der Bewegungsgleichung für die Translation.....	11
3.2 Herleitung der Bewegungsgleichung für die Rotation.....	12
3.3 Approximation der Partikelbewegung.....	14
3.4 Numerische Integrationsverfahren.....	15
3.4.1 Explizites Euler Verfahren.....	15
3.4.2 Gear Prädiktor Korrektor.....	17
3.5 Kräfte und Momente.....	18
3.5.1 Partikel-Partikel Interaktion.....	19
3.5.2 Normalenrichtung.....	20
3.5.2.1 Feder in Normalenrichtung.....	20
3.5.2.2 Dämpfer in Normalenrichtung.....	22
3.5.3 Tangentialrichtung.....	23



3.5.3.1 Feder in Tangentialrichtung .....	24
3.5.3.2 Dämpfer in Tangentialrichtung .....	24
3.5.4 Reibung .....	25
3.5.5 Drehrichtung .....	25
3.5.6 Gewichtskraft .....	26
3.5.7 Randbedingungen .....	27
3.5.8 Fluid-Partikel Interaktion .....	28
3.5.8.1 Bilineare Interpolation .....	29
3.5.8.2 Shepards Method .....	30
3.5.9 Gravitationskräfte .....	31
3.6 Zusammenhang von Stoßzahl und Lehrscher Dämpfung .....	31
<b>4 Realisierung des Programms .....</b>	<b>34</b>
4.1 Hauptprogramm .....	34
4.1.1 Eingabe .....	34
4.1.2 Definition von Konstanten und Berechnung von Parametern .....	35
4.1.3 Berechnung .....	37
4.1.4 Ausgabe von Diagrammen .....	40
4.2 Unterprogramm Write_Startvalues.m .....	41
4.3 Unterprogramm Read_Startvalues.m .....	43
4.4 Unterprogramm Write_Boundary.m .....	43
4.5 Unterprogramm Read_Boundary.m .....	46
4.6 Unterprogramm Write_Conditions.m .....	46
4.7 Unterprogramm Read_Conditions.m .....	48
4.8 Unterprogramm Import_CFD .....	49
4.9 Funktion Calculate_Forces.m .....	50
4.9.1 Partikel-Partikel Interaktion .....	50
4.9.2 Randbedingung-Partikel Interaktion .....	56
4.9.3 Fluid-Partikel Interaktion .....	58
4.9.4 Gravitation .....	61
4.9.5 Aufsummierung der Kräfte .....	61
4.10 Funktion kreuz.m .....	62
4.11 Unterprogramm Delete_Data.m .....	62
4.12 Unterprogramm Convert_ml2vtk.m .....	63



---

<b>5 Verifizierung des Modells .....</b>	<b>64</b>
5.1 Beispielrechnung 1.....	64
5.2 Beispielrechnung 2.....	67
<b>6 Zusammenfassung.....</b>	<b>70</b>
6.1 Fazit .....	70
6.2 Ausblick.....	71
Literaturverzeichnis .....	72
Erklärung über eigenständige Erstellung der Arbeit .....	75
<b>Anhang .....</b>	<b>A</b>

## Abbildungsverzeichnis

Abb. 2.1: Kinetische Definition eines Partikels.....	5
Abb. 2.2: Feder-Dämpfer-Modell nach Cundall und Strack .....	6
Abb. 2.3: Feder-Dämpfer Modell für die Drehrichtung .....	6
Abb. 2.4: Zerbrechen eines Felsens [Int2].....	9
Abb. 2.5: Beschuss einer Betonscheibe [Int3] .....	9
Abb. 2.6: Trommeltrockner [Int4] .....	10
Abb. 2.7: Auslaufen eines Silos [Int5].....	10
Abb. 2.8: Verdichten eines Schüttgutes [Int6].....	10
Abb. 3.1: Translations- und Rotationsfreiheitsgrade [Int1].....	11
Abb. 3.2: Modifiziertes explizites Euler-Verfahren .....	16
Abb. 3.3: Korrektor-Schritt des Praedikator-Korrektor-Verfahrens.....	17
Abb. 3.4: Kontakterkennung zwischen Partikeln.....	19
Abb. 3.5: Feder-Dämpfer Modell in Normalenrichtung .....	20
Abb. 3.6: Kraft-Weg Diagramm des Kugel-Kugel Kontaktes.....	21
Abb. 3.7: Feder-Dämpfer Modell in Tangentialrichtung .....	23
Abb. 3.8: Feder-Dämpfer Modell mit Reibkopplung.....	25
Abb. 3.9: Feder-Dämpfer Modell in Drehrichtung .....	26
Abb. 3.10: Feder-Dämpfer Modell zwischen Partikel und Wand.....	27
Abb. 3.11: Feder-Dämpfer Modell für die Drehfeder zwischen Partikel und Wand...	28
Abb. 3.12: Bilineare Interpolation.....	30
Abb. 3.13: Zusammenhang von Restitutionskoeffizient $e$ und Dämpfungsfaktor $\alpha$ ...	32
Abb. 3.14: Restitutionskoeffizient über dem doppelten Dämpfungsmaß .....	33
Abb. 4.1: Hauptmenü.....	34
Abb. 4.2: Ausgabe von Informationen .....	35
Abb. 4.3: Ausgabe der Berechnungsergebnisse .....	41
Abb. 4.4: Berechnungsgebiet mit Randbedingungen .....	46
Abb. 4.5: Zentrischer Kontakt zweier Kugeln.....	53
Abb. 4.6: Zentrischer Kontakt zweier Kugeln mit Dämpfung.....	54
Abb. 4.7: Testrechnungen für die tangentiale Feder.....	54
Abb. 4.8: Berechnung eines Kontaktes mit unterschiedlichen Zeitschrittweiten .....	55





---

Abb. 4.9: Testrechnungen für die tangentielle Dämpfung .....	56
Abb. 4.10: Einzugsgebiet der Wand .....	56
Abb. 4.11: Dreieck aus Partikelmittelpunkt und der Kante AB .....	57
Abb. 4.12: Berechnung des Hilfsvektors.....	58
Abb. 4.13: Abprallen einer Stahlkugel vom Boden .....	58
Abb. 4.14: Testrechnung, Interpolation mit der Shepards Method.....	59
Abb. 4.15: Testrechnung, Interpolation mit interp2 .....	60
Abb. 5.1: Kugel-Kugel Kontakt.....	64
Abb. 5.2: Beschleunigungsverlauf .....	65
Abb. 5.3: Ermittlung der Kontaktzeit .....	65
Abb. 5.4: Beschleunigungsverlauf .....	66
Abb. 5.5: Ermittlung der Kontaktzeit .....	66
Abb. 5.6: Strömungsfeld .....	68
Abb. 5.7: Partikel schwebt noch nicht.....	69
Abb. 5.8: Schweben des Partikels .....	69



## Tabellenverzeichnis

Tab. 3.1: Unterschiedliche Arten von Kräften .....	18
Tab. 4.1: csv-Dateien .....	37
Tab. 4.2: Daten der statvalue.txt Datei .....	42
Tab. 4.3: Randbedingungen .....	43
Tab. 4.4: Daten der boundary.txt Datei.....	44
Tab. 4.5: Daten der conditions.txt Datei.....	48
Tab. 4.6: Aufbau der vtk-Datei.....	49
Tab. 5.1: Parameter für die Berechnung .....	65
Tab. 5.2: Eigenschaften.....	68



## Symbolübersicht

### Griechische Buchstaben

Symbol	Bezeichnung	Einheit
$\alpha$	Winkelbeschleunigung	$\text{rad}\cdot\text{s}^{-2}$
	Dämpfungsfaktor	-
	Winkel im Dreieck	$^{\circ}$
$\beta$	Winkel im Dreieck	$^{\circ}$
$\delta$	Überlappung	m
$\delta_0$	Abklingkonstante	$\text{s}^{-1}$
$\varepsilon$	Abbruchkriterium Praediktor-Korrektor Iteration	-
$\theta$	Trägheitstensor	$\text{kg}\cdot\text{m}^2$
$\mu$	Reibungskoeffizient	-
$\nu$	Poisson-Zahl	-
$\rho$	Dichte	$\text{kg}\cdot\text{m}^{-3}$
$\varphi$	Rotationswinkel	rad
$\omega$	Winkelgeschwindigkeit	$\text{rad}\cdot\text{s}^{-1}$
$\omega_0$	Eigenfrequenz	$\text{s}^{-1}$
$\Delta$	Symbol für Differenzen	-

### Lateinische Buchstaben

Symbol	Bezeichnung	Einheit
a	Beschleunigung	$\text{m}\cdot\text{s}^{-2}$
c	Dämpfungskonstante (Translation)	$\text{N}\cdot\text{s}\cdot\text{m}^{-1}$
	Dämpfungskonstante (Rotation)	$\text{N}\cdot\text{m}\cdot\text{s}\cdot\text{rad}^{-1}$
	Unterrelaxationsfaktor	-
	Index (hochgestellt) für Korrektorterm	-
$c_w$	Widerstandsbeiwert	-
d	Abstand	m
e	Stoßzahl, Restitutionskoeffizient	-
	Einservektor	-
f	Index für Fluid	-
g	Erdbeschleunigung	$\text{m}\cdot\text{s}^{-2}$
h	Abstand	
k	Federsteifigkeit (Translation)	$\text{N}\cdot\text{m}^{-1}$
	Drehfedersteifigkeit (Rotation)	$\text{N}\cdot\text{m}\cdot\text{rad}^{-1}$
m	Masse	kg
n	Anzahl der Partikel	-
	Normalenvektor	-
	Index für die Normalenrichtung	-
p	Impuls	$\text{N}\cdot\text{s}$
	Index für Partikel	-
	Index (hochgestellt) für Prädiktorterm	-



r	Position	m
	Index für die Rotation	-
t	Zeit	s
	Tangentialvektor	-
	Index für die Tangentialrichtung	-
v	Geschwindigkeit	$\text{m}\cdot\text{s}^{-1}$
w	Gewichtung	-
x	Index für 1. Koordinatenrichtung (x-Achse)	-
y	Index für 2. Koordinatenrichtung (y-Achse)	-
z	Index für 3. Koordinatenrichtung (z-Achse)	-
A	Projektionsfläche des Partikels	$\text{m}^2$
D	Lehrsches Dämpfungsmaß	-
E	Elastizitätsmodul	$\text{N}\cdot\text{m}^{-2}$
F	Kraft	N
G	Gravitationskonstante	$\text{m}^3\cdot\text{kg}^{-1}\cdot\text{s}^{-2}$
J	Massenträgheitsmoment	$\text{kg}\cdot\text{m}^2$
L	Drehimpuls	$\text{N}\cdot\text{m}\cdot\text{s}$
M	Moment	$\text{N}\cdot\text{m}$
R	Partikelradius	m
V	Partikelvolumen	$\text{m}^3$

## Formelzeichen

Symbol	Bezeichnung
*	Der folgende Operator wird elementweise angewendet
$\times$	Kreuzprodukt
$\otimes$	Dyadisches Produkt
*	Elementweise Multiplikation von Vektoren und Matrizen
<sup>T</sup>	Transponiert
<sup>-1</sup>	Inverse

## Allgemeines:

Tensoren erster Stufe bzw. Vektoren sind mit einem einfachen Pfeil über der Variablen dargestellt. Tensoren 2. Stufe, bzw. Matrizen sind mit einem Doppelpfeil über der Variablen dargestellt.



# 1 Einleitung

## 1.1 Ausformulierung der Aufgabenstellung

Das Thema dieser Master-Thesis lautet „Entwicklung eines Discrete-Element Modells zur Simulation von Partikelbewegungen und -kollisionen in einer Mehrphasenströmung“. Diese Arbeit findet im Rahmen des Projektes SimPneuTrans statt und soll dazu dienen einerseits einen tieferen Einblick in das Berechnungsverfahren der Discrete Element Method zu erlangen und andererseits ein Programm für erste Versuchsrechnungen zu erhalten. Das Projekt SimPneuTrans wird in Kapitel 1.2 kurz vorgestellt.

Um in die Thematik dieser Arbeit zu gelangen, wird in Kapitel 2 die Discrete Element Method ausführlich beschrieben und ein kurzer Einblick in ihre Anwendungsgebiete gegeben. Außerdem werden die unterschiedlichen Kontaktmodelle, die innerhalb dieses Verfahrens genutzt werden können, erläutert.

Im dritten Kapitel wird die Theorie der Discrete Element Method anhand der dazugehörigen Gleichungen hergeleitet. Besonderer Wert wird dabei auf die Bewegungsgleichungen und die numerischen Iterationsverfahren gelegt, die für die spätere Implementierung des Programms benötigt werden. Außerdem wird das verwendete Kontaktmodell, das Soft-Particle Modell, betrachtet und die Kräfte und Momente, die auf die Partikel wirken, werden ausführlich beschrieben.

Das vierte Kapitel behandelt den Hauptteil dieser Arbeit, die praktische Implementierung der Discrete Element Method mit Kopplung einer Strömungssimulation als Programm. Hierbei werden das Hauptprogramm, sowie alle Unterprogramme und Funktionen beschrieben und deren Entwicklung erklärt. Zudem werden Validierungen durchgeführt, die sicherstellen, dass das Programm stabil rechnet und nachvollziehbare Ergebnisse liefert.

Das fünfte Kapitel ist der praktischen Anwendung des Programms gewidmet. Hierzu wird die Berechnung zweier Testfälle durchgeführt. Im ersten Fall wird die



analytische Lösung für einen Kugel-Kugel Kontakt von Heinrich Hertz aus dem Jahre 1881 nachgerechnet. Der zweite Fall behandelt das in der Aufgabenstellung vorgeschlagene Schweben einer Kugel im Luftstrom.

Im sechsten Kapitel werden die Erkenntnisse, die im Laufe dieser Ausarbeitung gewonnen wurden, zusammengefasst und es wird ein Ausblick auf mögliche Erweiterungen und Verbesserungen des entwickelten Programms gegeben.

## 1.2 Das Projekt SimPneuTrans

Das Projekt SimPneuTrans beschäftigt sich mit der Entwicklung eines Modells für die numerische **Simulation** und Optimierung des **pneumatischen Transportes** von pulverförmigen Medien und wird durch das Bundesministerium für Bildung und Forschung gefördert. Industriepartner des Projektes ist die FLSmidth Hamburg GmbH mit Sitz in Pinneberg. Die FLSmidth Hamburg GmbH ist ein Unternehmen aus dem Bereich des Schüttguttransportes, das sowohl Anlagenbauteile entwickelt, als auch gesamte Anlagen zum Transport von feinen Pulvern konzipiert. Diese Pulver, wobei es sich hauptsächlich um Flugasche aus Kohlekraftwerken handelt, werden mittels der Dichtstromförderung transportiert.

Die Dichtstromförderung ist ein Verfahren, bei dem feine Schüttgüter mit Hilfe von Druckluft bei sehr hohen Beladungsgraden durch Rohrleitungen gepumpt werden. Das Massenverhältnis von Schüttgut zu Transportluft kann hierbei bis zu 20 zu 1 liegen [Int7]. Dieses Verfahren bietet zwei große Vorteile gegenüber der sonst noch häufig verwendeten Flugförderung. Zum einen ist, wie bereits erwähnt, das Verhältnis von Schüttgutmenge zu Luftmenge sehr hoch, dadurch wird weniger Druckluft und somit auch weniger Energie zur Bereitstellung dieser benötigt. Zum anderen ist die Transportgeschwindigkeit mit 2 bis 3 m/s [Lit21] wesentlich niedriger, das die Kornzerstörung des Schüttgutes und den Verschleiß der Anlagen reduziert. Ein Effekt der bei der Dichtstromförderung auftritt, ist die Fluidisierung des pulverförmigen Mediums. Dabei setzt sich die Luft zwischen die einzelnen Pulverteilchen und verringert dadurch deren Reibung, wodurch sich das fluidisierte Pulver wie eine Art Flüssigkeit verhält.



Ziel des Projektes SimPneuTrans ist es ein Modell zu entwickeln, das den Prozess der Dichtstromförderung mit Hilfe einer mehrphasigen CFD-Simulation (Computational Fluid Dynamics) nach dem Euler-Euler Modell<sup>1</sup> abbildet. Mit Hilfe dieses Modells sollen zuverlässige Scale-Up Methoden vom Labor- zum Industriemaßstab entwickelt werden, so dass die Kosten der Anlagenentwicklung, als auch die Anzahl der experimentellen Versuche reduziert werden können.

Ein wichtiger Schritt bei der Erstellung dieses Modells sind die genauen Kenntnisse des Zusammenhangs der Material- und Transporteigenschaften des Schüttgutes. Ein mathematisches Modell hierfür bietet die Discrete Element Method (DEM). Mit der DEM lässt sich ein mikromechanisches Modell des Schüttgutes erstellen, das anschließend durch die Anpassung an Ergebnisse aus Experimenten kalibriert werden kann. Durch eine anschließende Multiskalentransformation sollen dann aus dem DEM-Modell konstitutive Gleichungen für das Fließverhalten des Schüttgutes ermittelt werden. Diese gehen dann in die Entwicklung des Mehrphasenmodells ein. [vgl. Lit21]

### 1.3 Auswahl der verwendeten Software

Nach Erhalt der Aufgabenstellung wurde zunächst eine geeignete Software für die Implementierung des Programms ausgewählt. Dabei wurde sich für die Software Matlab von der Firma The MathWorks entschieden. Die Auswahl begründete sich zum einen aus dem Vorhandensein von Vorkenntnissen, die während des Master-Studiums erworben wurden und zum anderen aufgrund der vielen internen Funktionen die Matlab bereit stellt. Als CFD-Programm für die Strömungsberechnung wurde sich für das Softwarepaket OpenFOAM entschieden. Dieses ist ein allgemeiner Differentialgleichungslöser, der unter der General Public Licence frei verfügbar ist. Zur Visualisierung der Berechnungsergebnisse wurde sich für das Visualisierungstool ParaView entschieden, eine ebenfalls frei verfügbare Software. Beide Programme befinden sich als Windows-Version auf der CD-ROM im Anhang.

---

<sup>1</sup> Beim Euler-Euler Modell werden alle Phasen als kontinuierliche Phasen modelliert, Gegenteil ist das Euler-Lagrange Modell mit kontinuierlichen und diskreten Phasen.



## 2 Hinführung zum Thema

### 2.1 Die Discrete Element Method

Die Discrete-Element Method ist ein numerisches Verfahren zur Berechnung des Bewegungs- und Kollisionsverhaltens einer großen Anzahl von Partikeln. Sie wurde erstmals im Jahre 1979 von den beiden Geotechnologen Peter Cundall und Otto Strack in ihrer Ausarbeitung „*A discrete numerical model for granular assemblies*“ veröffentlicht.

Bei der Discrete Element Method werden die Partikel als diskrete Elemente modelliert. Dies bedeutet, dass die einzelnen Elemente abzählbar und im Gegensatz zu kontinuierlichen Verfahren nicht geometrisch aneinander gebunden sind. Bei der DEM können sich die einzelnen Partikel frei im Raum bewegen und werden nur durch die auf sie wirkenden Kräfte und Momente bzw. auf sie übertragenen Impulse und Drehimpulse in ihrem Bewegungs- und Rotationsverhalten beeinträchtigt. Ob eine Kraft auf ein Partikel wirkt, oder ein Impuls auf ein Partikel übertragen wird, bewirkt im Endeffekt dieselbe Änderung der Bewegung und hängt nur von dem gewählten Kontaktmodell ab. Es werden heutzutage überwiegend drei unterschiedliche Kontaktmodelle verwendet, das Soft-Particle Modell, das Hard-Particle Modell und das FEM-Particle Modell. Da die Modellierung des Kontaktes bei der DEM im Vordergrund steht, werden die drei Kontaktmodelle in Kapitel 2.3 detailliert beschrieben.

Die Partikel können bei der DEM unterschiedliche Geometrie aufweisen. Diese reicht von der einfachen Kreis- bzw. Kugelform bis hin zu komplexen Geometrien wie Polyedern. Letztere sind Stand aktueller Forschung, da für nicht kreis- bzw. kugelförmige Körper die Kollisionserkennung, sowie die dabei zwischen den Körpern auftretenden Kräfte und Momente nur sehr kompliziert zu ermitteln sind. Ein einfacher, heute schon erfolgreich verwendeter Ansatz zur Berechnung von Partikeln mit komplexerer Form ist die Approximation der Geometrie mittels Kugelkonglomeraten [Lit14]. Hierbei werden mehrere Kugeln mit unterschiedlichen Radien so aneinandergelegt, dass sie die Geometrie möglichst genau annähern.



## 2.2 Kontaktmodelle

Da je nach Kontaktmodell die DEM Berechnung sehr unterschiedlich abläuft, werden die drei am häufigsten verwendeten Kontaktmodelle anhand ihres Berechnungsablaufs und ihrer jeweiligen Vor- und Nachteile beschrieben.

### 2.2.1 Soft-Particle Modell

Das Soft-Particle Modell ist das am häufigsten verwendete Kontaktmodell [Lit20]. Es ist das Modell, das auch von Cundall und Strack in ihrer Veröffentlichung beschrieben wird [Lit15]. Bei diesem Berechnungsmodell werden für jedes Partikel seine aktuelle Position  $\vec{r}$ , Geschwindigkeit  $\vec{v}$  und Beschleunigung  $\vec{a}$ , sowie Rotationswinkel  $\vec{\varphi}$ , Winkelgeschwindigkeit  $\vec{\omega}$  und Winkelbeschleunigung  $\vec{\alpha}$  ausgerechnet. Die Beschleunigung und Winkelbeschleunigung werden direkt aus den auf das Partikel wirkenden Kräften  $\vec{F}$  und Momenten  $\vec{M}$  ermittelt. Die Position und die Geschwindigkeit, sowie der Rotationswinkel und die Winkelgeschwindigkeit werden als Startwerte gegeben und für die nächsten Zeitschritte über Approximationen aus der jeweiligen Beschleunigung ermittelt. Eine Detaillierung folgt in Kapitel 3. Vorteil dieses Modells ist, dass sämtliche Kräfte und Momente, die auf und zwischen Partikeln wirken, in dieses integriert werden können.

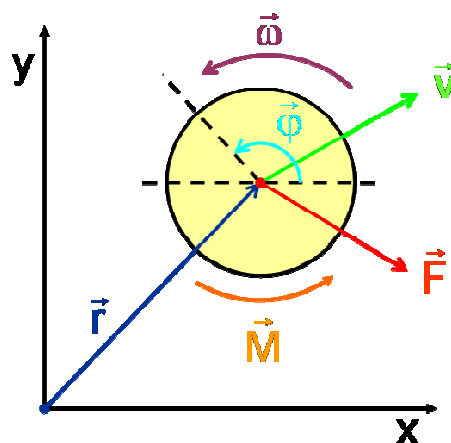


Abb. 2.1: Kinetische Definition eines Partikels

Die Kontaktkräfte zwischen zwei Partikeln werden beim Soft-Particle Modell über ein Feder-Dämpfer-System zwischen den Partikeln realisiert. Hierbei wird am häufigsten das von Cundall und Strack entwickelte Feder-Dämpfer System [Lit15] verwendet.

Dieses besteht aus jeweils einem Feder-Dämpfer-System in Normalen- und Tangentialrichtung, die über eine Reibkupplung miteinander verbunden sind (s. Abb. 2.2). Somit verhalten sich zwei oder mehrere kollidierende Partikel wie ein Zwei- bzw. Mehrmassen-Schwinger. Die Federn simulieren in diesem Fall die elastische Verformung der Kugeln. Die Dämpfer simulieren die Energiedissipation, die aufgrund von plastischer Verformung und Entstehung von Wärme auftritt.

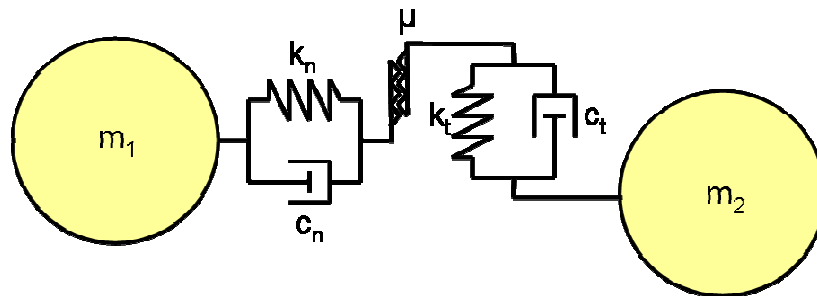


Abb. 2.2: Feder-Dämpfer-Modell nach Cundall und Strack

Laut [Lit15] muss für die Realisierung des aufeinander Abrollens der Partikel ein zusätzliches Feder-Dämpfer-System in Drehrichtung für die Übertragung von Momenten zwischen den kollidierenden Partikeln integriert werden (s. Abb. 2.3).

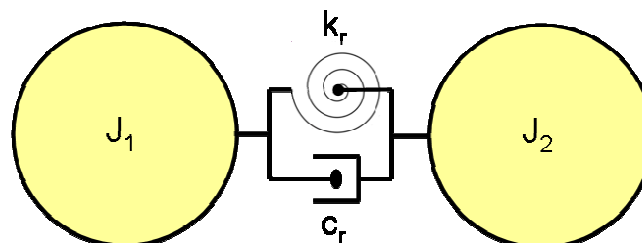


Abb. 2.3: Feder-Dämpfer Modell für die Drehrichtung

Der Name Soft-Particle Modell stammt daher, dass die Partikel als deformierbar angenommen werden. Das heißt, dass die Partikel Kräfte aufgrund elastischer Verformung aufnehmen und wieder abgeben können. Allerdings wird die Verformung über das Feder-Dämpfer-System abgebildet, so dass sich die Geometrie des Partikels während des Kontaktes nicht ändert.



Vorteile:

- Mehrfachkontakte lassen sich problemlos berechnen.
- Sehr flexibel, es lassen sich alle möglichen auf das Partikel wirkenden Kräfte und Momente integrieren.
- Kontaktberechnung von Partikeln mit unterschiedlichen Geometrie- und Materialeigenschaften möglich.
- Berechnung der Kontaktzeit ist möglich.

Nachteile:

- Aufgrund der zeitlich sehr feinen Auflösung des Kontaktes, sind lange Rechenzeiten notwendig.
- Wenn die Zeitschrittweite zu groß gewählt wird, treten enorme Fehler auf.

### 2.2.2 Hard-Particle Modell

Beim Hard-Particle Modell werden die Partikel als unverformbare, harte Kugeln angenommen. Das Hard-Particle Modell entspricht im Wesentlichen dem, was in der Technischen Mechanik als Stoß bezeichnet wird. Dabei wird der Impuls bei Kontakt direkt von einem auf das andere Partikel übertragen. Eine feine zeitliche Auflösung des Kontaktes ist deshalb nicht notwendig. Nur die Kollision selbst muss mit dem gewählten Zeitschritt detektiert werden. Die Dissipation beim Kontakt lässt sich über den Restitutionskoeffizienten  $e$  sehr einfach und genau einstellen.

Im eindimensionalen Fall lassen sich die Geschwindigkeiten der Partikel nach einem teilelastischen Stoß nach den Formeln 2.1 und 2.2 berechnen:

$$v_1' = \frac{m_1 v_1 + m_2 v_2 - m_2 (v_1 - v_2) e}{m_1 + m_2} \quad (2.1)$$

$$v_2' = \frac{m_1 v_1 + m_2 v_2 - m_1 (v_2 - v_1) e}{m_1 + m_2} \quad (2.2)$$

Ein gravierender Nachteil dieser Methode ist, dass innerhalb eines Zeitschrittes nur der Kontakt zwischen jeweils zwei Partikeln berechnet werden kann. Die Berechnung eines Mehrfachkontaktes, der gerade bei großen Partikelanzahlen andauernd auftritt, kann mit diesem Verfahren nicht, bzw. nur mit großen Fehlern berechnet werden.



Außerdem ist die Modellierung der tangentialen Berührung zweier Partikel und das aufeinander Abrollen der Partikel sehr aufwendig.

Vorteile:

- Sehr schnell, für eine Kontaktberechnung ist nur ein Zeitschritt notwendig.
- Dissipation lässt sich einfach über den Restitutionskoeffizienten integrieren.

Nachteile:

- Es können keine Mehrfachkontakte berechnet werden.
- Modellierung von tangentialer Berührung und Abrollen sehr aufwendig.

### **2.2.3 FEM-Particle Modell**

Das FEM-Particle Modell ist das neueste Kontaktmodell. Hierbei werden die Partikel selbst als Finite Elemente Körper berechnet. Es ist das genaueste Verfahren, da hierbei die Verformungen des Partikelkörpers und die daraus resultierenden Steifigkeitsänderungen sehr genau wiedergegeben werden. Außerdem können auf diese Weise auch Körper berechnet werden, die keine Kugelform haben. Der große Nachteil dieser Methode ist allerdings, dass aufgrund der Auflösung der Kugelgeometrie mittels FEM enorm viel Rechenleistung benötigt wird und sich Rechenzeiten enorm verlängern. Schließlich muss zu jedem Zeitschritt eine komplette FE-Berechnung aller Kugeln durchgeführt werden. Aufgrund dessen wird dieses Verfahren nur dann eingesetzt, wenn nur wenige Partikel miteinander interagieren. So wird es z.B. für die Verifizierung von anderen Kontaktmodellen verwendet.

Vorteile:

- Sehr genaue Berechnung des Kontaktes ist möglich.
- Berechnung von sehr komplexen Geometrie- und Materialeigenschaften der Partikel ist möglich.
- Ansonsten hat es dieselben Vorteile wie das Soft-Particle Modell

Nachteil:

- Es werden sehr hohe Rechenkapazitäten und -zeiten benötigt.

### 2.3 Einsatzgebiete der Discrete Element Method

Die Discrete Element Method wird heutzutage in vielen unterschiedlichen Bereichen eingesetzt. Ein Bereich ist immer noch der, für den die DEM eigentlich entwickelt wurde, die Berechnung des Bruchverhaltens von spröden Materialien. Hierzu zählen hauptsächlich Materialien wie Felsen und andere Gesteine. Aber auch Materialien aus denen Werkstücke gefertigt werden, wie Gusseisen, Glas oder Keramiken werden heutzutage mit der DEM auf ihr Bruchverhalten untersucht.

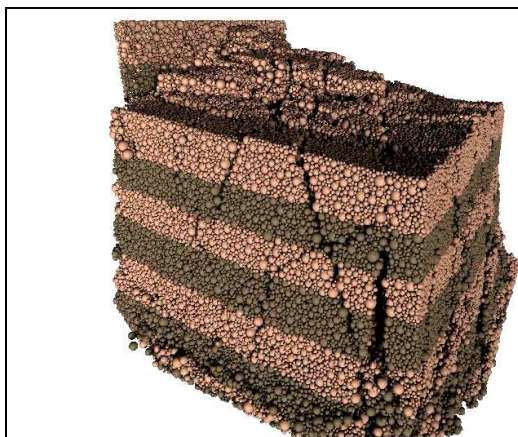


Abb. 2.4: Zerbrechen eines Felsens [Int2]

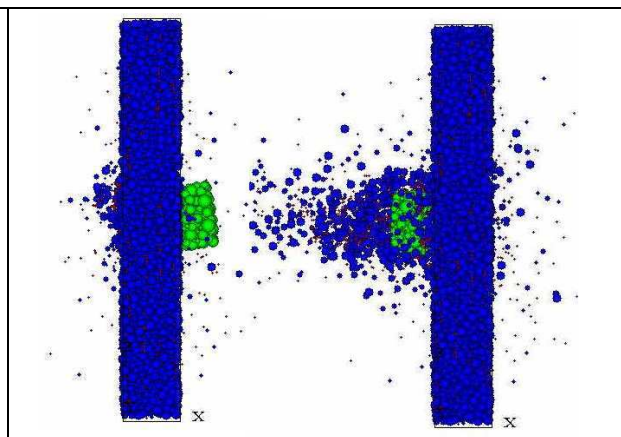
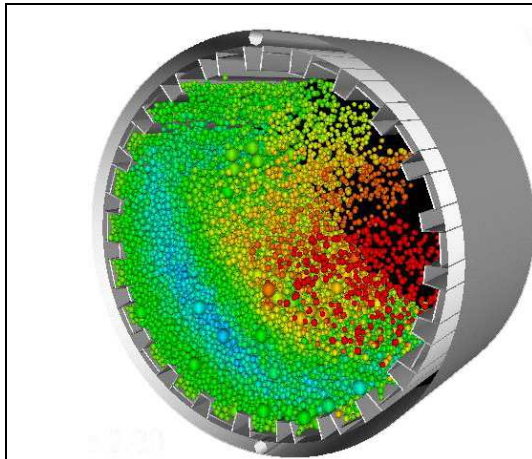
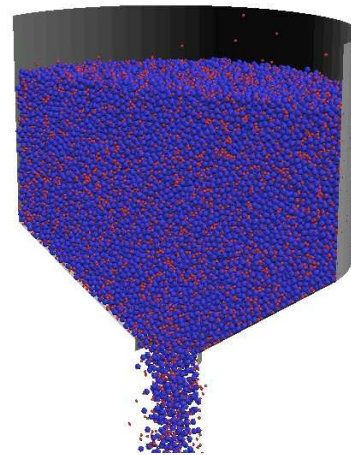


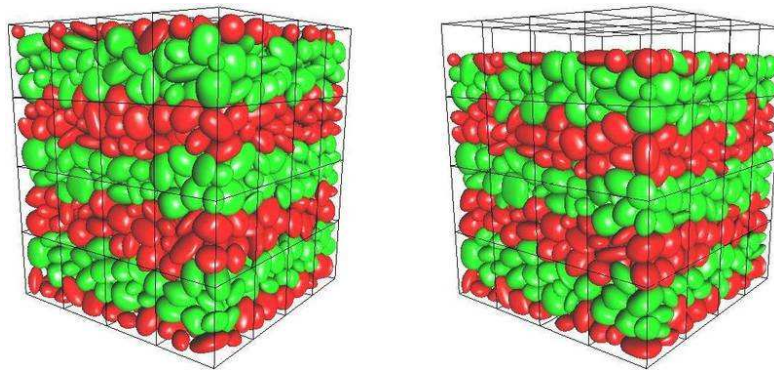
Abb. 2.5: Beschuss einer Betonscheibe [Int3]

In Abbildung 2.4 ist das Bruchverhalten eines Felsens dargestellt und in Abbildung 2.5 wurde das Beschießen einer Betonscheibe mit einem Geschoss simuliert.

Ein weiterer Bereich in der die DEM eingesetzt wird, ist die Simulation von Schüttgütern. Hierbei können unterschiedlichste Verarbeitungs- und Transportprozesse untersucht werden. Bei den Schüttgütern handelt es sich um ein weites Spektrum von Materialien. Auch die Größenordnung der Partikel ist je nach Industriezweig sehr unterschiedlich. So werden zum einen hochfeine Pulver im Nanometer-Bereich für die Herstellung von Materialien mit Nano-Eigenschaften (z.B. Sonnencreme mit Nanopartikeln aus Titandioxid) und zum anderen der Abtransport von Felsbrocken aus Steinbrüchen im Meter-Bereich simuliert. In Abbildung 2.6 wird zum Beispiel das Trocknungsverhalten von Kunststoffgranulaten in einem Trommeltrockner simuliert.

**Abb. 2.6: Trommeltrockner [Int4]****Abb. 2.7: Auslaufen eines Silos [Int5]**

Ein häufig simulierter Prozess ist das Auslaufen eines Schüttgutes aus einem Vorratssilo (s. Abb. 2.7). Hierbei wird vor allem das Bilden von Brücken und Gewölben durch Verkanten der Partikel untersucht, die das Auslaufen des Silos negativ beeinflussen. Außerdem werden Partikelverdichtungsprozesse, wie z.B. beim Sintern (s. Abb. 2.8), sowie Misch- und Agglomerationsprozesse mit Hilfe der DEM untersucht.

**Abb. 2.8: Verdichten eines Schüttgutes [Int6]**

### 3 Theorie der Discrete Element Method

In diesem Kapitel soll auf die theoretischen Grundlagen der Discrete Element Method detailliert eingegangen werden. Dazu werden zunächst die Bewegungsgleichungen der Partikel hergeleitet und anschließend die numerischen Iterationsverfahren vorgestellt, die die Grundlage des zu implementierenden Programms bilden. Danach werden die Kräfte und Momente, die auf die Partikel wirken erläutert.

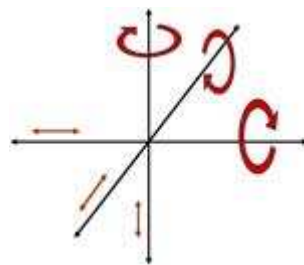


Abb. 3.1: Translations- und Rotationsfreiheitsgrade [Int1]

Um die Bewegung der Partikel zu beschreiben, werden Differentialgleichungen benötigt, die die physikalische Bewegung der Partikel mathematisch beschreiben. Im allgemeinen Fall hat jedes Partikel im dreidimensionalen Raum sechs voneinander unabhängige Freiheitsgrade (s. Abb. 3.1), davon sind drei die Translation in die drei Raumrichtungen und die anderen drei die Rotation um die drei Koordinatenachsen.

#### 3.1 Herleitung der Bewegungsgleichung für die Translation

Grundlage für die Bewegungsgleichung der Translation eines Partikels ist das 2. Newtonsche Gesetz, wonach die Summe aller äußeren Kräfte auf ein Partikel gleich seiner zeitlichen Impulsänderung ist [vgl. Lit9].

$$\sum \vec{F} = \frac{d\vec{p}}{dt} \quad (3.1)$$

Der Impuls  $\vec{p}$  ist dabei als das Produkt aus Masse  $m$  und Geschwindigkeit  $\vec{v}$  eines Partikels definiert, so dass gilt:

$$\vec{p} = m\vec{v} \quad (3.2)$$



Da im Soft-Particle Modell die Partikel sowohl ihre Form als auch ihre Dichte behalten, ist somit auch zwangsläufig die Masse der Partikel konstant. Somit folgt aus Gleichung 3.2, dass die Kraft proportional zur zeitlichen Änderung der Partikelgeschwindigkeit, also proportional zur Partikelbeschleunigung ist. Der Proportionalitätsfaktor ist die Partikelmasse.

$$\sum \vec{F} = \frac{d\vec{p}}{dt} = m \frac{d\vec{v}}{dt} = m\vec{a} \quad (3.3)$$

Wird diese Gleichung für n Partikel im dreidimensionalen Raum verallgemeinert, dann folgt daraus, dass die Beschleunigungsmatrix gleich dem Produkt aus inverser Massenmatrix und der Kraftmatrix ist.

$$\vec{a} = \vec{m}^{-1} \sum \vec{F} \quad (3.4)$$

$$\text{mit } \begin{bmatrix} a_{1x} & a_{1y} & a_{1z} \\ a_{2x} & a_{2y} & a_{2z} \\ \vdots & \vdots & \vdots \\ a_{nx} & a_{ny} & a_{nz} \end{bmatrix} = \begin{bmatrix} m_1 & 0 & \dots & 0 \\ 0 & m_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & m_n \end{bmatrix}^{-1} \sum \begin{bmatrix} F_{1x} & F_{1y} & F_{1z} \\ F_{2x} & F_{2y} & F_{2z} \\ \vdots & \vdots & \vdots \\ F_{nx} & F_{ny} & F_{nz} \end{bmatrix} \quad (3.5)$$

Der erste Index zeigt dabei die Nummer des Partikels an und der zweite Index die Raumrichtung. Auf diese Weise lassen sich aus den äußeren Kräften zu jedem Zeitpunkt die Beschleunigungen auf das Partikel berechnen.

### 3.2 Herleitung der Bewegungsgleichung für die Rotation

Grundlage für die Bewegungsgleichung der Rotation ist der Eulersche Drehimpulssatz, welcher analog zum 2. Newtonschen Gesetz ist. Hierbei ist die Summe der äußeren Drehmomente auf ein Partikel gleich seiner zeitlichen Drehimpulsänderung.

$$\sum \vec{M} = \frac{d\vec{L}}{dt} \quad (3.6)$$

Der Drehimpuls  $\vec{L}$  ist dabei als das Produkt aus Trägheitstensor  $\vec{\theta}$  und Winkelgeschwindigkeit  $\vec{\omega}$  eines Partikels definiert, so dass gilt:

$$\vec{L} = \vec{\theta}\vec{\omega} \quad (3.7)$$





Handelt es sich bei dem geometrischen Körper um einen rotationsymmetrischen Körper, der um seinen Schwerpunkt rotiert, so hat der Trägheitstensor die Form einer Diagonalmatrix, mit den Hauptträgheitsmomenten um die drei Koordinatenachsen als Elementen. Bei einer Kugel sind diese alle gleich, so dass das Trägheitsmoment als Faktor vor die Einheitsmatrix geschrieben werden kann.

$$\bar{\bar{\theta}} = \begin{matrix} \text{Allgemein} \\ \begin{bmatrix} J_x & J_{xy} & J_{xz} \\ J_{yx} & J_y & J_{yz} \\ J_{zx} & J_{zy} & J_z \end{bmatrix} \end{matrix} \quad \begin{matrix} \text{Rotationssymmetrischer Körper} \\ \begin{bmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{bmatrix} \end{matrix} \quad \begin{matrix} \text{Kugel } J_x=J_y=J_z \\ \begin{bmatrix} J & 0 & 0 \\ 0 & J & 0 \\ 0 & 0 & J \end{bmatrix} = J\bar{\bar{I}} \end{matrix} \quad (3.8)$$

Aus Gleichung 3.7 und Gleichung 3.8 folgt somit, dass die Summe der Drehmomente, die auf das Partikel wirken, proportional zur zeitlichen Änderung der Winkelgeschwindigkeit des Partikels, also der Winkelbeschleunigung des Partikels ist.

$$\sum \bar{\bar{M}} = \frac{d\bar{\bar{L}}}{dt} = \bar{\bar{\theta}} \frac{d\bar{\bar{\omega}}}{dt} = J\bar{\bar{I}}\bar{\bar{\alpha}} = J\bar{\bar{\alpha}} \quad (3.9)$$

Wird diese Gleichung für n Partikel im dreidimensionalen Raum verallgemeinert, dann folgt daraus, dass die Winkelbeschleunigungsmatrix gleich dem Produkt aus inverser Massenträgheitsmomentmatrix und der Drehmomentmatrix ist.

$$\bar{\bar{\alpha}} = \bar{\bar{J}}^{-1} \sum \bar{\bar{M}} \quad (3.10)$$

$$\text{mit } \begin{bmatrix} \alpha_{1x} & \alpha_{1y} & \alpha_{1z} \\ \alpha_{2x} & \alpha_{2y} & \alpha_{2z} \\ \vdots & \vdots & \vdots \\ \alpha_{nx} & \alpha_{ny} & \alpha_{nz} \end{bmatrix} = \begin{bmatrix} J_1 & 0 & \dots & 0 \\ 0 & J_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & J_n \end{bmatrix}^{-1} \sum \begin{bmatrix} M_{1x} & M_{1y} & M_{1z} \\ M_{2x} & M_{2y} & M_{2z} \\ \vdots & \vdots & \vdots \\ M_{nx} & M_{ny} & M_{nz} \end{bmatrix} \quad (3.11)$$

Der erste Index zeigt dabei die Nummer des Partikels an und der zweite Index die Drehachse. Auf diese Weise lassen sich aus den äußeren Momenten zu jedem Zeitpunkt die Winkelbeschleunigungen auf das Partikel berechnen.



### 3.3 Approximation der Partikelbewegung

Um aus den Bewegungsgleichungen der Partikel die Positions- und Rotationswinkeländerung der Partikel zu berechnen, müssen die Gleichungen 3.4 und 3.10 doppelt über die Zeit integriert werden. Der Übersichtlichkeit halber werden die folgenden Formeln nur anhand eines Partikels erläutert.

$$\vec{r} = \int \left( \int \vec{a} \, dt \right) dt = \int \left( \int m^{-1} \sum \vec{F} \, dt \right) dt \quad (3.12)$$

$$\vec{\varphi} = \int \left( \int \vec{\alpha} \, dt \right) dt = \int \left( \int J^{-1} \sum \vec{M} \, dt \right) dt \quad (3.13)$$

Dies ist nur für sehr einfache Fälle, z.B. wenn nur eine konstante Kraft, wie die Schwerkraft auf das Partikel wirkt (s. Gleichung 3.14), analytisch lösbar.

$$\vec{r} = \int \left( \int \vec{a} \, dt \right) dt = m^{-1} \int \left( \int \vec{F}_g \, dt \right) dt = m^{-1} \int \left( \int m \vec{g} \, dt \right) dt = \frac{1}{2} \vec{g} t^2 + \vec{v}_0 t + \vec{r}_0 \quad (3.14)$$

Da aber, wie in Kapitel 3.5 noch beschrieben wird, in den Kraft- und Momententermen die Variablen für Position, Geschwindigkeit und Winkelgeschwindigkeit selbst auftauchen, kann die Gleichung nicht analytisch gelöst werden. Um die Gleichung trotzdem zu lösen, wird das Integral über eine Reihenfunktion approximiert. Am besten bietet sich hierfür eine Taylor-Reihe mit der Entwicklungsstelle 0, die sogenannte MacLaurin Reihe an, die im allgemeinen Fall folgendermaßen aussieht.

$$f(x_0 + h) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!} h^n = f(x_0) + f'(x_0)h + \frac{1}{2} f''(x_0)h^2 + \dots \quad (3.15)$$

Wird diese für die Position und die Geschwindigkeit angewendet, ergeben sich folgende Gleichungen:

$$\begin{aligned} \vec{r}(t_0 + \Delta t) &= \vec{r}(t_0) + \frac{d\vec{r}}{dt}(t_0) \cdot \Delta t + \frac{1}{2} \frac{d^2\vec{r}}{dt^2}(t_0) \cdot \Delta t^2 \\ &= \vec{r}(t_0) + \vec{v}(t_0) \cdot \Delta t + \frac{1}{2} \vec{a}(t_0) \cdot \Delta t^2 \end{aligned} \quad (3.16)$$

$$\vec{v}(t_0 + \Delta t) = \vec{v}(t_0) + \frac{d\vec{v}}{dt}(t_0) \cdot \Delta t = \vec{v}(t_0) + \vec{a}(t_0) \cdot \Delta t \quad (3.17)$$



Für den Rotationswinkel und die Winkelgeschwindigkeit ergeben sich analoge Gleichungen.

$$\begin{aligned}\bar{\varphi}(t_0 + \Delta t) &= \bar{\varphi}(t_0) + \frac{d\bar{\varphi}}{dt}(t_0) \cdot \Delta t + \frac{1}{2} \frac{d^2\bar{\varphi}}{dt^2}(t_0) \cdot \Delta t^2 \\ &= \bar{\varphi}(t_0) + \bar{\omega}(t_0) \cdot \Delta t + \frac{1}{2} \bar{\alpha}(t_0) \cdot \Delta t^2\end{aligned}\tag{3.18}$$

$$\bar{\omega}(t_0 + \Delta t) = \bar{\omega}(t_0) + \frac{d\bar{\omega}}{dt}(t_0) \cdot \Delta t = \bar{\omega}(t_0) + \bar{\alpha}(t_0) \cdot \Delta t\tag{3.19}$$

Auf diese Weise lassen sich die Werte für Position und Geschwindigkeit bereits ermitteln. Die noch fehlenden Werte, Beschleunigung und Winkelbeschleunigung, werden über die Bewegungsgleichungen 3.4 und 3.10 ermittelt.

### 3.4 Numerische Integrationsverfahren

Es gibt eine Vielzahl von numerischen Integrationsverfahren, allerdings lassen sich nicht alle auf das hier vorliegende Problem anwenden. In der Moleküldynamik, einer zur DEM verwandten Disziplin, werden für die doppelte Integration der Bewegungsgleichung Verfahren, wie z.B. Störmer-Verlet, Velocity-Verlet oder die Leap-Frog benutzt [Lit6]. Allerdings sind all diese Verfahren darauf ausgelegt, dass die Beschleunigung bzw. Kraft im nächsten Zeitschritt allein von der Position abhängig ist. Da im vorliegenden Fall aber geschwindigkeitsproportionale Dämpfung, sowie geschwindigkeitsabhängige Fluidwiderstandskraft auftritt, ist die Kraft sowohl von der Position, als auch von der Geschwindigkeit abhängig. Aus diesem Grund werden hier zwei andere Verfahren, das explizite Euler-Verfahren und das Praediktor-Korrektor-Verfahren nach Gear verwendet. Diese beiden Integrationsverfahren werden in den Kapiteln 3.4.1 und 3.4.2 anhand der translatorischen Bewegung erklärt. Die Berechnung für die Rotation ist analog hierzu.

#### 3.4.1 Explizites Euler Verfahren

Das explizite Euler-Verfahren, auch Polygonzugverfahren genannt, ist ein numerisches Verfahren von 1. Ordnung. Es wurde von Leonard Euler entwickelt und ist das einfachste Verfahren für die Lösung eines Anfangswertproblems.

Generell muss für die Anwendung des expliziten Euler-Verfahrens ein Anfangswertproblem der folgenden Form vorliegen.

$$\dot{y} = f(t, y) \text{ mit } y(t_0) = y_0 \quad (3.20/3.21)$$

Dann lässt sich der Funktionswert im folgenden Zeitschritt über die folgende Formel berechnen:

$$y(t_0 + \Delta t) = y(t_0) + f(t_0, y_0) \cdot \Delta t \quad (3.22)$$

Bei Vergleich von Formel 3.22 mit der Formel 3.17 fällt auf, dass sich diese stark ähneln. Hierbei entsprechen die Geschwindigkeit der Funktion  $y$  und die Beschleunigung der Funktion  $f$ , die die Ableitung von  $y$  ist. Somit entspricht die MacLaurin-Reihe mit zwei Gliedern dem expliziten Euler-Verfahren.

Für die Berechnung der Position kann man das explizite Euler Verfahren nicht verwendet, da für die Position folgendes Anfangswertproblem vorliegt:

$$\ddot{r} = f(t, \dot{r}, r) \text{ mit } r(t_0) = r_0 \text{ und } \dot{r}(t_0) = v_0 \quad (3.23)$$

Somit muss für die Positionsberechnung die MacLaurin-Reihe mit drei Gliedern nach Gleichung 3.16 verwendet werden. In Abbildung 3.2 ist das verwendete Verfahren, benannt als modifiziertes explizites Euler-Verfahren, schematisch dargestellt. Hierbei wird zunächst die Beschleunigung  $a(t_0)$  über die Bewegungsgleichung aus den bekannten Positions- und Geschwindigkeitswerten der vorherigen Berechnung ermittelt und anschließend werden die Geschwindigkeit  $v(t_0 + \Delta t)$  und die Position  $r(t_0 + \Delta t)$  für den nächsten Zeitschritt aus der Beschleunigung  $a(t_0)$  berechnet.

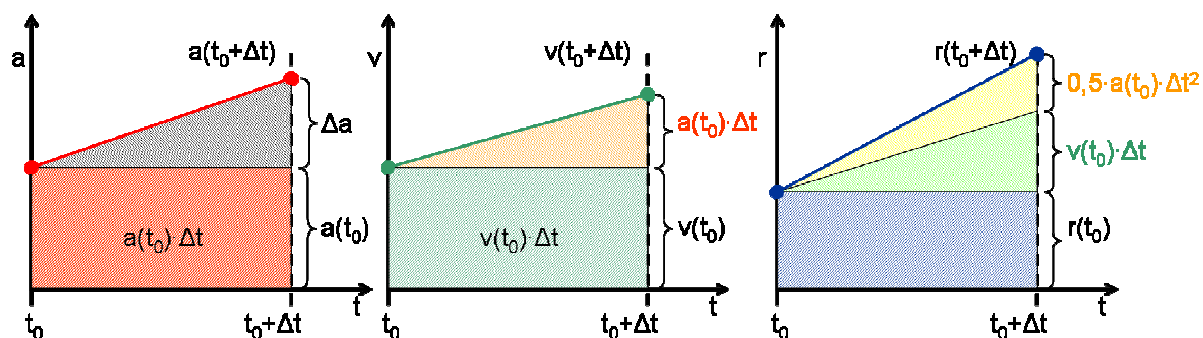


Abb. 3.2: Modifiziertes explizites Euler-Verfahren

### 3.4.2 Gear Prädiktor Korrektor

Eine Verbesserung des expliziten Euler Verfahrens ist das Prädiktor-Korrektor-Verfahren nach Gear. Es handelt sich hierbei um ein iteratives Integrationsverfahren 2. Ordnung [Lit1]. Hierbei werden zunächst die Positionen und Geschwindigkeiten für den folgenden Zeitschritt explizit nach den Formeln 3.16 und 3.17 als sogenannte Prädiktorwerte ausgerechnet. Die Prädiktorbeschleunigung  $\bar{a}^p$  ist gleich der Beschleunigung des vorherigen Zeitschrittes. Anschließend werden die Prädiktorwerte in die Bewegungsgleichung eingesetzt und die Beschleunigung ermittelt. Diese Beschleunigung  $\bar{a}^c$  wird als Korrektorwert bezeichnet. In Abbildung 3.3 ist der Korrektor-Schritt schematisch dargestellt.

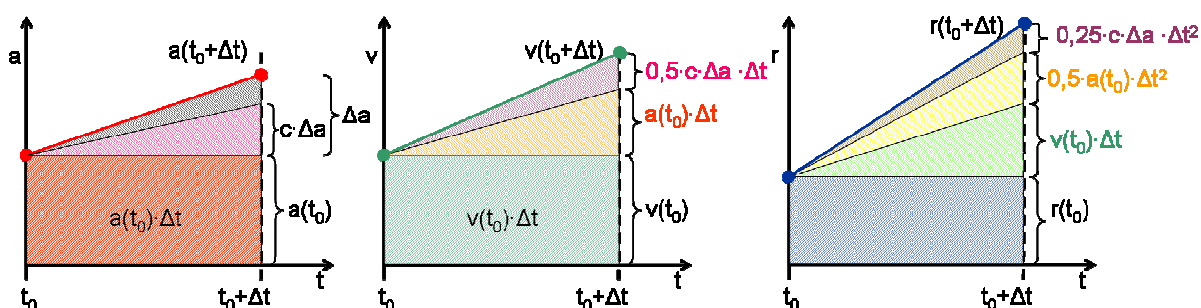


Abb. 3.3: Korrektor-Schritt des Prädiktor-Korrektor-Verfahrens

Über die folgende Formel wird die Differenz aus Korrektor- und Prädiktorbeschleunigung bestimmt.

$$\Delta\bar{a} = \bar{a}^c - \bar{a}^p \quad (3.24)$$

Anschließend wird überprüft, ob die einzelnen Elemente von  $\Delta\bar{a}$  kleiner als das Abbruchkriterium  $\varepsilon$  sind. Ist dies der Fall so ist  $\bar{a}^c$  die Beschleunigung des aktuellen Zeitschrittes und der nächste Zeitschritt kann berechnet werden. Ist auch nur ein Element von  $\Delta\bar{a}$  größer als das Abbruchkriterium  $\varepsilon$ , so wird innerhalb einer Iterationsschleife nach den Formeln 3.25, 3.26 und 3.27 neue Prädiktorwerte für die Position, die Geschwindigkeit und die Beschleunigung ermittelt. Dies wird solange wiederholt, bis das Abbruchkriterium erreicht wird.

$$\bar{r}^p := \bar{r}^p + \frac{1}{4}c \cdot \Delta\bar{a} \cdot \Delta t^2 \quad (3.25)$$

$$\bar{v}^p := \bar{v}^p + \frac{1}{2}c \cdot \Delta\bar{a} \cdot \Delta t \quad (3.26)$$



$$\bar{a}^p := \bar{a}^p + c \cdot \Delta \bar{a} \quad (3.27)$$

Die Konstante  $c$  ist hierbei eine Art Unterrelaxationsfaktor, der zwischen 0 und 1 liegen darf. Über ihn lässt sich die „Härte“ der Aktualisierung einstellen. Ist  $c=1$ , so wird der Prädiktorwert direkt auf den Korrektorwert gesetzt. Dies kann zu Konvergenzproblemen führen, da so auch Weg und Geschwindigkeit korrigiert werden und somit eine stark veränderte Korrekturbeschleunigung auftreten kann.

### 3.5 Kräfte und Momente

Um die Beschleunigung über die Bewegungsgleichung zu erhalten, müssen zunächst die Kräfte und Momente, die auf die einzelnen Partikel wirken genauer betrachtet werden. Laut Aufgabenstellung sind dies die Kontaktkräfte und -momente zwischen den Partikeln, die Schwerkraft, sowie die Widerstandskraft aufgrund eines Fluids. Zusätzlich werden noch der statische Auftrieb, sowie die Gravitationskraft betrachtet, da diese in das zu implementierende Programm eingebaut werden sollen.

Insgesamt können drei unterschiedliche Krafttypen unterschieden werden, die bei der späteren Implementierung des Programms auch unterschiedlich berücksichtigt werden müssen. Als erstes gibt es statische Kräfte, diese sind konstant und wirken auf alle Partikel, unabhängig von Position des Partikels und Zeit. Als zweites gibt es dynamische Kräfte, diese wirken auch auf alle Partikel sind aber aufgrund ihrer Zeitabhängigkeit nicht konstant. Als drittes gibt es die Kontaktkräfte, diese wirken nur auf Partikel die sich in Kontakt befinden und sind durch ihre zusätzliche Zeitabhängigkeit auch nicht konstant. Somit ergibt sich, dass die statischen Kräfte nur einmal zu Beginn der Iteration ermittelt werden müssen, während die anderen Kräfte in jedem Zeitschritt neu berechnet werden müssen.

	Umwelt-Partikel	Partikel-Partikel	Fluid-Partikel
statisch	Gewichtskraft	-	Statischer Auftrieb*
dynamisch	-	Gravitation	Fluidwiderstand
Im Kontaktfall	Rand-Partikel	Kontaktkräfte	-

Tab. 3.1: Unterschiedliche Arten von Kräften

\* bei konstanter Dichte des Fluids

### 3.5.1 Partikel-Partikel Interaktion

Das Hauptaugenmerk bei der Diskrete Element Method liegt auf der Berechnung von Kontaktkräften zwischen zwei Partikeln. Als erster Schritt hierfür muss erst einmal definiert werden, wann ein Kontakt ereignis auftritt und wie dieses berechnet wird. Ein Kontakt ereignis tritt dann auf, sobald sich beide Partikeloberflächen durchdringen. Je nach Partikelgeometrie kann dies sehr kompliziert zu bestimmen sein. Da in dieser Arbeit aber nur mit kreisförmigen Partikeln gerechnet werden soll, lässt sich der Kontakt wie folgt bestimmen: Kontakt herrscht dann, wenn der Abstand der Partikelmittelpunkte subtrahiert mit der Summe der beiden Partikelradien negativ ist.

$$\delta_{ij} = |\vec{r}_i - \vec{r}_j| - R_i - R_j \quad (3.28)$$

Hierbei ist  $\vec{r}$  der Positionsvektor der Mittelpunkte der beiden Partikel  $i$  und  $j$  und  $R$  der Radius der jeweiligen Partikel. Das Ergebnis  $\delta_{ij}$  wird als Überlappung definiert. Wenn die Partikel nicht im Kontakt sind ist  $\delta$  positiv, bei gerade stattfindender Berührung ist  $\delta$  Null und bei Kontakt ist  $\delta$  negativ (s. Abb. 3.4).

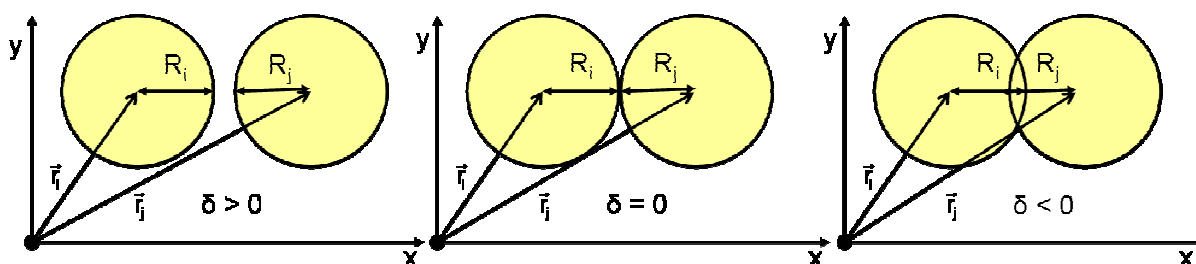


Abb. 3.4: Kontakterkennung zwischen Partikeln

Als Kontaktmodell wird das Soft-Particle Modell verwendet, welches in den nächsten Kapiteln detaillierter beschrieben wird. Wie bereits in Kapitel 2.2.1 erwähnt, beruht das Soft-Particle Modell auf der Annahme, dass die Partikel im Kontaktfall durch ein komplexes Feder-Dämpfer-System miteinander verbunden sind. Dabei gibt es jeweils ein parallel geschaltetes Feder-Dämpfer-System in Richtung der Berührungsnormale und tangential zu dieser. Beide Systeme sind über eine Reibkupplung gekoppelt. Außerdem sind beide Partikel über eine Parallelschaltung von Drehfeder und Drehdämpfer miteinander verbunden, damit ein realitätsnahes aufeinander Abrollen der Partikel stattfinden kann. Im Folgenden werden die einzelnen Bauteile des Feder-Dämpfer-Systems mathematisch beschrieben.

### 3.5.2 Normalenrichtung

Die Normalenrichtung ist die Richtung der Berührungsnormalen, welche im Kontaktfall als Vektor zwischen den beiden Partikelmittelpunkten definiert ist. Für die Normalenrichtung wird ein Feder-Dämpfer-System wie in Abbildung 3.5 benutzt. Dieses wird für die Berechnung des zentrischen Stoßes benötigt.

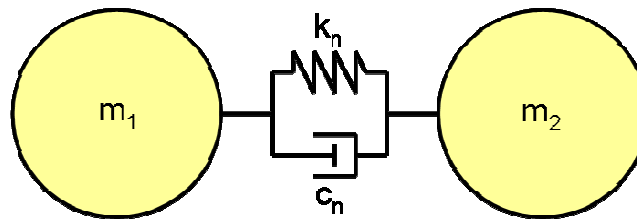


Abb. 3.5: Feder-Dämpfer Modell in Normalenrichtung

#### 3.5.2.1 Feder in Normalenrichtung

Für die mathematische Beschreibung der Feder gibt es unterschiedliche Modelle, wobei ein lineares Modell mit konstanter Federsteifigkeit das einfachste wäre. Bei der Recherche fiel auf, dass aber am häufigsten für kugel- bzw. kreisförmige Partikel ein nichtlineares Federmodell verwendet wird. Dieses Federmodell beruht auf der Ausarbeitung von Heinrich Hertz [Lit10] aus dem Jahre 1881. Hierin wird die Druckverteilung zwischen zwei kugelförmigen Körpern, die miteinander in Kontakt stehen hergeleitet. Dieses Modell gilt nur für Verformungen der Kugeln, die im Vergleich zum Radius der Objekte relativ klein sind. Außerdem wird ein rein-elastisches Materialverhalten nach dem Hookeschen Federgesetz vorausgesetzt. In der Arbeit von Johnson [Lit12] wird diese Theorie weitergeführt. Letztendlich wird eine Gleichung für die Kraft erhalten, die zwischen zwei kugelförmigen Körpern im Kontaktfall entsteht. Diese Kraft lässt sich über folgende Formel berechnen:

$$F_{ij} = -k_{\text{Hz}} \delta_{ij}^{3/2} = -k_n (\delta_{ij}) \delta_{ij} \quad (3.29)$$

$\delta$  ist in diesem Fall die Überlappung der beiden Kugeln und lässt sich nach folgender Formel berechnen.

$$\delta_{ij} = |\vec{r}_i - \vec{r}_j| - R_i - R_j \quad (3.30)$$



Die sogenannte Hertzsche Federsteifigkeit  $k_{Hz}$  kann nach [Lit15, Lit20] mit folgender Formel bestimmt werden.

$$k_{Hz} = \frac{4}{3} E_{\text{eff}} \sqrt{R_{\text{eff}}} \quad (3.31)$$

Hierbei ist  $E_{\text{eff}}$  der effektive Elastizitätsmodul der beiden Kugeln. Er lässt sich mittels Gleichung 3.32 ermitteln.  $R_{\text{eff}}$  beschreibt den effektiven Radius. Dieser lässt sich laut [Lit10, Lit15] über Gleichung 3.33 bestimmen.

$$E_{\text{eff}} = \left( \frac{1 - \nu_i^2}{E_i} + \frac{1 - \nu_j^2}{E_j} \right)^{-1} \quad (3.32)$$

$$R_{\text{eff}} = \left( \frac{1}{R_i} + \frac{1}{R_j} \right)^{-1} \quad (3.33)$$

Insgesamt ergibt sich somit folgende Kraft auf die beiden Partikel:

$$F_{ij} = -k_{Hz} \delta_{ij}^{3/2} = -\frac{4}{3} E_{\text{eff}} \sqrt{R_{\text{eff}}} \delta_{ij}^{3/2} \quad (3.34)$$

In Abbildung 3.6 ist das Kraft-Weg Diagramm einer Stahlkugel mit 1m Durchmesser dargestellt. Der Weg entspricht hierbei der Überlappung  $\delta$ .

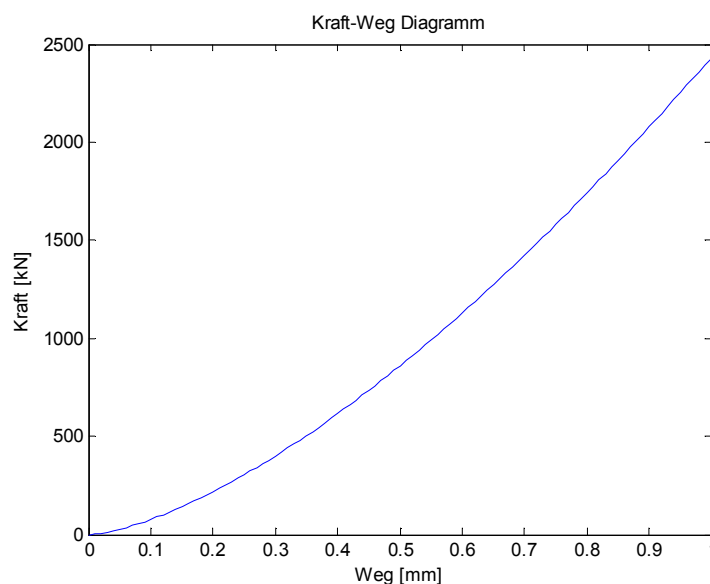


Abb. 3.6: Kraft-Weg Diagramm des Kugel-Kugel Kontaktes



### 3.5.2.2 Dämpfer in Normalenrichtung

Für die mathematische Beschreibung der Dämpfungskraft zwischen zwei Partikeln wurde in [Lit15] die Gleichung 3.35 gefunden. Diese ist proportional zur relativen Geschwindigkeit der beiden Partikel, die über Gleichung 3.36 berechnet werden kann.

$$F_{ij} = -c_n(\delta_{ij}) \cdot \vec{v}_{rel} \cdot \vec{n} \quad (3.35)$$

$$\vec{v}_{rel} = \vec{v}_i - \vec{v}_j + \vec{\omega}_i \times (\mathbf{R}_i \cdot \vec{n}) - \vec{\omega}_j \times (-\mathbf{R}_j \cdot \vec{n}) \quad (3.36)$$

$$c_n = \alpha \sqrt{m_{eff}} \cdot k_{Hz} \sqrt[4]{\delta_{ij}} \quad (3.37)$$

Die Dämpfungskonstante in Normalenrichtung kann über Formel 3.37 [Lit15] ermittelt werden. Allerdings wurde hierzu keine Herleitung gefunden, weshalb diese über das, aus Schwingungsberechnungen bekannte, Lehrsche Dämpfungsmaß hergeleitet wird. Für eine lineare Differentialgleichung eines Schwingungssystems ist das Lehrsche Dämpfungsmaß als das Verhältnis von Abklingkonstante  $\delta_0$  zur Eigenfrequenz  $\omega_0$  definiert.

$$D_n = \frac{\delta_0}{\omega_0} \quad (3.38)$$

Dabei ist die Eigenfrequenz die Wurzel aus dem Verhältnis von Steifigkeit  $k$  zur Masse des Systems  $m_{eff}$  (s. Formel 3.39). Die Abklingkonstante ist das Verhältnis von Dämpfungskonstante  $c$  und der doppelten Masse des Systems (s. Formel 3.40).

$$\omega_0 = \sqrt{\frac{k}{m_{eff}}} \quad \text{und} \quad \delta_0 = \frac{c}{2m_{eff}} \quad (3.39/3.40)$$

Die effektive Systemmasse lässt sich nach folgender Formel bestimmen.

$$m_{eff} = \left( \frac{1}{m_i} + \frac{1}{m_j} \right)^{-1} \quad (3.41)$$

Da diese Beziehung nur für lineare Systeme gilt, wird versucht das vorhandene nichtlineare System zu quasi-linearisieren. Dazu wird der nichtlineare Anteil in der Federkraft mit in die Federkonstante gezogen, so dass gilt:

$$F_{ij} = -k_{Hz} \delta_{ij}^{3/2} = -\underbrace{k_{Hz} \sqrt{\delta_{ij}}}_{kn(\delta_{ij})} \delta_{ij} = -k_n(\delta_{ij}) \delta_{ij} \quad (3.42)$$

Dieser Term kann in die Formel für das Lehrsche Dämpfungsmaß eingefügt werden.

$$D_n = \frac{c\sqrt{m_{\text{eff}}}}{2m_{\text{eff}}\sqrt{k_n(\delta_{ij})}} = \frac{c}{2\sqrt{m_{\text{eff}} \cdot k_n(\delta_{ij})}} \quad (3.43)$$

Diese Gleichung lässt sich nach der Dämpfungskonstanten umstellen. Da der nichtlineare Term  $\sqrt[4]{\delta}$  in dieser vertreten ist, ist diese allerdings nicht mehr konstant, sondern ändert sich während des Kontaktes

$$c_n = 2D_n\sqrt{m_{\text{eff}} \cdot k_n(\delta_{ij})} = 2D_n\sqrt{m_{\text{eff}} \cdot k_{\text{Hz}}} \cdot \sqrt[4]{\delta} \quad (3.44)$$

Bei Vergleich mit Formel 3.37 stellt sich heraus, dass das darin enthaltene  $\alpha$  dem doppelten Dämpfungsmaß entspricht.

$$\alpha = 2D_n \quad (3.45)$$

Insgesamt ergibt sich somit folgende Dämpfungskraft auf die beiden Partikel:

$$F_{ij} = -2D_n\sqrt{m \cdot k_{\text{Hz}}} \cdot \sqrt[4]{\delta} \vec{v}_{\text{rel}} \vec{n} = -c_n \vec{v}_{\text{rel}} \vec{n} \quad (3.46)$$

### 3.5.3 Tangentialrichtung

Um den exzentrischen Stoß mit Hilfe der DEM abzubilden wird neben dem Feder-Dämpfer-System in Normalenrichtung ein zweites Feder-Dämpfer-System in Richtung der Berührungstangenten eingesetzt (s. Abb. 3.7).

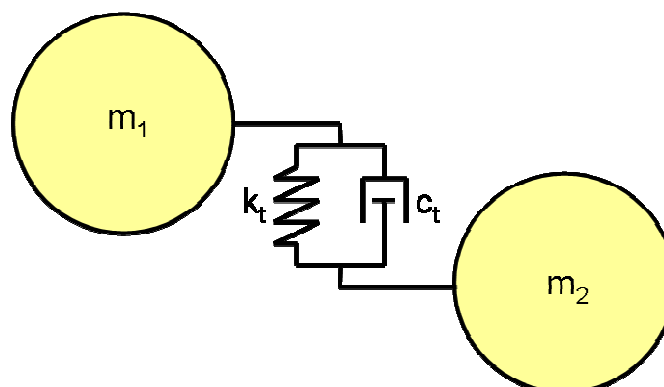


Abb. 3.7: Feder-Dämpfer Modell in Tangentialrichtung



### 3.5.3.1 Feder in Tangentialrichtung

Die Federkraft in Tangentialrichtung lässt sich nach [Lit15, Lit19] durch Gleichung 3.47 ausdrücken. Sie ist proportional zur tangentialen Verschiebung der Partikel, die während des Kontaktes auftritt.

$$F_{ij} = -k_t \delta_t \quad (3.47)$$

Die Federkonstante ist nach [Lit15, Lit19] durch folgende Gleichung zu bestimmen:

$$k_t = 8G_{\text{eff}} \sqrt{R_{\text{eff}}} \delta \quad (3.48)$$

Der effektive Schubmodul kann laut [Lit8] durch das folgende Verhältnis bestimmt werden.

$$G_{\text{eff}} = \frac{E_{\text{eff}}}{2(1 + \nu_{\text{eff}})} \quad (3.49)$$

Somit ergibt sich für die Federsteifigkeit in Tangentialrichtung:

$$k_t = 4 \frac{E_{\text{eff}} \sqrt{R_{\text{eff}}} \delta}{(1 + \nu_{\text{eff}})} \quad (3.50)$$

Die effektive Poisson-Zahl lässt sich auf folgende Weise berechnen.

$$\nu_{\text{eff}} = \left( \frac{1}{\nu_i} + \frac{1}{\nu_j} \right)^{-1} \quad (3.51)$$

Die tangentialen Verschiebung oder auch tangentialen Überlappung wird laut [Lit15, Lit19] als Integral der relativen tangentialen Geschwindigkeit über die Zeit berechnet.

$$\delta_t = \left( \int_{t_0}^t \vec{v}_{\text{rel},t}(\tau) d\tau \right) \cdot \vec{t} \quad (3.52)$$

### 3.5.3.2 Dämpfer in Tangentialrichtung

Für die Dämpfungskonstante in tangentialer Richtung  $c_t$  wird laut [Lit2, Lit15] dieselbe Formel verwendet, wie für den Dämpfer in Normalenrichtung, lediglich das Dämpfungsmaß  $D_t$  lässt sich unabhängig davon einstellen.

$$c_t = 2D_t \sqrt{m_{\text{eff}} \cdot k_n(\delta_{ij})} = 2D_t \sqrt{m_{\text{eff}} \cdot k_{\text{Hz}}} \sqrt[4]{\delta_{ij}} \quad (3.53)$$

Die Dämpfungskraft ist proportional zur tangentialen Relativgeschwindigkeit der Partikel und lässt sich nach der folgenden Formel ermitteln.

$$\vec{F}_{ij} = -2D_t \sqrt{m_{\text{eff}} \cdot k_{\text{Hz}}} \cdot \sqrt[4]{\delta_{ij}} \cdot \vec{v}_{\text{rel}} \vec{t} = -c_t \cdot \vec{v}_{\text{rel}} \vec{t} \quad (3.54)$$

### 3.5.4 Reibung

Zwischen zwei Partikeln mit nicht ideal glatter Oberfläche herrscht bei Kontakt Reibung. Die Reibung ist dabei eine Kraft, die die Relativbewegung der Partikel in tangentialer Richtung abbremst. Laut [Lit7] lässt sich die Reibkraft als Produkt aus Reibungskoeffizient  $\mu$  und Normalkraft  $F_n$  bestimmen. Für das in dieser Arbeit verwendete Kontaktmodell wird die Reibung zur Kopplung von Normalkraft und Tangentialkraft eingesetzt. Somit kann nur der Anteil der Tangentialkraft von einem Partikel auf das andere übertragen werden, der kleiner gleich der Normalkraft ist. Somit wird die Übertragung von Tangentialkräften durch die Reibung begrenzt (s. Gleichung 3.55).

$$|\vec{F}_t| \leq \mu \cdot |\vec{F}_n| \quad (3.55)$$

Abschließend lassen sich alle Einzelmodelle zu dem bereits aus Kapitel 2 bekannten Modell zusammensetzen. Dieses ist in Abbildung 3.8 schematisch dargestellt.

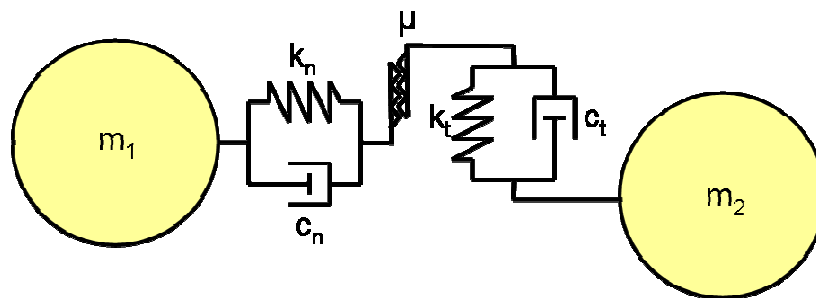


Abb. 3.8: Feder-Dämpfer Modell mit Reibkopplung

### 3.5.5 Drehrichtung

Um das Abrollen der Partikel aufeinander abzubilden, wird nach [Lit15] ein System bestehend aus Drehfeder und Drehdämpfer zusätzlich zwischen beide Partikel eingebaut. Bei der Literaturrecherche stellte sich heraus, dass dieses Feder-Dämpfer-System nur selten verwendet wird. Aus diesem Grund war es leider auch

nicht möglich Formeln für die Drehsteifigkeit  $k_r$  und die Drehdämpfung  $c_r$  zu ermitteln. Hier müssen empirische Werte eingesetzt werden, die am besten zum jeweiligen Versuch passen.

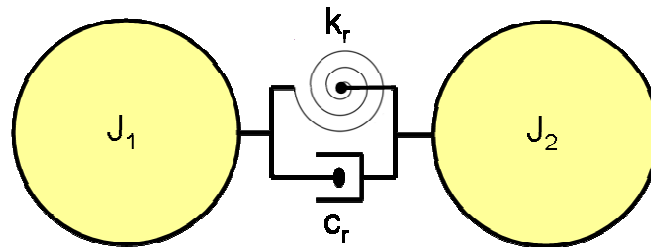


Abb. 3.9: Feder-Dämpfer Modell in Drehrichtung

Durch die Drehfeder wirkt ein Drehmoment auf die beiden Partikel, dieses lässt sich durch die folgende Formel bestimmen.

$$M_{k,ij} = -k_r \left( \int_{t_0}^t \vec{v}_{\text{roll},ij}(\tau) d\tau \right) \vec{t} \quad (3.56)$$

Hierzu wird allerdings noch die relative Abrollgeschwindigkeit der beiden Partikel benötigt, die wie folgt berechnet werden kann.

$$\vec{v}_{\text{roll},ij} = -R_{\text{eff}} (\vec{\omega}_i - \vec{\omega}_j) \times \vec{n} - \frac{1}{2} \frac{R_j - R_i}{R_j + R_i} \vec{v}_t \quad (3.58)$$

Abschließend wird das Drehmoment, das durch den Drehdämpfer auf das Partikel wirkt, durch die folgende Formel bestimmt.

$$M_{c,ij} = -c_r \vec{v}_{\text{roll},ij} \vec{t} \quad (3.59)$$

### 3.5.6 Gewichtskraft

Die Gewichtskraft, die auf ein Partikel wirkt, ist gleich dem Produkt aus Partikelmasse und Erdbeschleunigung. Die Gewichtskraft ist dabei immer in Richtung der Erdbeschleunigung gerichtet.

$$\vec{F}_g = m_p \vec{g} \quad (3.60)$$

### 3.5.7 Randbedingungen

Bei den Randbedingungen handelt es sich in diesem Fall um feste Wände, die mit den Partikeln interagieren. Je nach Einstellung des Reibungskoeffizienten  $\mu_w$  zwischen Partikel und Wand kann eine reibungsfreie oder eine reibungsbehaftete Wand realisiert werden. Bei einer reibungsfreien Wand fallen die tangentialen Anteile der Kraft weg und das Partikel gleitet auf der Wand ab. Bei einer reibungsbehafteten Wand führen die tangentialen Kraftanteile dazu, dass das Partikel anfängt auf der Wand abzurollen.

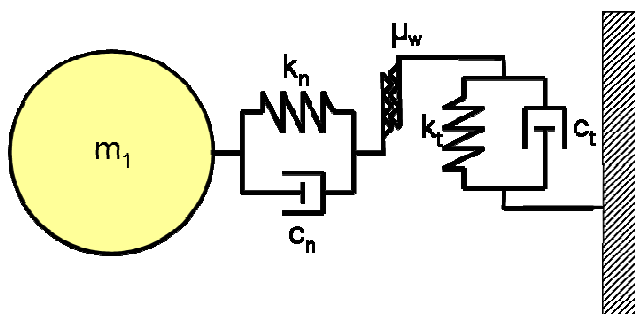


Abb. 3.10: Feder-Dämpfer Modell zwischen Partikel und Wand

Die Randbedingungen für ein DEM Modell werden auf ähnliche Weise realisiert, wie der Partikel-Partikel Kontakt (s. Abb. 3.10 und Abb. 3.11). Auch hier wird zwischen Wand und Partikel dasselbe Feder-Dämpfer System eingesetzt wie beim Partikel-Partikel Kontakt. Lediglich die Werte für den effektiven Radius und die effektive Masse sind unterschiedlich.

Für eine Wand werden folgende Bedingungen angenommen. Sie hat einen Elastizitätsmodul  $E_w$ , eine Poisson-Zahl  $\nu_w$  und einen Reibungskoeffizienten, der zwischen Partikel und Wand definiert ist. Die Wand wird dabei als eine Kugel mit unendlichem Radius und unendlicher Masse angenommen. Aus einer Grenzwertbildung ergeben sich somit folgende Eigenschaften für den effektiven Radius und die effektive Masse.

$$R_{\text{eff}} = \lim_{R_w \rightarrow \infty} \left( \frac{1}{R_p} + \frac{1}{R_w} \right)^{-1} = \left( \frac{1}{R_p} + 0 \right)^{-1} = R_p \quad (3.61)$$

$$m_{\text{eff}} = \lim_{m_w \rightarrow \infty} \left( \frac{1}{m_p} + \frac{1}{m_w} \right)^{-1} = \left( \frac{1}{m_p} + 0 \right)^{-1} = m_p \quad (3.62)$$

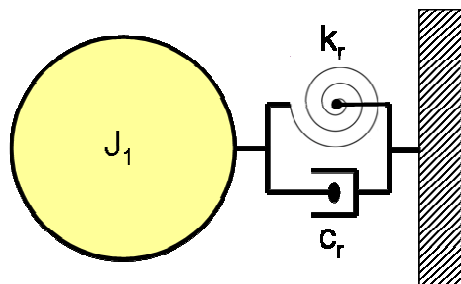


Abb. 3.11: Feder-Dämpfer Modell für die Drehfeder zwischen Partikel und Wand

### 3.5.8 Fluid-Partikel Interaktion

Die Fluid-Partikel Interaktion findet im Rahmen dieser Arbeit aufgrund zweier physikalischer Gegebenheiten statt. Zum einen gibt es den statischen Auftrieb, den jeder Gegenstand innerhalb eines Medium anderer Dichte erfährt. Zum anderen gibt es die Impulsübertragung vom Fluid auf das Partikel aufgrund des Strömungswiderstandes des Partikels. Die Impulsübertragung soll innerhalb dieser Arbeit nur in einfache Richtung erfolgen, so dass die Strömung zwar die Bewegung des Partikels beeinflusst, aber das Partikel keinen Einfluss auf die Strömung hat.

Nach dem archimedischen Prinzip ist die Auftriebskraft, die ein Körper erfährt, gleich der Gewichtskraft des von ihm verdrängten Fluidvolumens. Somit lässt sich die Auftriebskraft über folgende Gleichung ermitteln [Lit13]:

$$\vec{F}_A = -\rho_f V_p \vec{g} = -\frac{4}{3} \rho_f \pi R^3 \vec{g} \quad (3.63)$$

Hierbei ist  $\rho_f$  die Dichte des Fluids, in dem sich das Partikel mit dem Volumen  $V_p$  befindet.

Die Widerstandskraft auf einen Körper in einem Strömungsfeld lässt sich über die folgende von Lord Rayleigh entwickelte Gleichung bestimmen [Lit13]:

$$\vec{F}_W = -\frac{1}{2} \rho_f c_w A (\vec{v}_r \cdot \vec{v}_r) \frac{\vec{v}_r}{|\vec{v}_r|} \quad (3.64)$$

Hierbei ist  $\rho_f$  wieder die Dichte des Fluids,  $c_w$  ist der Widerstandsbeiwert und  $A$  ist die in Richtung der relativen Geschwindigkeit  $v_r$  projizierte Fläche des Partikels. Der  $c_w$ -





Wert eines kugelförmigen Körpers beträgt nach [Int1]  $c_w=0,47$  und die projizierte Fläche der Kugel lässt sich über die Formel zur Ermittlung einer Kreisfläche bestimmen.

$$A_p = \pi R_p^2 \quad (3.65)$$

Für die Bestimmung der Relativgeschwindigkeit wird die Strömungsgeschwindigkeit am Partikel benötigt. Dazu wird die Strömungsgeschwindigkeit am Partikelmittelpunkt interpoliert. Im Programm werden hierfür zwei unterschiedliche Verfahren verwendet, die nun kurz erläutert werden.

### 3.5.8.1 Bilineare Interpolation

Das erste Verfahren ist die Matlab Funktion `interp2`. Diese nutzt bei Standardeinstellung die bilineare Interpolation. Diese Methode dient dazu Zwischenwerte in einem zweidimensionalen strukturierten Raster zu bestimmen. Die folgende Beschreibung der Methode soll beispielhaft verdeutlichen, wie die bilineare Interpolation prinzipiell abläuft. In Matlab ist diese aufwändiger implementiert, so dass auch ganze Arrays mit Koordinaten gleichzeitig über Matrizenberechnungen interpoliert werden können.

Um die Fluidgeschwindigkeit am Partikelmittelpunkt mit den Koordinaten  $x_p$  und  $y_p$  zu ermitteln, werden zunächst lineare Interpolationen zwischen den bereits aus der Strömungssimulation bekannten Fluidgeschwindigkeiten  $v_f(x_1, y_1)$  und  $v_f(x_2, y_1)$ , sowie  $v_f(x_1, y_2)$  und  $v_f(x_2, y_2)$  durchgeführt. Diese liefern nach den Formeln 3.66 und 3.67 die interpolierten Fluidgeschwindigkeiten  $v_f(x_p, y_1)$  und  $v_f(x_p, y_2)$ .

$$v_f(x_p, y_1) = \frac{x_2 - x_p}{x_2 - x_1} v_f(x_1, y_1) + \frac{x_p - x_1}{x_2 - x_1} v_f(x_2, y_1) \quad (3.66)$$

$$v_f(x_p, y_2) = \frac{x_2 - x_p}{x_2 - x_1} v_f(x_1, y_2) + \frac{x_p - x_1}{x_2 - x_1} v_f(x_2, y_2) \quad (3.67)$$

Anschließend wird zwischen diesen beiden Punkten in die andere Koordinatenrichtung nach Formel 3.68 interpoliert. Dies liefert die interpolierte Fluidgeschwindigkeit am Partikelmittelpunkt.

$$v_f(x_p, y_p) = \frac{y_2 - y_p}{y_2 - y_1} v_f(x_p, y_1) + \frac{y_p - y_1}{y_2 - y_1} v_f(x_p, y_2) \quad (3.68)$$

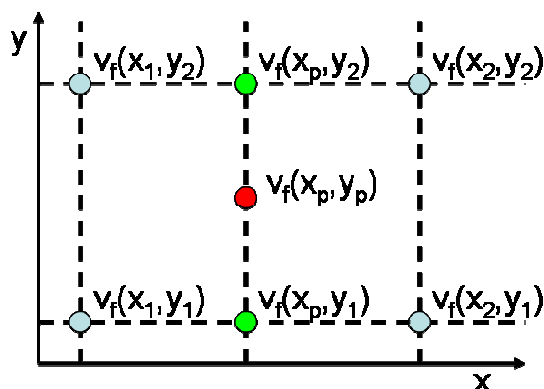


Abb. 3.12: Bilineare Interpolation

### 3.5.8.2 Shepards Method

Das zweite Verfahren ist die Shepards Method, die genau genommen kein Interpolationsverfahren ist. Es handelt sich um eine inverse Distanzwichtung, was bedeutet, dass der zu berechnende Wert aus seinen Nachbarwerten ermittelt wird. Diese werden dabei invers zu ihrem jeweiligen Abstand zum berechnenden Wert gewichtet, so dass nähere Nachbarn höher gewichtet werden als fernere Nachbarn.

Die Fluidgeschwindigkeit  $v_f$  am Partikelmittelpunkt mit den Koordinaten  $x_p$  und  $y_p$  lässt sich durch Summation der Produkte aus den Fluidgeschwindigkeiten  $v_i$  an den Nachbarpunkten mit den Koordinaten  $x_i$  und  $y_i$  und den Gewichtungsfunktion der Nachbarpunkte  $w_i$  bestimmen.

$$v_f(x_p, y_p) = \sum_{i=1}^N (v_i w_i) \quad \text{mit } v_i = v(x_i, y_i) \quad (3.69)$$

Es kann über eine beliebige Anzahl von Nachbarpunkten  $N$  summiert werden. Die Fluidgeschwindigkeiten  $v_i$  sind bekannt und die Gewichtungsfunktionen  $w_i$  lassen sich über die Formel (3.70) bestimmen.

$$w_i = \frac{h_i^{-p}}{\sum_{j=1}^N h_j^{-p}} \quad (3.70)$$

Hierbei ist  $p$  der Gewichtungsexponent der laut [Int8] in den meisten Fällen  $p=2$  beträgt.  $h_i$  ist der Abstand zwischen Partikelmittelpunkt und den Nachbarpunkten (s. Formel 3.71).

$$h_i = \sqrt{(x_p - x_i)^2 + (y_p - y_i)^2} \quad (3.71)$$



Diese Methode bietet im Vergleich zur interp2-Funktion einen großen Vorteil. Sie ist nicht auf strukturierte Netze bei der Strömungssimulation angewiesen.

### 3.5.9 Gravitationskräfte

Die Gravitationskraft spielt bei sehr kleinen Partikeln natürlich keine Rolle. Dieses Kraftmodell wurde nur zu Testzwecken für die spätere Implementierung abstandsabhängiger kontinuierlicher Kräfte in das Programm integriert. Dies sind z.B. die Van-der-Waals Kraft, die elektrostatische Kraft nach Coulomb usw., die für sehr feine Partikel eine große Rolle spielen, allerdings komplizierter zu bestimmen sind.

Nach dem Newtonschen Gravitationsgesetz ist die Kraft, die zwei Körper aufeinander ausüben gleich dem Produkt ihrer Massen durch ihren Abstand  $d$  ins Quadrat multipliziert mit der Gravitationskonstanten  $G$ .

$$F_G = G \frac{m_1 m_2}{d^2} \quad (3.72)$$

## 3.6 Zusammenhang von Stoßzahl und Lehrscher Dämpfung

In diesem Kapitel soll der Zusammenhang zwischen der Stoßzahl und dem Lehrschen Dämpfungsmaß erläutert werden. Die Stoßzahl oder auch der Restitutionskoeffizient  $e$  wird bei der Berechnung von Stößen bei denen ein Impulsaustausch zwischen zwei Körpern stattfindet, verwendet [Lit8]. Das Lehrsche Dämpfungsmaß  $D$  wird bei der Berechnung von Schwingungen von Feder-Masse-Dämpfer-Systemen verwendet. Beide erfüllen dabei den Zweck die Dissipation, die in einem System auftritt, mathematisch zu beschreiben. Bei einem rein-elastischen Stoß ist  $e=1$  und bei einem rein-plastischen Stoß ist  $e=0$ . Bei einer ungedämpften Schwingung ist  $D=0$  und bei Eintreten des aperiodischen Grenzfalles ist  $D=1$ .

Ein großes Problem, das hierbei auftritt ist, dass die Dämpfung im vorliegenden Fall nichtlinear ist. Die Dämpfungskonstante und somit auch das Lehrsche Dämpfungsmaß sind abhängig von der Überlappung  $\delta$ . In der Literatur [Lit2] wurde deshalb nur ein Diagramm (s. Abb. 3.13) gefunden, in dem der Zusammenhang

zwischen den beiden Größen dargestellt ist. Hierbei ist  $\alpha$  der Dämpfungsfaktor nach Formel 3.37 und entspricht dem doppelten Lehrschen Dämpfungsmaß.

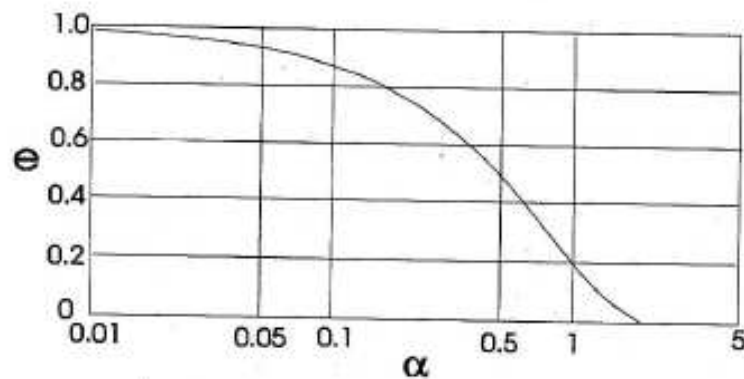


Abb. 3.13: Zusammenhang von Restitutionskoeffizient  $e$  und Dämpfungsfaktor  $\alpha$

Da das Ergebnis so recht unbefriedigend ist, wird nun der Zusammenhang zwischen Stoßzahl und Lehrscher Dämpfung bei linearer Dämpfung hergeleitet. Der Restitutionskoeffizient wird hierbei im allgemeinen Fall über das Verhältnis der relativen Geschwindigkeiten nach und vor dem Stoß berechnet.

$$e = \frac{v'_1 - v'_2}{v_1 - v_2} \quad (3.73)$$

Allerdings kann er auch über einen Fallversuch ermittelt werden. Dann ist der Restitutionskoeffizient gleich der Wurzel des Höhenverhältnisses nach und vor dem Stoß.

$$e = \sqrt{\frac{h'}{h}} \quad (3.74)$$

Das Lehrsche Dämpfungsmaß lässt sich auf folgende Weise durch das logarithmische Dekrement beschreiben.

$$D = \frac{\Lambda}{\sqrt{(2\pi)^2 + \Lambda^2}} \quad (3.75)$$

Das logarithmische Dekrement ist der natürliche Logarithmus des Höhenverhältnisses vor und nach dem Stoß.

$$\Lambda = \ln\left(\frac{h}{h'}\right) \quad (3.76)$$

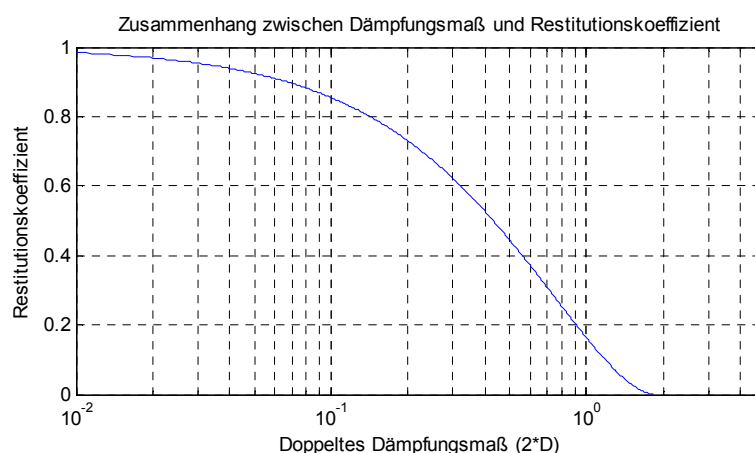
Gleichung 3.74 lässt sich nach dem Höhenverhältnis umstellen und anschließend in die Gleichung 3.76 einsetzen. Es entsteht folgendes Verhältnis zwischen logarithmischen Dekrement und Restitutionskoeffizient.

$$\Lambda = \ln(e^{-2}) \quad (3.77)$$

Dies lässt sich nun in Gleichung 3.75 einsetzen und es entsteht eine Formel, die die Abhängigkeit von Lehrschen Dämpfungsmaß und Restitutionskoeffizient anzeigt.

$$D = \frac{\ln(e^{-2})}{\sqrt{(2\pi)^2 + (\ln(e^{-2}))^2}} = \frac{-2 \cdot \ln(e)}{\sqrt{(2\pi)^2 + 4(\ln(e))^2}} \quad (3.78)$$

Anschließend wird der Restitutionskoeffizient in einem Diagramm über dem doppelten Lehrschen Dämpfungsmaß (dies entspricht  $\alpha$ ) aufgetragen (s. Abb. 3.14). Hierbei ist zunächst zu erkennen, dass beide Randbedingungen, wenn  $D=0$  ist, muss  $e=1$  sein und wenn  $2 \cdot D=2$  ist, muss  $e=0$  sein, recht gut eingehalten werden. Bei Vergleich mit Abbildung 3.13 fällt außerdem auf, dass die beiden Diagramme nahezu identisch sind. Von daher kann davon ausgegangen werden, dass die Nichtlinearität keinen allzu großen Einfluss auf den Zusammenhang beider Größen hat. Somit kann das Lehrsche Dämpfungsmaß, zumindest für die erste Annäherung, über Formel 3.78 aus der Stoßzahl ermittelt werden.



**Abb. 3.14: Restitutionskoeffizient aufgetragen über dem doppelten Lehrschen Dämpfungsmaß**

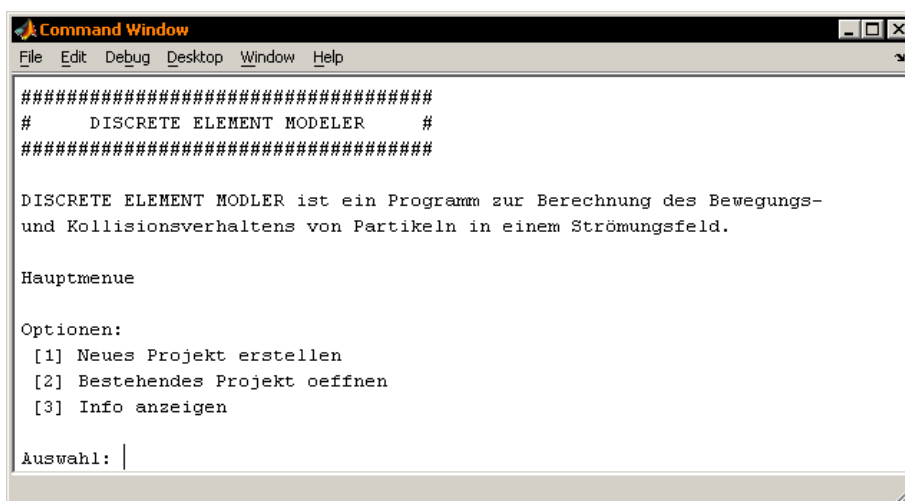
## 4 Realisierung des Programms

### 4.1 Hauptprogramm

Das Hauptprogramm bildet sozusagen den Rahmen des gesamten Programms, alle Unterprogramme und Funktionen werden aus diesem aufgerufen. Aufgrund der Größe des Programms mit ca. 1500 Zeilen Programmcode, kann bei der Beschreibung nicht auf jedes Detail eingegangen werden. Deshalb werden nur die wesentlichen Bausteine des Programms ausführlich beschrieben. Das Hauptprogramm gliedert sich in vier Abschnitte, die jeweils als einzelnes Unterkapitel beschrieben werden.

#### 4.1.1 Eingabe

Der erste Abschnitt des Hauptprogramms beinhaltet die Eingabe von Startwerten, die Definition der Geometrie des Berechnungsgebietes und dessen Randbedingungen, sowie die Einstellung der Berechnungsparameter. Die gesamte Eingabe findet über ein Menü (s. Abb. 4.1) statt, das im Command-Window von Matlab ausgegeben wird. Der Benutzer kann hierbei zwischen drei Optionen, der Erstellung eines neuen Projektes, dem Öffnen eines vorhandenen Projektes und der Anzeige von Programminformationen wählen.



```
Command Window
File Edit Debug Desktop Window Help
#####
#   DISCRETE ELEMENT MODELER   #
#####

DISCRETE ELEMENT MODLER ist ein Programm zur Berechnung des Bewegungs-
und Kollisionsverhaltens von Partikeln in einem Strömungsfeld.

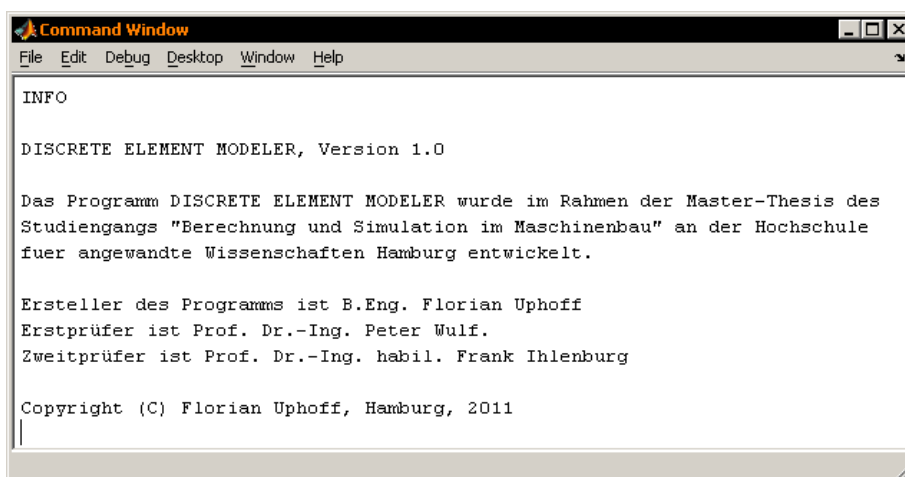
Hauptmenue

Optionen:
[1] Neues Projekt erstellen
[2] Bestehendes Projekt oeffnen
[3] Info anzeigen

Auswahl: |
```

Abb. 4.1: Hauptmenü

Wird Option 1 gewählt, so werden nacheinander die sechs Unterprogramme Write\_Startvalues, Read\_Startvalues, Write\_Boundary, Read\_Boundary, Write\_Conditions und Read\_Conditions gestartet. Die Programme, die mit „Write“ beginnen dienen der Eingabe von Daten, die in den entsprechenden Dateien startvalues.txt, boundary.txt und conditions.txt abgespeichert werden. Die Programme, die mit „Read“ beginnen, dienen dem Einlesen der Daten aus den Dateien startvalues.txt, boundary.txt und conditions.txt in die jeweiligen Variablen des Matlab-Workspace. Eine detaillierte Beschreibung der Unterprogramme findet in den Kapitel 4.2 bis 4.7 statt. Wird die zweite Option gewählt, starten nur die Unterprogramme die mit „Read“ beginnen um die entsprechen Daten in den Matlab-Workspace einzulesen. Bei Wahl von Option 3 werden im Command-Window Informationen zum Programm ausgegeben (siehe Abb. 4.2).



```
Command Window
File Edit Debug Desktop Window Help
INFO
DISCRETE ELEMENT MODELER, Version 1.0
Das Programm DISCRETE ELEMENT MODELER wurde im Rahmen der Master-Thesis des Studiengangs "Berechnung und Simulation im Maschinenbau" an der Hochschule fuer angewandte Wissenschaften Hamburg entwickelt.
Ersteller des Programms ist B.Eng. Florian Uphoff
Erstprüfer ist Prof. Dr.-Ing. Peter Wulf.
Zweitprüfer ist Prof. Dr.-Ing. habil. Frank Ihlenburg
Copyright (C) Florian Uphoff, Hamburg, 2011
```

Abb. 4.2: Ausgabe von Informationen

#### 4.1.2 Definition von Konstanten und Berechnung von Parametern

Der zweite Abschnitt behandelt zunächst die Definition von System-Konstanten. Hierzu zählen das verwendete Koordinatensystem, die Gravitations- und Erdbeschleunigungskonstante, sowie der Widerstandsbeiwert der Partikel und die Berechnungszeit der Simulation. Danach werden globale Variablen für die Geschwindigkeitsintegrale, welche für die Berechnung der Federkräfte zwischen den Partikeln, sowie zwischen Partikel und Wand benötigt werden. Diese Variablen müssen als global definiert werden, da sie ansonsten in der Funktion Calculate\_Forces.m, in der sie später berechnet werden, nicht verändert werden



können. Zusätzlich wird noch die globale Variable *deleted* definiert, in der die Partikel, die das Berechnungsgebiet verlassen haben, abgespeichert werden.

Dann wird das Array  $F_{act}$  definiert. In diesem wird abgespeichert, welche Kräfte und Kontaktmodelle für die Berechnung verwendet werden sollen. Dies sind die Kräfte und Kontaktmodelle, die in der Eingabemaske des Unterprogramms `Write_Conditions.m` abgefragt wurden.

Wenn im Berechnungsgebiet Partikeleinlässe vorhanden sind, werden die Geometrie- und Materialarrays  $R$ ,  $\rho_p$ ,  $E$  und  $nue$  mit den, in der `conditions.txt` Datei abgelegten Eigenschaften, aufgefüllt. Anschließend werden innerhalb einer for-Schleife, die über die Anzahl der Partikeleinlässe iteriert, die Partikelgeschwindigkeiten an den jeweiligen Einlässen kontrolliert. Hierbei wird zunächst überprüft, ob der Geschwindigkeitsvektor in das Berechnungsgebiet zeigt, wenn das Partikel an dem entsprechenden Einlass startet. Danach wird der Betrag der Geschwindigkeit kontrolliert. Ist dieser zu gering, kann es passieren, dass ein Partikel mit einem Partikel, das zuvor aus dem gleichen Einlass gestartet ist in der Startphase kollidiert. Hierbei kann es dann zu fehlerhaften sehr großen Überlappung und daraus resultierenden enormen Beschleunigungen der Partikel kommen. Zum Abschluss werden die Startpositionen der Partikel am jeweiligen Einlass ermittelt. Diese werden mit Hilfe der `rand()` Funktion zufällig entlang des Partikeleinlasses festgelegt.

Darauf folgt die Berechnung der Partikelmasse, des Massenträgheitsmomentes und der Projektionsfläche. Danach findet die Berechnung der effektiven Material- und Geometrieigenschaften  $E_{eff}$ ,  $R_{eff}$ ,  $m_{eff}$  und  $v_{eff}$  nach den Formeln 3.32, 3.33, 3.41 und 3.51 statt. Diese werden jeweils als Matrix abgespeichert, wobei jedes Element einer Partikelpaarung zugewiesen ist. Auf den Elementen der nicht benötigten Hauptdiagonalen der Matrizen werden die effektiven Eigenschaften zwischen dem jeweiligen Partikel und der Wand abgelegt.

Anschließend werden die Federsteifigkeiten und Dämpfungskonstanten nach den Formeln 3.31, 3.44, 3.50 und 3.53 berechnet und nach demselben Schema wie die





Material- und Geometrieigenschaften als Matrizen abgespeichert. Es ist zu beachten, dass es sich hierbei immer nur um den konstanten Term handelt. Der nicht konstante Term, in dem das  $\delta$  enthalten ist, wird erst innerhalb der Funktion Calculate\_Forces.m mit hinzugerechnet. Danach werden die statischen Kräfte, also die Schwerkraft und der statische Auftrieb nach den Formeln 3.60 und 3.63 berechnet.

Als letzter Schritt bevor die Berechnung startet, werden mit fopen() die Dateien geöffnet, in der die Zwischen- und Endergebnisse der Berechnung abgelegt werden sollen. Hierzu wird das Verzeichnis „CSV“ im Projektordner erstellt und in diesem die csv-Dateien (Character Seperated Values) erstellt und geöffnet, in der zu berechnenden Werte abgelegt werden. In Tabelle 4.1 sind die Namen der csv-Dateien und ihre Bedeutung aufgelistet.

Name	Bedeutung
r_x.csv, r_y.csv	Positionsdaten der Partikel in x- und y-Richtung
phi.csv	Daten der Rotationswinkels der Partikel
v_x.csv, v_y.csv	Geschwindigkeitsdaten der Partikel in x- und y-Richtung
omega.csv	Daten der Winkelgeschwindigkeiten der Partikel
a_x.csv, a_y.csv	Beschleunigungsdaten der Partikel in x- und y-Richtung
alpha.csv	Daten der Winkelbeschleunigungen der Partikel

Tab. 4.1: csv-Dateien

### 4.1.3 Berechnung

Zu Beginn des Berechnungsabschnittes werden einige Zähler initialisiert, die für die Berechnung notwendig sind und es wird der Ordner „Postprocessing“ erstellt, in den die für die spätere grafische Ausgabe mit ParaView benötigten Dateien abgelegt werden. Die Berechnung der Partikelbewegungen und -kollisionen findet anschließend in einer for-Schleife statt, die über die Zeitschritte  $dt$  iteriert. Der Inhalt der for-Schleife lässt sich in mehrere Blöcke unterteilen.



Der erste Block ist die Implementierung des Partikeleinlasses. Dieser Block wird über eine if-Bedingung immer dann aktiviert, wenn ein neues Partikel in das Berechnungsgebiet eintreten soll. Dies ist der Fall, wenn die folgende Bedingung erfüllt ist.

$$\frac{t \cdot pps}{1s} > \frac{new\_part}{\text{Anzahl der Einlässe}} + \frac{dt}{1s} \quad (4.1)$$

Die Variable *pps* ist in diesem Fall der Partikelstrom, *t* ist die aktuelle Berechnungszeit und in der Variablen *new\_part* ist die Anzahl der bereits ins Berechnungsgebiet eingebrachten Partikel abgelegt.

Der Partikeleinlass wird als eine for-Schleife realisiert, die über die Anzahl der Partikeleinlässe iteriert und dabei die Partikelzähler *part* (Anzahl der Partikel im Berechnungsgebiet) und *new\_part* um eins erhöht. Außerdem werden noch die Vektoren für Position, Winkel und die entsprechenden Geschwindigkeiten und Beschleunigungen um die Startwerte für neue Partikel ergänzt.

Der zweite Block behandelt die Implementierung des Einlesens der Daten aus der CFD-Simulation. Diese Daten werden dabei gleichmäßig über die Zeit verteilt in das Programm eingelesen, d.h. wenn 10 CFD-Daten vorhanden sind und die Gesamtberechnungszeit 10 s beträgt, wird jede Sekunde eine neue Datei eingelesen. Aufgrund dessen wird der Block über eine if-Bedingung immer dann aktiviert, wenn folgende Bedingung gilt.

$$t > \frac{cfd\_counter \cdot t_{end}}{cfd\_data} \quad (4.2)$$

Die Variable *cfd\_counter* beinhaltet die Anzahl der bereits verwendeten CFD-Dateien, *cfd\_data* beinhaltet die Anzahl aller zur Verfügung stehenden CFD-Dateien und  $t_{end}$  ist das Ende der Berechnungszeit.

Ist die Bedingung erfüllt, werden mit dem Unterprogramm *Import\_CFD.m* die Daten einer neuen CFD-Datei in die Berechnung geladen. Die in *Import\_CFD.m* bestimmte



Variable *PointVelXYZ*, die die Fluidgeschwindigkeit in den Knoten des CFD-Netzes enthält, wird an die Variablen *vx\_f* und *vy\_f* übergeben. Diese werden für die in Kapitel 4.9.3 beschriebene Berechnung der Widerstandskräfte vom Fluid auf das Partikel benötigt.

Der dritte Block ist die Berechnung der Partikelbewegung und -kollisionen, der nur dann aufgerufen wird, wenn Partikel im Berechnungsgebiet vorhanden sind. Ist dies der Fall so werden beim ersten Iterationsschritt den Werten für die Position, den Rotationswinkel und deren Geschwindigkeiten die Startwerte aus der *startvalues.txt* Datei zugewiesen. Anschließend wird mit der Funktion *Calculate\_Forces.m* die, aufgrund der Startwerte, entstehenden Kräfte und Momente berechnet. Daraus werden dann über die Bewegungsgleichung die Beschleunigung und Winkelbeschleunigung ermittelt. Ab dem zweiten Iterationsschritt werden die Position und der Winkel als Praediktorwerte über die dreigliedrige MacLaurin-Reihe (s. Formel 3.16 und 3.18) ermittelt. Die Geschwindigkeit und die Winkelgeschwindigkeit werden als Praediktorwerte über die zweigliedrige MacLaurin-Reihe (s. Formel 3.17 und 3.19) bestimmt. Die Praediktorbeschleunigungen (Translation und Rotation) werden auf den Wert der vorherigen Iteration gesetzt.

Anschließend werden die Korrektorbeschleunigungen über die Kräfte und Momente ermittelt. Ist die Variable *max\_pk\_iter* auf Null gesetzt, so wird anschließend der nächste Iterationsschritt berechnet. Dies entspricht dann dem modifizierten Euler-Verfahren. Ist *max\_pk\_iter* größer als Null, so beginnt jetzt die Praediktor-Korrektor Iteration. Dazu wird als erstes die Differenz der Korrektor- und Praediktorbeschleunigungen ermittelt. Ist eine der Differenzen größer als das Abbruchkriterium  $\epsilon$ , so werden die Praediktorwerte nach den Gleichungen 3.25, 3.26 und 3.27 korrigiert. Aus diesen korrigierten Werten werden dann neue Korrektorbeschleunigungen berechnet und die Differenz zur korrigierten Praediktorbeschleunigung berechnet. Dies wird solange wiederholt, bis das Abbruchkriterium erfüllt ist oder die maximale Anzahl an Praediktor-Korrektor Iterationen erreicht ist.



Der vierte Block sorgt dafür, dass die berechneten Daten für Position, Winkel und deren entsprechende Geschwindigkeiten und Beschleunigungen in den bereits geöffneten csv-Dateien abgespeichert werden. Anschließend werden die Daten mit Hilfe des Unterprogramms `Convert_ml2vtk.m` in das vtk-Format konvertiert. Dies ist nötig, damit die Berechnungsergebnisse später im Visualisierungsprogramm ParaView dargestellt werden können. Mit Hilfe der Variablen `vtk_save`, die über das Hauptmenü definiert wird, lässt sich einstellen jede wievielte Berechnung konvertiert werden soll. Zum Schluss werden die aktuellen Werte für Position, Rotationswinkel, sowie Geschwindigkeit und Beschleunigung für Translation und Rotation den Variablen für die Startwerte des nächsten Zeitschritts zugewiesen.

Der fünfte und letzte Block beschäftigt sich mit der grafischen Ausgabe der Partikelbewegung während der Berechnung. Im Falle von aufwendigen Berechnungen sollte diese ausgestellt werden, da sie viel Rechenzeit benötigt. Das Partikel wird als gelber Kreis dargestellt. Um diesen gelben Kreis rotiert ein roter kleinerer Kreis, der die Drehung des Partikels visualisieren soll. Außerdem werden noch die Ränder je nach Randbedingung verschiedenfarbig dargestellt. Wände werden grün, Auslässe blau und Einlässe rot dargestellt. Außerdem wird die Strömung im Berechnungsgebiet mit Pfeilen, die den Geschwindigkeitsvektoren entsprechen, angedeutet. Im Command-Window wird währenddessen in definierten Abständen die aktuelle Berechnungszeit ausgegeben.

### 4.1.4 Ausgabe von Diagrammen

Der letzte Teil des Hauptprogramms behandelt die grafische Ausgabe der Berechnungsergebnisse. Hierzu werden zunächst die csv-Dateien, in denen die Berechnungsergebnisse abgelegt sind mit Hilfe der Funktion `importdata()` eingelesen und unter ihren jeweiligen Variablen im Matlab-Workspace abgelegt. Diese Variablen liegen bei mehreren Partikeln als Matrix und bei nur einem Partikel als Zeilenvektor vor.

Die grafische Ausgabe findet anschließend mit der `plot()` Funktion von Matlab statt. Es werden insgesamt 9 Daten über die Zeit aufgetragen (s. Abb. x.x). Dies sind die

Partikelpositionen, -geschwindigkeiten und -beschleunigungen in x- und y-Richtung, sowie der Rotationswinkel, die Winkelgeschwindigkeit und die Winkelbeschleunigung.

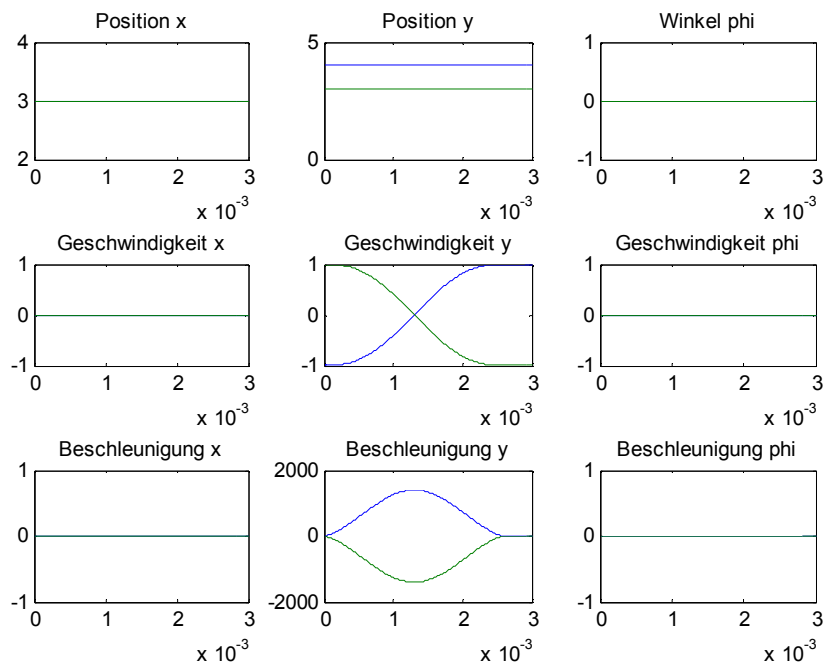


Abb. 4.3: Ausgabe der Berechnungsergebnisse

## 4.2 Unterprogramm Write\_Startvalues.m

Das Unterprogramm Write\_Startvalues.m dient dazu, eine Datei mit Startwerten für die Partikel, die sich zu Berechnungsbeginn bereits im Berechnungsgebiet befinden sollen, zu erstellen. Diese Datei wird dann im Projektordner unter dem Namen startvalue.txt abgespeichert.

Das Programm beginnt mit einer Überprüfung, ob bereits eine Datei mit Startwerten im Projektordner vorhanden ist. Diese wird mit Hilfe der exist() Funktion von Matlab durchgeführt. Ist bereits eine Datei vorhanden, so wird der Benutzer gefragt, ob er diese überschreiben will. Wenn die Datei nicht überschrieben werden soll, endet das Programm an dieser Stelle.

Wenn die Datei überschrieben werden soll oder noch nicht existiert, wird sie im Projektordner neu erstellt und im Schreibmodus geöffnet. Anschließend wird eine Informationszeile „Datei mit Startwerten der Partikel“ in die Datei geschrieben und



der Benutzer wird gefragt, wie viele Partikel er eingeben möchte. Die Anzahl wird in der Variablen *particle\_counter* abgespeichert. Wenn der Benutzer mehr als ein Partikel eingeben möchte, wird er gefragt, ob alle Partikel dieselben Materialeigenschaften haben sollen. Daraufhin können innerhalb einer for-Schleife die Partikeldaten, welche in Tab. 4.2 aufgelistet sind nacheinander eingegeben werden. Will der Benutzer allen Partikeln dieselben Materialeigenschaften geben, so werden diese durch eine if-Bedingung nur beim ersten Schleifendurchlauf abgefragt und anschließend für alle weiteren Partikel übernommen. Am Ende eines Schleifendurchgangs werden die eingegeben Daten von einem Partikel in dem Array *startvalues* abgespeichert. Nach Beendigung der Schleife werden die Daten aus dem Array *startvalue* mit Hilfe des Befehls `fprintf()` in die Datei geschrieben. Hierbei werden zwischen den unterschiedlichen Partikeldaten Kommentare eingefügt, damit der Benutzer diese später besser lesen kann. Außerdem wird vor jede Zeile mit Partikeldaten ein dreistelliges Schlüsselwort geschrieben, anhand dessen die Datei wieder ausgelesen werden kann. Danach wird die Datei *startvalue.txt* wieder geschlossen und ein Kommentar, ob die Erstellung der Datei erfolgreich war, wird im Command-Window ausgegeben.

Schlüsselwort	Variable	Bedeutung	Einheit
#RX	<i>r_x</i>	Startposition x-Koordinate des Partikels	m
#RY	<i>r_y</i>	Startposition y-Koordinate des Partikels	m
#VX	<i>v_x</i>	Partikelstartgeschwindigkeit in x-Richtung	$\text{m}\cdot\text{s}^{-1}$
#VY	<i>v_y</i>	Partikelstartgeschwindigkeit in y-Richtung	$\text{m}\cdot\text{s}^{-1}$
#PH	<i>phi</i>	Partikelstartrotationswinkel	rad
#OM	<i>omega</i>	Partikelstartwinkelgeschwindigkeit	$\text{rad}\cdot\text{s}^{-1}$
#RA	<i>R</i>	Partikelradius	m
#RH	<i>rho</i>	Partikeldichte	$\text{kg}\cdot\text{m}^{-3}$
#EM	<i>E</i>	Elastizitätsmodul der Partikel	$\text{N}\cdot\text{m}^{-2}$
#NU	<i>nue</i>	Poisson-Zahl der Partikel	-
//(Leerzeichen)	-	Kommentar in der Datei	-

Tab. 4.2: Daten der *startvalue.txt* Datei



### 4.3 Unterprogramm Read\_Startvalues.m

Das Unterprogramm Read\_Startvalues.m dient dazu, die Startwerte der Partikel aus der startvalue.txt Datei in den Matlab-Workspace einzulesen. Hierzu wird zunächst mit Hilfe der exist() Funktion überprüft, ob die entsprechende startvalues.txt Datei überhaupt im Projektordner vorhanden ist. Ist dies nicht der Fall, so wird eine Fehlermeldung im Command-Window ausgegeben.

Wenn die startvalues.txt Datei im Projektordner vorhanden ist, wird diese im Lesemodus geöffnet. Anschließend wird innerhalb einer while-Schleife in jedem Schleifendurchgang eine Zeile der Datei in die Variable *dataline* geschrieben. Die ersten drei Zeichen in der Variablen *dataline* sind das Schlüsselwort. Mit Hilfe einer switch-case-Unterscheidung anhand dieser Schlüsselworte, werden die Daten in den entsprechenden Variablen (s. Tab. 4.4) im Matlab-Workspace abgelegt. Die Kommentare in der startvalue.txt Datei werden einfach übersprungen. Die while-Schleife endet, sobald das Ende der Datei erreicht ist. Anschließend wird die startvalue.txt Datei wieder geschlossen und ein Kommentar, ob das Schließen der Datei erfolgreich war, wird im Command-Window ausgegeben.

### 4.4 Unterprogramm Write\_Boundary.m

Das Unterprogramm Write\_Boundary.m dient dazu, die Geometrie des Berechnungsgebiets und dessen Randbedingungen in der Datei boundary.txt im Projektordner abzuspeichern. Das Programm beginnt mit einer Überprüfung, ob bereits eine Datei mit Randbedingungen im Projektordner vorhanden ist. Diese wird wieder mit Hilfe der exist() Funktion durchgeführt. Ist bereits eine Datei vorhanden, so wird der Benutzer gefragt, ob er diese überschreiben will. Wenn die Datei nicht überschrieben werden soll, endet das Programm an dieser Stelle.

Ziffer	Randbedingung
1	Feste Wand
2	Partikelauslass
3	Partikeleinlass

Tab. 4.3: Randbedingungen



Wenn die Datei überschrieben werden soll oder noch nicht existiert, wird sie im Projektordner neu erstellt und im Schreibmodus geöffnet. Anschließend wird eine Informationszeile „Datei mit Geometrie und Randbedingungen“ in die Datei geschrieben. Innerhalb einer while-Schleife kann der Benutzer die x- und y-Koordinaten der Punkte, sowie deren Randbedingung eingeben. Die Randbedingung wird dabei über eine Ziffer (s. Tab. 4.3) eingegeben. Anschließend werden die eingegebenen Daten in dem Array *points* gespeichert. Wenn der Benutzer keine Punkte mehr eingeben will endet die Schleife. Die Daten aus dem Array *points* werden dann anhand von Schlüsselworten (s. Tab. 4.4) in der Datei *boundary.txt* abgelegt.

Schlüsselwort	Variable	Bedeutung	Einheit
#PX	<i>points(:,1)</i>	x-Koordinate des Punktes	m
#PY	<i>points(:,2)</i>	y-Koordinate des Punktes	m
#PZ	<i>points(:,3)</i>	z-Koordinate des Punktes	m
#PB	<i>points(:,4)</i>	Randbedingung des Punktes	-
#E1	<i>edges(:,1)</i>	Startpunkt der Kante	-
#E2	<i>edges(:,2)</i>	Endpunkt der Kante	-
#EB	<i>edges(:,3)</i>	Randbedingung der Kante	-
#WE	<i>E_wall</i>	Elastizitätsmodul der Wand	$\text{N}\cdot\text{m}^{-2}$
#WN	<i>nue_wall</i>	Poisson-Zahl der Wand	-
#WM	<i>mue_wall</i>	Reibungskoeffizient Wand - Partikel	-
#VX	<i>new_vx</i>	Partikelstartgeschwindigkeit in x-Richtung	$\text{m}\cdot\text{s}^{-1}$
#VY	<i>new_vy</i>	Partikelstartgeschwindigkeit in y-Richtung	$\text{m}\cdot\text{s}^{-1}$
#PH	<i>new_phi</i>	Partikelstartrotationswinkel	rad
#OM	<i>new_omega</i>	Partikelstartwinkelgeschwindigkeit	$\text{rad}\cdot\text{s}^{-1}$
#RA	<i>new_R</i>	Partikelradius	m
#RH	<i>new_rho</i>	Partikeldichte	$\text{kg}\cdot\text{m}^{-3}$
#EM	<i>new_E</i>	Elastizitätsmodul der Partikel	$\text{N}\cdot\text{m}^{-2}$
#NU	<i>new_nue</i>	Poisson-Zahl der Partikel	-
//(Leerzeichen)	-	Kommentar	-

Tab. 4.4: Daten der *boundary.txt* Datei





In der folgenden while-Schleife kann der Benutzer die Kanten des Berechnungsgebietes definieren. Hierbei muss jeweils ein Start- und ein Endpunkt, sowie eine Randbedingung (s. Tab. 4.3) eingegeben werden. Die Anzahl der Kanten mit Randbedingung feste Wand und Randbedingung Partikeleinlass werden dabei innerhalb der Schleife gezählt. Die eingegebenen Daten werden in dem Array *edges* gespeichert. Es ist darauf zu achten, dass die Kanten für äußere Grenzen im Uhrzeigersinn und Kanten für innere Grenzen entgegen dem Uhrzeigersinn definiert werden, da ansonsten die Normalenvektoren der Kanten falsch berechnet werden. Wenn der Benutzer alle Kanten definiert hat, wird die Schleife beendet und die Daten aus dem Array *edges* werden anhand von Schlüsselworten (s. Tab. 4.4) in der Datei *boundary.txt* abgelegt.

Wenn der Zähler für die festen Wände größer als Null ist, muss der Benutzer den Elastizitätsmodul und die Poisson-Zahl der Wand, sowie den Reibungskoeffizienten zwischen Partikel und Wand angeben. Wenn der Zähler für die Partikeleinlässe größer Null ist, muss der Benutzer die Startwerte (s. Tab. 4.4, Zeile 11-18) für die Partikel, die aus dem Einlass kommen sollen angeben. Um die Eingabe bei mehreren Einlässen zu erleichtern, wird der Benutzer gefragt, ob er für alle Einlässe dieselben Einstellungen benutzen möchte. Die eingegebenen Daten werden anschließend in der *boundary.txt* Datei abgespeichert. Beim Abspeichern werden, wie bereits beim Unterprogramm *Write\_Startvalues.m*, zwischen den einzelnen Einträgen Kommentare geschrieben, um dem Benutzer das Lesen der Datei zu erleichtern. Anschließend wird die *boundary.txt* Datei wieder geschlossen und ein Kommentar, ob das Schließen der Datei erfolgreich war, wird im Command-Window ausgegeben.

Um die Eingabe des Berechnungsgebietes zu kontrollieren, kann sich der Benutzer dieses grafisch ausgeben lassen. Die Punkte werden dabei als Kreise und die Kanten als Linien dargestellt. Über einen Farbcode, können die eingestellten Randbedingungen überprüft werden. Partikeleinlässe werden rot, Partikelauslässe werden blau und feste Wände werden grün dargestellt (s. Abb. 4.4).

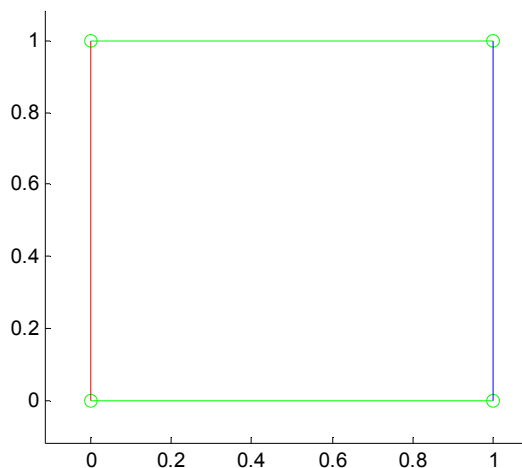


Abb. 4.4: Berechnungsgebiet mit Randbedingungen

#### 4.5 Unterprogramm Read\_Boundary.m

Das Unterprogramm `Read_Boundary.m` dient dazu, die Geometrie des Berechnungsgebietes und dessen Randbedingungen aus der `boundary.txt` Datei in den Matlab-Workspace einzulesen. Das Programm ist ähnlich aufgebaut wie das Programm `Read_Startvalues.m`. Auch hier wird zunächst überprüft, ob die auszulesende Datei überhaupt existiert. Anschließend werden die Daten anhand der Schlüsselworte aus Tab. 4.4 ausgelesen und in den Matlab-Workspace übertragen.

#### 4.6 Unterprogramm Write\_Conditions.m

Das Unterprogramm `Write_Conditions.m` dient dazu, die Berechnungsparameter, die aktiven Kräfte und Momente, sowie Einstellungen zum Abspeichern der Daten in der Datei `conditions.txt` im Projektordner abzuspeichern.

Zu Beginn überprüft das Programm mit Hilfe der `exist()` Funktion, ob bereits eine Datei mit dem Namen `conditions.txt` im Projektordner existiert. Ist eine Datei vorhanden, so wird der Benutzer gefragt, ob er diese überschreiben will. Wenn der Benutzer die Datei nicht überschreiben möchte, endet das Programm an dieser Stelle.



Wenn die Datei überschrieben werden soll oder noch nicht existiert, wird diese neu erstellt und im Schreibmodus geöffnet. Anschließend wird eine Informationszeile „Datei mit den Berechnungsparametern“ in die Datei geschrieben. Dann wird der Benutzer aufgefordert, die Berechnungsparameter einzugeben. Diese werden dann unter ihrem jeweiligen Schlüsselwort (s. Tab. 4.5, Zeile 1-11) in der Datei abgespeichert.

Als nächstes muss der Benutzer eingeben, welche Kräfte bzw. Kontaktmodelle für die Berechnung aktiviert werden sollen. Zur Auswahl stehen hierbei die Schwerkraft, die statische Auftriebskraft, die Fluidwiderstandskraft und die Gravitationskraft. Außerdem kann der Benutzer entscheiden, ob Partikel-Partikel Kontakte und Partikel-Wand Kontakte aktiviert werden sollen. Kräfte bzw. Kontaktmodelle, die nicht unbedingt benötigt werden, sollten nicht aktiviert werden, um bei der Berechnung Rechenzeit zu sparen. Bei Aktivierung wird unter dem jeweiligen Schlüsselwort (s. Tab. 4.5, Zeile 12-17) eine 1 und ansonsten eine 0 gespeichert.

Letzter Teil der Eingabe sind die Werte für das Abspeichern von Daten und zur grafischen Ausgaben. Als erstes wird der Benutzer gefragt, die Daten jedes wievielten Berechnungsschrittes er abspeichern möchte. Die Daten werden hierbei im csv-Format abgespeichert. Wenn später nur eine grafische Ausgabe erfolgen soll und die Daten gar nicht benötigt werden, kann hier eine Null eingegeben werden. Als nächstes wird der Benutzer gefragt, die Daten jedes wievielten Berechnungsschrittes er später grafisch mit dem Programm ParaView ausgeben möchte. Diese Daten werden mit Hilfe des Unterprogramms ml2vtk.m in das für ParaView lesbare vtk-Format konvertiert. Als letztes kann der Benutzer angeben jeden wievielten Berechnungsschritt er während der Berechnung grafisch in Matlab ausgegeben haben möchte. Hierbei ist zu beachten, dass die grafische Ausgabe während der Berechnung die Rechenzeit signifikant erhöht. Anschließend wird die conditons.txt Datei wieder geschlossen und ein Kommentar, ob das Schließen der Datei erfolgreich war, wird im Command-Window ausgegeben.



Schlüsselwort	Variable	Bedeutung	Einheit
#PS	<i>pps</i>	Anzahl der Partikel pro Sekunde am Einlass	s <sup>-1</sup>
#DT	<i>dt</i>	Zeitschrittweite	s
#TE	<i>t_end</i>	Endzeit der Berechnung	s
#EP	<i>epsilon</i>	Abbruchkriterium der PK-Schleife	-
#PK	<i>max_pk_iter</i>	Anzahl der max. Iteration der PK-Schleife	-
#CO	<i>corrector</i>	Korrekturfaktor PK-Schleife	-
#Dn	<i>Dn</i>	Dämpfungsmaß Normalenrichtung	-
#Dt	<i>Dt</i>	Dämpfungsmaß Tangentialrichtung	-
#Dr	<i>Dr</i>	Dämpfungsmaß Drehrichtung	-
#RF	<i>rho_f</i>	Fluiddichte	kg·m <sup>-3</sup>
#MU	<i>mue</i>	Reibungskoeffizient Partikel - Partikel	-
#Fg	<i>Fg_act</i>	Schwerkraft aktivieren	-
#FA	<i>FA_act</i>	Statische Auftriebskraft aktivieren	-
#Fw	<i>Fw_act</i>	Fluidwiderstandskraft aktivieren	-
#FG	<i>FG_act</i>	Gravitationskraft aktivieren	-
#FP	<i>FP_act</i>	Partikel-Partikel Kontakt aktivieren	-
#FB	<i>FB_act</i>	Partikel-Wand Kontakt aktivieren	-
#SC	<i>save_csv</i>	Speicherung jedes wievielten Datensatzes	-
#SV	<i>save_vtk</i>	Speicherung jeder wievielten Grafik	-
#PI	<i>picture</i>	Ausgabe jeder wievielten Grafik (Matlab)	-
//(Leerzeichen)	-	Kommentar	-

Tab. 4.5: Daten der conditions.txt Datei

## 4.7 Unterprogramm Read\_Conditions.m

Das Unterprogramm Read\_Conditions.m dient dazu, die Berechnungsparameter, die aktiven Kräfte und Momente, sowie Einstellungen zum Abspeichern der Daten aus der conditions.txt Datei in den Matlab-Workspace einzulesen. Das Programm ist auch wieder ähnlich aufgebaut wie das Programm Read\_Startvalues.m. Auch hier wird zunächst überprüft, ob die auszulesende Datei überhaupt existiert. Anschließend werden die Daten anhand der Schlüsselworte aus Tabelle 4.5 ausgelesen und in den Matlab-Workspace übertragen.



## 4.8 Unterprogramm Import\_CFD

Das Unterprogramm Import CFD dient dazu, die Daten aus der Strömungssimulation mittels CFD in den Matlab-Workspace einzulesen. Die CFD Berechnung findet dabei mit dem Programm OpenFOAM statt und die berechneten Daten liegen danach im ASCII-Format als vtk-Datei vor. Um die Daten einzulesen, muss zunächst verstanden werden, wie diese vtk-Datei aufgebaut ist. In den Kommentaren in Tab. 4.6 ist beschrieben, was jede einzelne Zeile der vtk-Datei bedeutet.

# vtk DataFile Version 2.0	% Dateityp und -version
noname	% Name der Berechnung
ASCII	% Dateiformat
DATASET UNSTRUCTURED_GRID	% Typ des Datensatzes unstrukturiertes Netz
POINTS 882 float	% 882 Knoten im float Format
0 0 0 ...	% Knotenkoordinaten in 3D
CELLS 400 3600	% 400 Zellen mit insgesamt 3600 Einträgen
8 1 442 463 22 0 441 462 21 ...	% Anzahl der Knoten und die IDs der jeweiligen Knoten
CELL_TYPES 400	% Zelltypen für die 400 Zellen
12 12 12 12 12 12 12 12 12 12	% Zelltyp 12 bedeutet Hexaeder
...	
CELL_DATA 400	% Zellendaten für 400 Zellen
FIELD attributes 3	% 3 Feldattribute folgen
cellID 1 400 int	% 400 integer IDs für die 400 Zellen
0 1 2 3 4 5 6 7 8 9 10	% IDs der Zellen
...	
p 1 400 float	% Druck in den 400 Zellen
0 ...	
U 3 400 float	% Geschwindigkeitsvektoren in den 400 Zellen
0 0 0 ...	
POINT_DATA 882	% Interpolierte Daten in den 882 Knoten
FIELD attributes 2	% 2 Feldattribute folgen
p 1 882 float	% Druck an den 882 Knoten
0 0 0 0 0 0 0 0 0 0 0	
...	
U 3 882 float	% Geschwindigkeitsvektoren an den 882 Knoten
0 1.75 0 ...	

Tab. 4.6: Aufbau der vtk-Datei

Das Programm beginnt mit dem Laden der globalen Variablen PointXYZ, in der die Punktkoordinaten abgelegt werden, und PointVelXYZ, in der die Geschwindigkeitsvektoren an den Punkten abgelegt werden. Anschließend werden die Variablen *nodes* und *pointvelocity* initialisiert und die vtk-Datei wird im Lesemodus geöffnet. Aufgrund des Aufbaus der vtk-Datei werden die ersten fünf



Zeilen, in denen Dateiinformatoren enthalten sind, übersprungen. Anschließend werden innerhalb einer while-Schleife die Koordinaten der Knoten des Berechnungsnetzes in die Variable *nodes* geschrieben. Diese while Schleife bricht ab, sobald alle Punktkoordinaten eingelesen sind, bzw. wenn in der eingelesenen Zeile das „C“ von „CELLS“ (siehe Tab. 4.6, Zeile 8) abgelegt ist. Das Einlesen der Geschwindigkeitsvektoren an den Knoten in die Variable *pointvelocity* erfolgt in gleicher Weise. Allerdings wird hier die while-Schleife abgebrochen, sobald das Ende der Datei erreicht ist. Anschließend wird die Datei wieder geschlossen.

Die Daten in *nodes* liegen somit in einem eindimensionalen Array vor. Für die weitere Verwendung werden sie als Matrix unter *PointXYZ* abgespeichert. Dasselbe gilt für die Daten in der Variablen *pointvelocity*. Diese werden in der Matrix *PointVelXYZ* abgespeichert.

## 4.9 Funktion Calculate\_Forces.m

Die Funktion Calculate\_Forces.m dient der Berechnung der zeitlich veränderlichen Kräfte und Momente auf die Partikel. Außerdem werden in ihr die Partikel detektiert, die das Berechnungsgebiet durch den Partikelaustritt verlassen haben.

Als erster Schritt werden die globalen Variablen für die Geschwindigkeitsintegrale (Integral der relativen Tangentialgeschwindigkeit und Integral der relativen Rollgeschwindigkeit), sowie die Variable *deleted* geladen, in welcher die Indizes der partikel abgespeichert werden, die das Berechnungsgebiet verlassen haben. Anschließend beginnt die Berechnung der Kräfte und Momente.

### 4.9.1 Partikel-Partikel Interaktion

Um die Kräfte und Momente zu berechnen, die zwischen zwei Partikeln wirken, muss zunächst auf effektive Weise der Abstand, bzw. die Überlappung zwischen allen Partikeln bestimmt werden. Dieses wird mit Hilfe von Matrixoperationen realisiert, da diese in Matlab besonders effektiv rechnen und wesentlich schneller als for-Schleifen sind.

Hierzu wird zunächst der Abstand aller Partikelschwerpunkte in x-Richtung bestimmt. Dabei werden die Positionsvektoren der Partikelschwerpunkte zunächst über die dyadische Multiplikation mit einem Einservektor<sup>2</sup> zu einer Matrix aufgebläht.

$$\vec{r}_x \otimes \vec{e} = \begin{bmatrix} r_{x,1} \\ \vdots \\ r_{x,n} \end{bmatrix} \otimes \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} r_{x,1} & \cdots & r_{x,1} \\ \vdots & \ddots & \vdots \\ r_{x,n} & \cdots & r_{x,n} \end{bmatrix} \quad (4.3)$$

Dann wird die Transponierte dieses dyadischen Produktes von der zuvor berechneten Matrix abgezogen und es entsteht:

$$\vec{r}_x \otimes \vec{e} - \vec{e} \otimes \vec{r}_x = \begin{bmatrix} r_{x,1} - r_{x,1} & \cdots & r_{x,1} - r_{x,n} \\ \vdots & \ddots & \vdots \\ r_{x,n} - r_{x,1} & \cdots & r_{x,n} - r_{x,n} \end{bmatrix} = \begin{bmatrix} 0 & \cdots & r_{x,1} - r_{x,n} \\ \vdots & \ddots & \vdots \\ r_{x,n} - r_{x,1} & \cdots & 0 \end{bmatrix} \quad (4.4)$$

Dieselbe Rechnung wird für die y-Richtung durchgeführt, so dass sich in den beiden entstehenden Matrizen die Abstände der Schwerpunkte aller Partikel in x- und y-Richtung befinden. Nun wird der Satz des Pythagoras auf die beiden Matrizen angewendet, um den Abstand zu berechnen. Hierzu werden beide Matrizen zunächst elementweise quadriert, dann addiert und anschließend wird elementweise die Wurzel gezogen. Somit erhält man eine Matrix in der sich die Schwerpunktabstände der Partikel befinden. Um die Abstände der Partikeloberflächen zu bestimmen müssen jetzt noch die Summe der beiden Partikelradien abgezogen werden. Hier wird der gleiche Trick mit dem dyadischen Produkt angewendet.

$$\vec{R} \otimes \vec{e} + \vec{e} \otimes \vec{R} = \begin{bmatrix} R_1 + R_1 & \cdots & R_1 + R_n \\ \vdots & \ddots & \vdots \\ R_n + R_1 & \cdots & R_n + R_n \end{bmatrix} = \left[ \begin{bmatrix} 2R_1 & \cdots & R_1 + R_n \\ \vdots & \ddots & \vdots \\ R_n + R_1 & \cdots & 2R_n \end{bmatrix} \right] \quad (4.5)$$

Um die Elemente auf der Hauptdiagonalen auf Null zu bekommen, wird eine Matrix mit den doppelten Radien auf der Hauptdiagonalen abgezogen. So erhält man letztlich die in Formel 4.6 beschriebene Form.

$$\vec{\delta} = \underbrace{\left[ \left[ \vec{r}_x \otimes \vec{e} - \vec{e} \otimes \vec{r}_x \right]^* + \left[ \vec{r}_y \otimes \vec{e} - \vec{e} \otimes \vec{r}_y \right]^* \right]^{*0.5}}_{\text{Abstand der Partikel-Schwerpunkte}} - \underbrace{\left[ \vec{R} \otimes \vec{e} + \vec{e} \otimes \vec{R} - \text{diag}(2\vec{R}) \right]}_{\text{Summe der Partikel-Radien}} \quad (4.6)$$

<sup>2</sup> Vektor mit einer Länge gleich der Partikelanzahl, der mit 1 gefüllt ist.

Nach der Berechnung ergibt sich die folgende Matrix mit Nullen auf der Hauptdiagonalen und den Abständen zwischen den Partikeln auf den übrigen Feldern.

$$\vec{\delta} = \begin{bmatrix} 0 & \delta_{12} & \dots & \delta_{1n} \\ \delta_{21} & 0 & \dots & \delta_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{n1} & \delta_{n2} & \dots & 0 \end{bmatrix} \quad (4.7)$$

Allerdings werden für die weitere Berechnung nur die negativen Elemente dieser Matrix benötigt. Mit der Formel 4.8 lassen sich alle nicht negativen Elemente eliminieren und durch Nullen ersetzen.

$$\vec{\delta} = 0.5(\vec{\delta} - \text{sign}(\vec{\delta}) * \vec{\delta}) \quad (4.8)$$

Die nun entstandene Matrix ist dünn besetzt, enthält aber noch die Einträge  $\delta_{ij}$  und  $\delta_{ji}$ , die identisch sind. Mit Hilfe der Funktion `triu()`, lassen sich die doppelten Elemente eliminieren, indem aus der Matrix eine obere Dreiecksmatrix gemacht wird. Mit Hilfe der Funktion `find()` werden nun die Indizes aller nicht Null Elemente herausgesucht.

Die Berechnung der Kontaktkräfte und -momente erfolgt innerhalb einer `for`-Schleife, die über die mit `find()` ermittelten Indizes iteriert. Um abzusichern, dass zu große Überlappungen, die am Partikel-Einlass entstehen könnten, nicht gerechnet werden, wird vor dem Start kontrolliert, dass die Überlappung maximal 1% des Partikelradius beträgt.

Innerhalb der `for`-Schleife wird für die jeweilige Partikelpaarung der Normalen- und Tangentialvektor berechnet und anschließend werden nach den Formeln 3.34 und 3.46 die Normalkräfte bestimmt. Die Tangentialkräfte werden über die Formeln 3.47 und 3.54 zunächst berechnet und anschließend wird überprüft, ob die Tangentialkraft kleiner gleich der Reibungskoeffizient multipliziert mit der Normalkraft ist. Ist dies nicht der Fall, dann wird die Tangentialkraft über die folgende Formel neu berechnet.

$$\vec{F}_t = \text{sign}(\vec{F}_t) * \left| \mu \cdot (\vec{z}_{dir} \times \vec{F}_n) \right| \quad (4.9)$$

Der Betrag wird in diesem Fall elementweise auf den Vektor angewendet. Alle Kräfte werden vektoriell bestimmt anschließend summiert und in der  $3 \times n$  Matrix `F_Col`

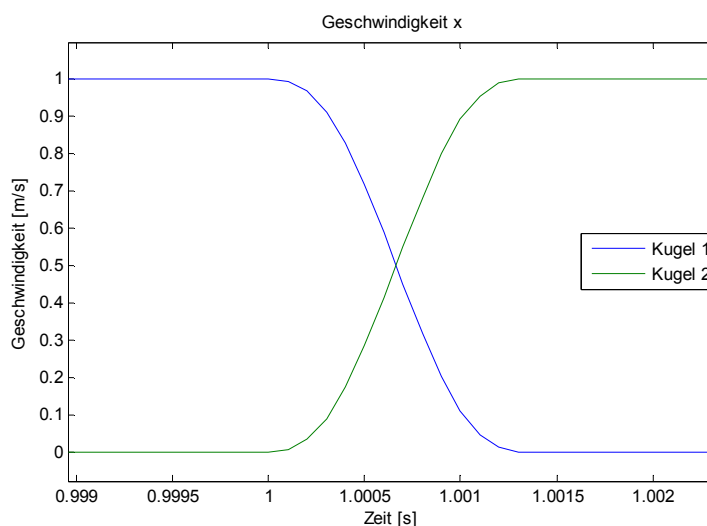


unter den jeweiligen Partikelindizes abgespeichert. Anschließend werden über die Formeln 3.46 und 3.59 die Momente ausgerechnet, die aufgrund des Feder-Dämpfer-Systems in Drehrichtung auf die Partikel wirken. Außerdem werden noch die Momente berechnet, die über die Tangentialkraft auf die Partikel wirken.

$$M_i = (\mathbf{R}_i \cdot \vec{\mathbf{n}}) \times \vec{\mathbf{F}}_t \quad (4.10)$$

Auch die Momente werden dann aufsummiert und in der  $3 \times n$  Matrix  $M\_Col$  unter den jeweiligen Partikelindizes abgespeichert. Anschließend werden einige Testrechnungen zur Validierung des Programms durchgeführt. Hierbei wird an bekannten Konstellationen kontrollieren, ob der Algorithmus nachvollziehbare Ergebnisse liefert.

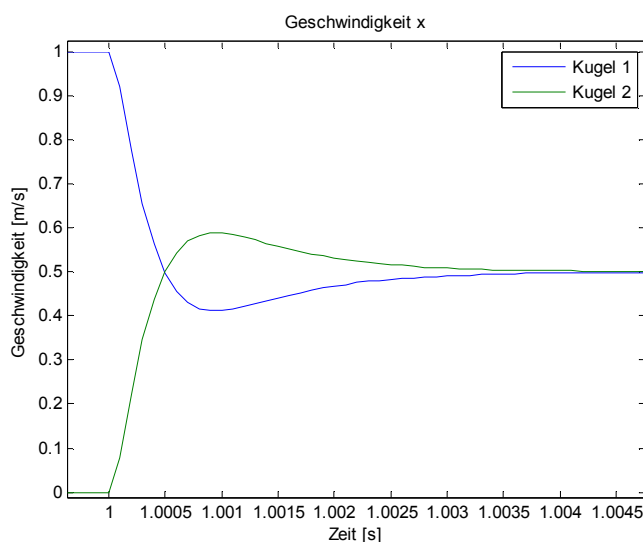
Die Validierung beginnt bei der Federkraft in Normalenrichtung. Hierbei wird der Kontakt zweier identischer Kugeln berechnet. Kugel 1 wird mit einer Geschwindigkeit von 1 m/s zentrisch auf die ruhende Kugel 2 geschossen. Beim Kontakt müsste nun Kugel 1 ihren gesamten Impuls auf Kugel 2 übertragen, so dass Kugel 1 stehen bleibt und Kugel 2 mit der Geschwindigkeit 1 m/s wegfliegt.



**Abb. 4.5: Zentrischer Kontakt zweier Kugeln**

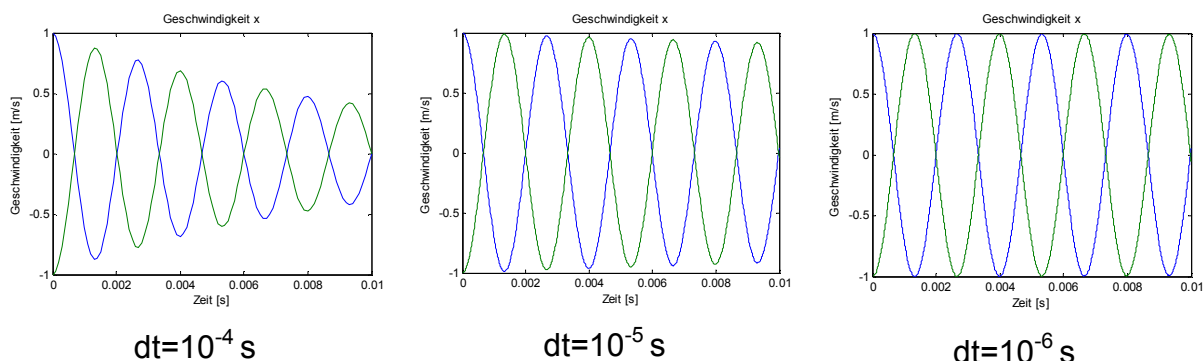
In Abbildung 4.5 ist der Geschwindigkeitsverlauf der beiden Kugeln zu sehen. Hierbei kann gut erkannt werden, dass der Impuls während des Kontaktes von Kugel 1 auf Kugel 2 übertragen wurde.

Als nächstes wird die Dämpfungskraft in Normalenrichtung untersucht. Dazu wird der vorherige Versuch wiederholt, nur dass diesmal das Dämpfungsmaß auf  $D_n=1$  gesetzt ist. Hierbei tritt dann ein rein-plastischer Stoß auf und beide Kugeln müssten nach dem Kontakt mit 0,5 m/s weiterfliegen. In Abbildung 4.6 ist das Ergebnis zu sehen. Auch hier stimmt die Vorhersage mit dem Berechnungsergebnis überein.



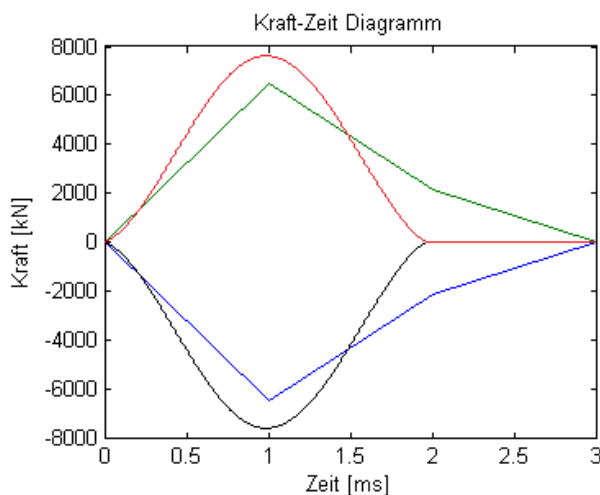
**Abb. 4.6: Zentrischer Kontakt zweier Kugeln mit Dämpfung**

Die tangentielle Feder wird überprüft indem zwei Partikel mit entgegengesetzten Geschwindigkeiten künstlich ineinander gelegt werden. Da die Partikel ihre Geschwindigkeit abgebaut haben, bevor sie sich getrennt haben. Wirken die Federkräfte und die Partikel fangen an zu schwingen.



**Abb. 4.7: Testrechnungen für die tangentielle Feder**

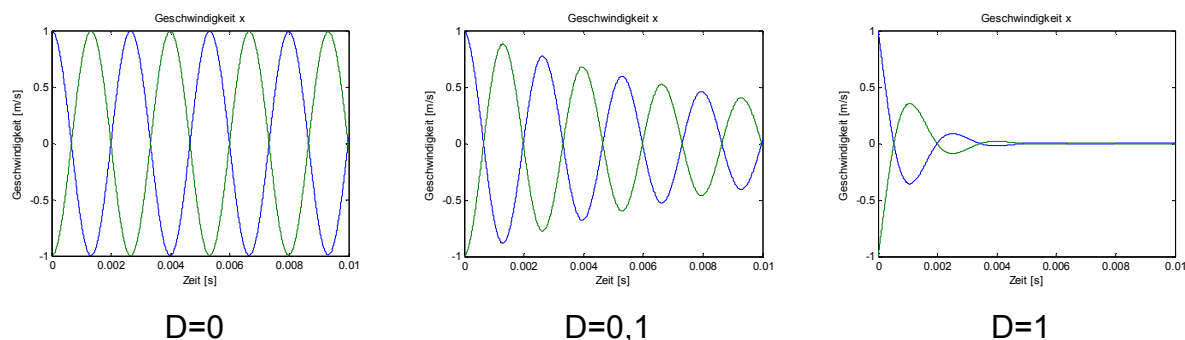
Beim Betrachten der Abbildung 4.7 fällt in den ersten beiden Diagrammen auf, dass die Geschwindigkeit in x-Richtung mit der Zeit immer kleiner wird. Dies liegt an der „numerischen Dämpfung“, welche entsteht, wenn die zeitliche Auflösung zu gering ist. Dadurch ist das Integral der Kraft über der Zeit kleiner, als es bei der analytischen Lösung wäre (s. Abb.4.8).



**Abb. 4.8: Berechnung eines Kontaktes mit unterschiedlichen Zeitschrittweiten**

In Abbildung 4.8 ist der Kraftverlauf während des Kontaktes zwischen zwei Partikeln dargestellt. Die rote und schwarze Kurve wurden bei einer zeitlichen Auflösung von  $10^{-6}$  s und die grüne und blaue Kurve wurden bei einer Auflösung von  $10^{-3}$  s aufgenommen. Es ist gut zu erkennen, dass erhebliche Abweichungen zwischen den berechneten Werten beider Kurven liegen.

Als letztes folgt die Untersuchung des tangentialen Dämpfers. Hierzu wird der vorherige Versuch wiederholt und es werden unterschiedliche Dämpfungsmaße eingestellt. Anschließend wird überprüft, ob die Ergebnisse nachvollziehbar sind. Dabei wird streng darauf geachtet, dass die Zeitschrittweite ausreichend klein ist, damit es nicht zu numerischer Dämpfung kommt.

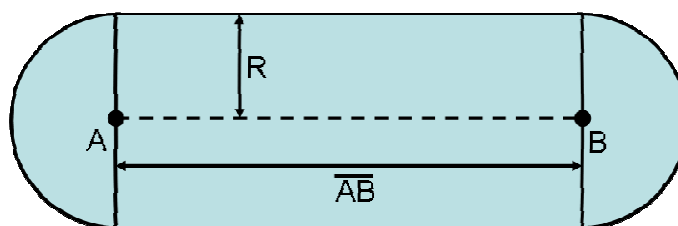


**Abb. 4.9: Testrechnungen für die tangentielle Dämpfung**

Die Berechnungsergebnisse in Abbildung 4.9 sind nachvollziehbar. Bei  $D=0$  liegt eine ungedämpfte, bei  $D=0,1$  eine leicht gedämpfte und bei  $D=1$  eine stark gedämpfte Schwingung vor.

#### 4.9.2 Randbedingung-Partikel Interaktion

Damit eine Interaktion zwischen Partikel und Wand stattfindet, muss sich das Partikel mit seinem Mittelpunkt im Einzugsbereich der Wand befinden. Dieser Bereich besteht aus einem Rechteck mit der Länge der Kante  $AB$  und der Breite gleich dem doppelten Partikelradius. Und hat an den Kanten jeweils einen Halbkreis mit dem Radius gleich Partikelradius (s. Abb. 4.10). Befindet sich ein Partikel innerhalb dieses Einzugsgebietes, so wirkt die Randbedingung „Wall“ auf dieses.



**Abb. 4.10: Einzugsgebiet der Wand**

Im Rechteck wird geprüft, ob sich der Partikelmittelpunkt in diesem befindet, indem die Höhe des Dreiecks ermittelt wird (s. Formel 4.11), welches durch die beiden Randpunkte und den Partikelmittelpunkt gebildet wird (s. Abb. 4.11). Um sicher zu gehen, dass es sich um ein stumpfes Dreieck handelt, werden die Winkel  $\alpha$  und  $\beta$  berechnet. Wenn diese beide kleiner gleich  $90^\circ$ , bzw.  $\pi/2$  sind, so befindet sich das Partikel im Einzugsbereich der Kante.

$$h_c = \frac{\sqrt{2(a^2b^2 + b^2c^2 + c^2a^2) - (a^4 + b^4 + c^4)}}{2c} \quad (4.11)$$

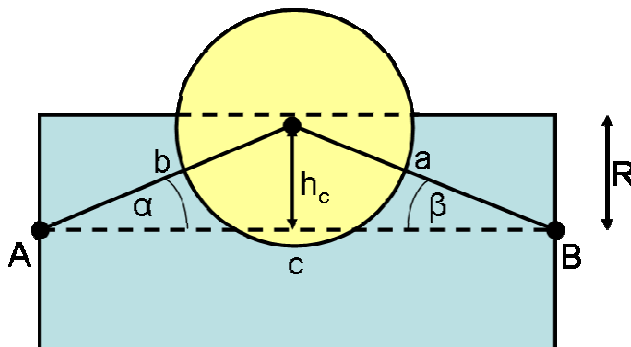


Abb. 4.11: Dreieck aus Partikelmittelpunkt und der Kante AB

Bei den beiden Halbkreisen wird nur ermittelt, ob der Abstand zwischen Kreismittelpunkt A bzw. B und Partikelmittelpunkt kleiner als der Radius des Partikels ist, ist dies der Fall, so befindet sich das Partikel im Einflussbereich des Punktes und die Randbedingungen wirken.

Ist der Abstand erkannt, so erfolgt anschließend die Berechnung der Kräfte und Momente, die auf das Partikel wirken. Die Implementierung der Berechnung findet auf nahezu dieselbe Weise, wie beim Partikel-Partikel Kontakt statt. Hierbei werden allerdings die Feder-Dämpfer Parameter auf der Hauptdiagonalen der Matrizen verwendet.

Falls die vorhandene Kante als Partikelauslass definiert ist, wird ein Hilfsvektor  $\vec{h}_v$  nach Abbildung 4.12 berechnet. Anschließend werden nach Formel 4.12 und 4.13 der Winkel zwischen dem Hilfsvektor und dem Normalenvektor  $\vec{n}$  der Kante und der Winkel zwischen der Partikelgeschwindigkeit und dem Normalenvektor der Kante berechnet.

$$\text{winkel1} = a \cos \left( \frac{\vec{n} \cdot \vec{h}_v}{|\vec{h}_v|} \right) \quad (4.12)$$

$$\text{winkel2} = a \cos \left( \frac{\vec{n} \cdot \vec{v}}{|\vec{v}|} \right) \quad (4.13)$$

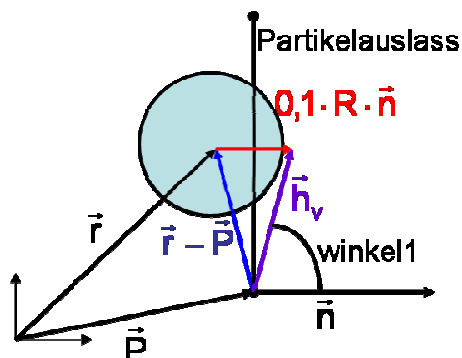


Abb. 4.12: Berechnung des Hilfsvektors

Sind die Beträge beider Winkel kleiner als  $\pi/2$ , so hat das Partikel das Berechnungsgebiet verlassen und dessen Index wird in der Variablen *deleted* abgespeichert, um mit dem Unterpogramm *Delete\_Data.m* gelöscht zu werden.

Auch für die Randbedingung findet anschließend eine Validierung statt. Hierzu wird eine Stahlkugel mit 0,5 m Radius aus 3 m Höhe fallen gelassen. Dabei wurden verschieden Dämpfungsmaße eingestellt. In Abb. 4.13 sind die Bahnen des Partikelmittelpunktes abgebildet. Die Ergebnisse des Versuchs sind durchaus nachvollziehbar.

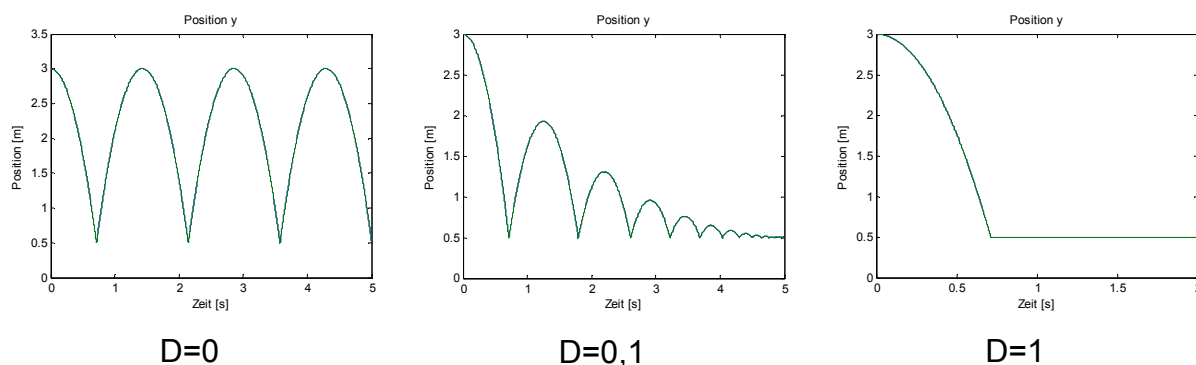
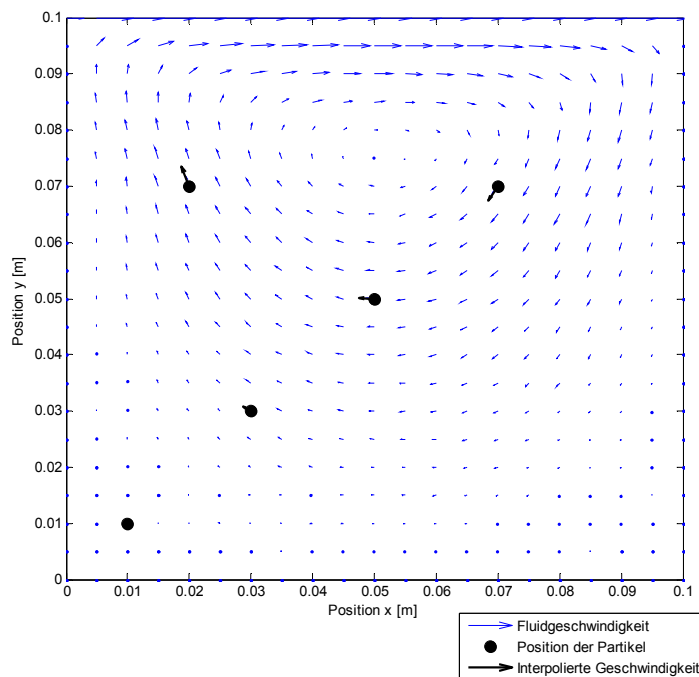


Abb. 4.13: Abprallen einer Stahlkugel vom Boden

### 4.9.3 Fluid-Partikel Interaktion

Die Berechnung der Widerstandskräfte vom Fluid auf das Partikel beginnt mit der Initialisierung der Variablen *F\_Fluid* als Nullmatrix mit 3 Zeilen und n Spalten. In dieser sollen die Kraftvektoren jedes Partikels gespeichert werden. Bei verschiedenen Testrechnungen hat sich herausgestellt, dass die Shepards Method

nur für weniger als vier Partikel schneller rechnet als interp2 Funktion. Aus diesem Grund wird über eine if-Bedingung je nach Anzahl der Partikel entschieden, welche der beiden Interpolationsmethoden verwendet wird.



**Abb. 4.14: Testrechnung, Interpolation mit der Shepards Method**

Die Shepards Method hat den großen Vorteil, dass sie sowohl für strukturierte, als auch für unstrukturierte Netze aus Strömungssimulationen verwendet werden kann. Ein Nachteil ist allerdings, dass ihre Berechnungsdauer linear mit der Anzahl der Partikel ansteigt. Für die Berechnung wird zunächst Variable *nop* (Number of Points) definiert. Sie gibt an wie viele Stützstellen (Knotenpunkte) für die Interpolation verwendet werden sollen. Standardmäßig ist die auf vier eingestellt. Dann wird mit Hilfe der Formel 4.14 der Abstand zwischen allen Knotenpunkten und allen Partikelmittelpunkten berechnet.

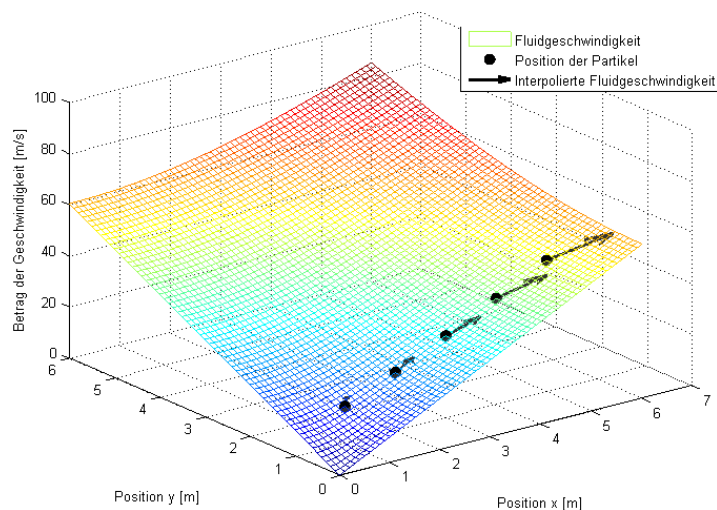
$$\vec{h} = \left[ \left[ \vec{x}_f \otimes \vec{e}_p - \vec{e}_f \otimes \vec{r}_x \right]^2 + \left[ \vec{y}_f \otimes \vec{e}_p - \vec{e}_f \otimes \vec{r}_y \right]^2 \right]^{0.5} \quad (4.14)$$

Mit Hilfe der Funktion `sort()` werden die Elemente dieser Matrix spaltenweise aufsteigend sortiert. Somit sind in den obersten Zeilen der Matrix die Abstände der jeweiligen Partikelmittelpunkte zu den nächsten Nachbarknoten. Diese werden nun für die Interpolation mit der Shepards Method nach den Formeln 3.69 und 3.70

verwendet. Die interpolierten Geschwindigkeiten in x- und y-Richtung werden dann in den beiden Arrays  $v_{xf}$  und  $v_{yf}$  abgespeichert.

Die `interp2` Funktion hat den großen Vorteil, dass ihre Berechnungsdauer nahezu unabhängig von der Anzahl der Partikel ist. Dafür bringt sie einen großen Nachteil mit sich. Sie ist auf Netze angewiesen, deren Punkte sowohl in x- als auch in y-Richtung in parallelen Reihen liegen, wobei jede Punktreihe gleich viele Punkte haben muss. Die Benutzung dieser Funktion ist recht einfach. Es müssen lediglich die Knotenkoordinaten des Netzes, die Geschwindigkeitsvektoren des Fluids an den Knoten, sowie die Partikelpositionen an diese übergeben werden. Die Ergebnisse der Interpolation werden in den beiden Arrays  $v_{xf}$  und  $v_{yf}$  abgespeichert.

Wurde die Interpolation durchgeführt, dann kann aus den interpolierten Geschwindigkeiten zunächst die relative Geschwindigkeit zwischen Partikel und Fluid ermittelt werden. Aus dieser lässt sich dann wiederum die Fluidwiderstandskraft auf die einzelnen Partikel berechnen. Hierzu wird Formel 3.64 verwendet. Die berechneten Kräfte werden dann in der Matrix  $F_{Fluid}$  unter ihrer jeweiligen Position abgespeichert.



**Abb. 4.15: Testrechnung, Interpolation mit `interp2`**

In Abbildung 4.15 ist eine Testrechnung für die Interpolation mit der `interp2` Funktion dargestellt. Das farbige Netz ist die Visualisierung des Fluidgeschwindigkeitsbetrags





in die x-Richtung. Hierbei wurde visuell überprüft, ob die interpolierten Fluidgeschwindigkeiten korrekt sind.

Wenn unstrukturierte Netze für die Strömungssimulation verwendet werden, kann nur die Shepards Method angewendet werden, da interp2 auf strukturierte Netze angewiesen ist

#### 4.9.4 Gravitation

Die Gravitation wird auf folgende Weise berechnet. Zuerst wird die Variable  $F\_G$  als Nullmatrix mit  $3 \times n$  Elementen initialisiert. Darauf wird in einer doppelten verschachtelten for-Schleife für alle Partikelpaarungen der Normalenvektor ermittelt. Dieser wird anschließend normiert und unter  $n\_norm$  abgespeichert. Daraufhin werden die Gravitationskräfte für jede Partikelpaarung aus dem Produkt aus Formel 3.72 und  $n\_norm$  berechnet. Zum Schluss werden alle Kräfte die jeweils auf ein Partikel wirken aufsummiert und an der entsprechenden Position in der Matrix  $F\_G$  abgespeichert.

#### 4.9.5 Aufsummierung der Kräfte

Der letzte Schritt der Funktion Dyn\_Force() ist die Erstellung der Rückgabewerte an das Hauptprogramm. Hierfür werden die Matrizen mit den Kräften bzw. Momenten zunächst aufsummiert und anschließend mit dem jeweiligen Richtungsvektor multipliziert, so dass die Kräfte in x- und y-Richtung bzw. das Moment um die z-Achse als Vektoren vorliegen.

$$\vec{F}_{\text{Dyn},x} = (\vec{F}_{\text{Kollision}} + \vec{F}_{\text{Boundary}} + \vec{F}_{\text{Fluid}} + \vec{F}_{\text{Gravitation}})^T \vec{x}_{\text{dir}}$$

$$\vec{F}_{\text{Dyn},y} = (\vec{F}_{\text{Kollision}} + \vec{F}_{\text{Boundary}} + \vec{F}_{\text{Fluid}} + \vec{F}_{\text{Gravitation}})^T \vec{y}_{\text{dir}}$$

$$\vec{M}_{\text{Dyn},z} = (\vec{M}_{\text{Kollision}} + \vec{M}_{\text{Boundary}})^T \vec{z}_{\text{dir}}$$



#### 4.10 Funktion `kreuz.m`

Die Funktion `kreuz.m` dient der Berechnung des Kreuzproduktes zwischen zwei dreidimensionalen Vektoren. Hierzu werden der Funktion zwei dreidimensionale Spaltenvektoren  $a$  und  $b$  übergeben und der berechnete Spaltenvektor  $c$  wird wieder zurückgegeben. Zwar ist in Matlab bereits das Kreuzprodukt unter der Funktion `cross()` integriert, allerdings ist diese Funktion erheblich langsamer, da sie auch für mehrdimensionale Vektoren definiert ist. Eine Testrechnung, bei der jeweils 100.000 Berechnungen mit denselben Vektoren durchgeführt wurden, ergab, dass die Funktion `kreuz()` ca. 6 bis 7 Mal schneller ist als `cross()`

Ablauf der Berechnung:  $c_1 = a_2 b_3 - b_2 a_3$

$$c_2 = a_3 b_1 - b_3 a_1$$

$$c_3 = a_1 b_2 - b_1 a_2$$

#### 4.11 Unterprogramm `Delete_Data.m`

Das Unterprogramm `Delete_Data.m` dient dazu, die Daten von Partikeln, die das Berechnungsgebiet verlassen haben, aus den jeweiligen Vektoren bzw. Matrizen zu löschen. In der Variablen `deleted` sind dazu die Indizes von Partikeln abgespeichert, die innerhalb des aktuellen Zeitschritts das Berechnungsgebiet verlassen haben. Die Reihen und Spalten mit denselben Indizes werden dann aus den folgenden Vektoren bzw. Matrizen gelöscht.

- Bewegungsdaten des Partikels
  - o  $r_x_p, r_y_p, phi_p, v_x_p, v_y_p, omega_p, a_x_p, a_y_p$  und  $alpha_p$
- Integrierte Geschwindigkeiten:
  - o  $int_v_t, int_v_roll, int_t_e, int_roll_e, int_t_p, int_roll_p$
- Effektive Geometrie- und Materialeigenschaften
  - o  $R_eff, E_eff, m_eff, nue_eff$
- Feder- und Dämpfungskonstanten
  - o  $kn, cn, kt, ct, kr, cr$



Außerdem wird Variable *part*, die die Anzahl der Partikel im Berechnungsgebiet beinhaltet, um die Anzahl der in *deleted* enthaltenen Indizes reduziert und abschließend wird die Variable *deleted* geleert.

### 4.12 Unterprogramm `Convert_ml2vtk.m`

Das Unterprogramm `Convert_ml2vtk.m` dient dazu, die mit dem Programm berechneten Partikel Daten in das für ParaView lesbare `vtk`-Datei umzuwandeln. Dazu werden zunächst Variablen initialisiert, in die die Berechnungsdaten eines Zeitschritts abgelegt werden. Dies sind die Position, Geschwindigkeit und Beschleunigung für Translation und Rotation, sowie der Durchmesser und die Partikelmasse. Innerhalb einer `for`-Schleife werden die Variablen aufgefüllt. Die Berechnungsdaten werden dabei als Zeichenkette abgespeichert. An bestimmte Stellen müssen Tabulatoren oder Zeilenumbrüche eingefügt werden, um das für die `vtk`-Datei notwendige XML-Format (Extensible Markup Language) zu erhalten [Int9]. Anschließend wird die `vtk`-Datei erstellt und im Schreibmodus geöffnet. Mit Hilfe der Funktion `fprintf()` wird nun die Datei mit den Daten im XML-Format gefüllt. Danach wird die Datei wieder geschlossen und die nicht mehr benötigten Variablen werden gelöscht.

## 5 Verifizierung des Modells

### 5.1 Beispielrechnung 1

Das erste Beispiel behandelt die Nachrechnung der von Heinrich Hertz in seiner Veröffentlichung „Ueber die Berührung fester elastischer Körper“ [Lit12] berechneten Kollisionszeiten von Stahlkugeln zur Kontrolle des Kollisionsalgorithmus.

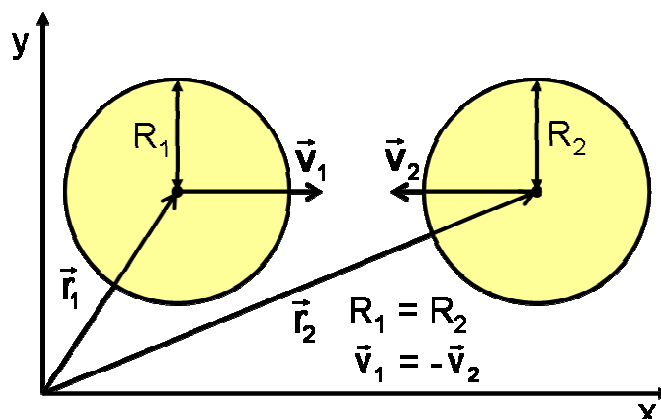


Abb. 5.1: Kugel-Kugel Kontakt

In seiner Veröffentlichung hat Hertz zwei unterschiedliche Kugelpaarungen untersucht und deren Kontaktzeit bei einer relativen Kollisionsgeschwindigkeit von 10 mm/s berechnet. Die Kugeln der ersten Paarung hatten einen Radius von 25 mm und die der zweiten Paarung waren so groß wie die der Planet Erde. Als Kollisionszeiten ermittelte er für die erste Paarung eine Zeitspanne von 0,38 ms und für die zweite Paarung eine Zeitspanne von ca. 27 Stunden.

Diese beiden Fälle sollen mit dem entwickelten Programm nachgerechnet werden. In seiner Veröffentlichung hat er bis auf den verwendeten Erdradius alle benötigten Werte zur Berechnung angegeben. Da die Veröffentlichung allerdings aus dem Jahr 1881 stammt ist der Elastizitätsmodul noch in der Einheit kg/mm<sup>2</sup> angegeben. Diese Einheit muss in die heute gebräuchliche Einheit für den Elastizitätsmodul N/mm<sup>2</sup> durch Multiplikation mit der Erdbeschleunigung umgerechnet werden.

$$E_{\text{Hertz}} = 20000 \frac{\text{kg}}{\text{mm}^2} \hat{=} 9,81 \frac{\text{m}}{\text{s}^2} \cdot 20000 \frac{\text{kg}}{\text{mm}^2} = 196200 \frac{\text{N}}{\text{mm}^2} \quad (5.1)$$

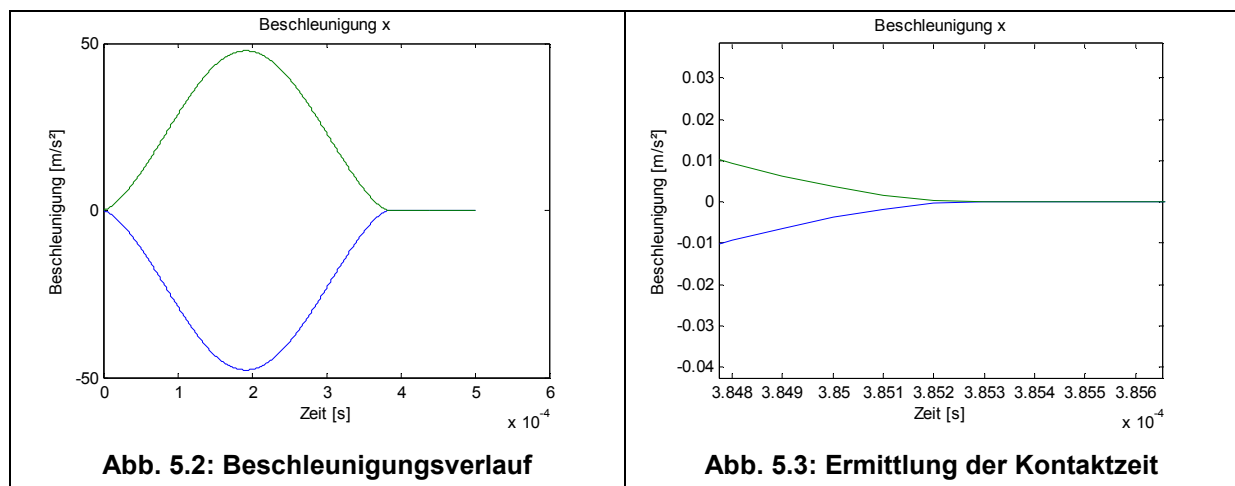
Fall\Eigenschaft	Dichte [ $\text{kg}\cdot\text{m}^{-3}$ ]	E-Modul [ $\text{N}\cdot\text{mm}^{-2}$ ]	Poisson-Zahl [-]	Radius	Zeit
2 kleine Stahlkugeln	7700	196200	0,3	0,025 m	0,3852 ms
2 große Stahlkugeln	7700	196200	0,3	6.000 km	27 h

Tab. 5.1: Parameter für die Berechnung

Für die Berechnung der ersten Kugelpaarung werden für die Kugeln die Eigenschaften aus Tabelle 5.1 1. Zeile, sowie folgende Berechnungseinstellungen verwendet:

- Rechenzeit:  $5\cdot 10^{-4}$  s
- Zeitschrittweite:  $10^{-7}$  s
- Abbruchkriterium der PK-Schleife:  $10^{-5}$

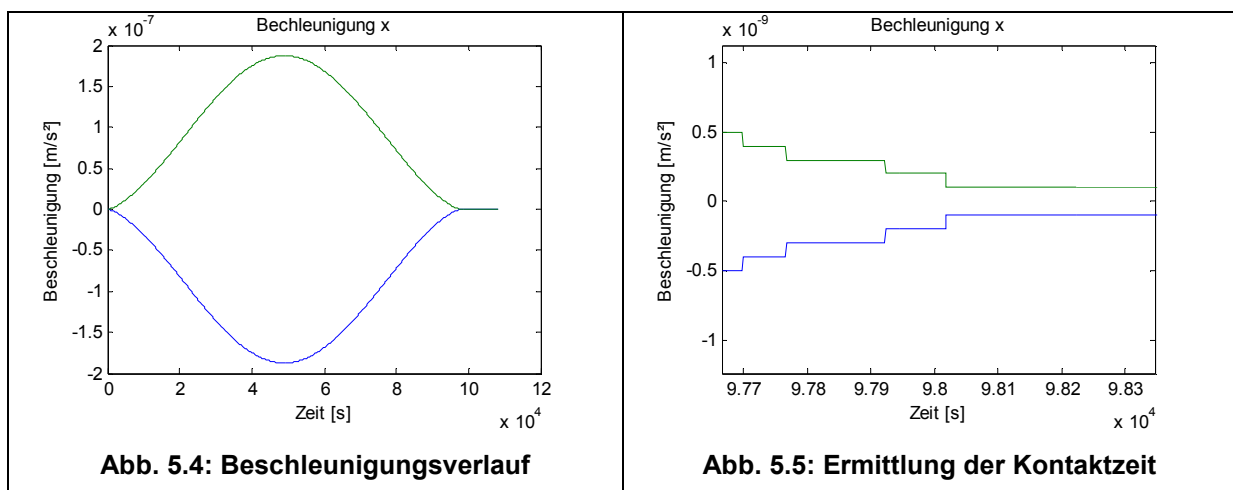
Das Ergebnis der Berechnung ist in Abbildung 5.2 und Abbildung 5.3 zu sehen. In Abb. 5.2 ist die Beschleunigung in x-Richtung der beiden Kugeln über die Berechnungszeit aufgetragen. Man kann schön erkennen, wie sich diese während des Kontaktes aufbaut und nach der maximalen Annäherung beider Kugeln bei ca.  $1,8\cdot 10^{-4}$  s wieder abbaut. In Abb. 5.3 wird das Ende des Kontaktes vergrößert dargestellt, um die Kontaktzeit zu ermitteln. Wie aus der Abbildung zu erkennen ist, dauert der Kontakt ca.  $3,852\cdot 10^{-4}$  s. Dies entspricht mit einer Abweichung von ca. 1,4 % dem Ergebnis von Hertz.



Als nächstes soll das Kugelpaar mit der Größe der Erde berechnet werden. Hierfür werden die Eigenschaft aus Tab. 5.1 2. Zeile, sowie folgende Berechnungseinstellungen verwendet:

- Rechenzeit: 108000 s = 30 h
- Zeitschrittweite: 1 s
- Abbruchkriterium der Praediktor-Korrektor Iteration:  $10^{-10}$

Das Ergebnis dieser Berechnung ist in den Abbildungen Abb. 5.4 und Abb. 5.5 dargestellt. Bei Vergrößerung des Endes des Kontaktes Abb. 5.5 lässt sich eine Kontaktzeit von  $9,805 \cdot 10^4$  s ablesen, was umgerechnet ca. 27,236 h entspricht. Hertz hat für diesen Kontakt eine Zeitdauer von ca. 27 h angegeben. Somit ist dies ein gutes Ergebnis.



Da beide Berechnungen nur sehr geringe Abweichungen zu den Ergebnissen von Hertz haben, kann davon ausgegangen werden, dass die Kontaktberechnung des entwickelten Programms sehr gut funktioniert. Auch die enormen Größenunterschiede zwischen der ersten und zweiten Kugelpaarung, und daraus folgende enorme Unterschiede der Beschleunigungen (Kugelpaar 1, maximale Beschleunigung ca.  $50 \text{ m/s}^2$ ; Kugelpaar 2, maximale Beschleunigung ca.  $2 \cdot 10^{-7} \text{ m/s}^2$ ) wirken sich nicht negativ auf die Berechnung aus.



## 5.2 Beispielrechnung 2

Das zweite Beispiel behandelt das Schweben eines Partikels im Luftstrom. Bei diesem Versuch soll durch eine Beispielrechnung untersucht werden, ob die einseitige Kopplung von Strömungsfeld und Partikel funktioniert. Hierzu wird das Schweben einer tischtennisballgroßen Styroporkugel in einem vertikalen Luftstrom mit Hilfe des entwickelten Programms nachgebildet.

Um die Strömungsgeschwindigkeit des Mediums zu ermitteln, wird folgende Rechnung gemacht. Der Zustand des Schwebens tritt ein, sobald sich ein Kräftegleichgewicht an der Kugel eingestellt hat. Hierzu müssen sich Erdanziehungs- und Fluidkräfte gegenseitig aufheben:

$$F_g + F_A + F_w = 0 \quad (5.2)$$

Hierbei ist  $F_g$  die Erdanziehungskraft,  $F_A$  der statische Auftrieb und  $F_w$  die Fluidwiderstandskraft. Werden nun die Formeln für die jeweiligen Kräfte eingesetzt, ergibt sich folgende Gleichung:

$$-\rho_p V_p g + \rho_f V_p g + \frac{1}{2} \rho_f c_w A_p v^2 = 0 \quad (5.3)$$

Diese Formel wird nun nach der Geschwindigkeit  $v$  umgestellt:

$$v = \sqrt{\frac{2V_p g(\rho_p - \rho_f)}{\rho_f c_w A_p}} = \sqrt{\frac{8\pi r^3 g(\rho_p - \rho_f)}{3\pi r^2 \rho_f c_w}} = \sqrt{\frac{8}{3} \frac{rg(\rho_p - \rho_f)}{\rho_f c_w}} \quad (5.4)$$

Nach Einsetzen der Werte aus Tab. 5.2 ergibt sich eine Fluidgeschwindigkeit von ca. 3,5 m/s. Diese Geschwindigkeit muss das Fluid haben, um die Kugel schweben zu lassen. Ist die Geschwindigkeit geringer, dann sinkt die Kugel ab. Bei höherer Geschwindigkeit wird die Kugel mit dem Fluidstrom mitgerissen.

$$v = \sqrt{\frac{8 \cdot 0,02 \cdot 9,81 \cdot (15 - 1,25)}{3 \cdot 1,25 \cdot 0,47}} \approx \underline{\underline{3,5 \frac{\text{m}}{\text{s}}}} \quad (5.5)$$

Eigenschaft	Wert
Kugelradius	0.02 m
Dichte des Styropors	15 kg/m <sup>3</sup>
Dichte der Luft	1,25 kg/m <sup>3</sup>
c <sub>w</sub> -Wert (Kugel)	0,47

Tab. 5.2: Eigenschaften

Anschließend wird mit dem Programm OpenFOAM eine laminare, inkompressible Strömungssimulation erstellt. Das Berechnungsgebiet hat dabei eine Größe von 6 m x 6 m. Der Einlass (Velocity-Inlet) befindet sich auf der Unterseite, der Auslass (Pressure-Outlet) auf der Oberseite und die linke und rechte Seite werden als Wände modelliert. Das Berechnungsgebiet wird so groß gewählt, da sich aufgrund der Reibung an den Wänden eine parabelförmige Verteilung der Geschwindigkeit ergibt. Somit ist die Geschwindigkeit direkt am Rand 0 m/s und in der Mitte ist sie etwas höher als die am Einlass eingestellte Geschwindigkeit. Je größer das Berechnungsgebiet, desto weniger Einfluss hat die Wandreibung auf die Strömung in der Mitte des Rechengebietes.

Die berechnete Strömungsgeschwindigkeit von 3,5 m/s wird nun als Einlassgeschwindigkeit eingestellt und es ergibt sich folgende Geschwindigkeitsverteilung im Rechengebiet (siehe Abb. 5.6). Wie an der Skala in der Abbildung zu sehen ist, ist die maximale Geschwindigkeit in der Mitte des Berechnungsgebiets mit 3,592 m/s etwas höher als die gewünschte.

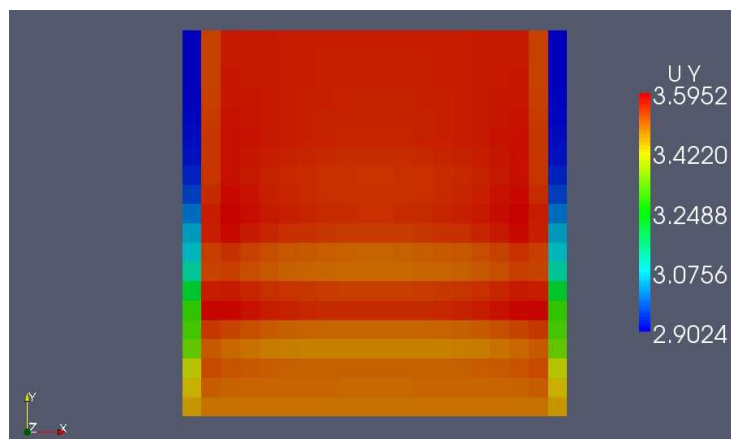
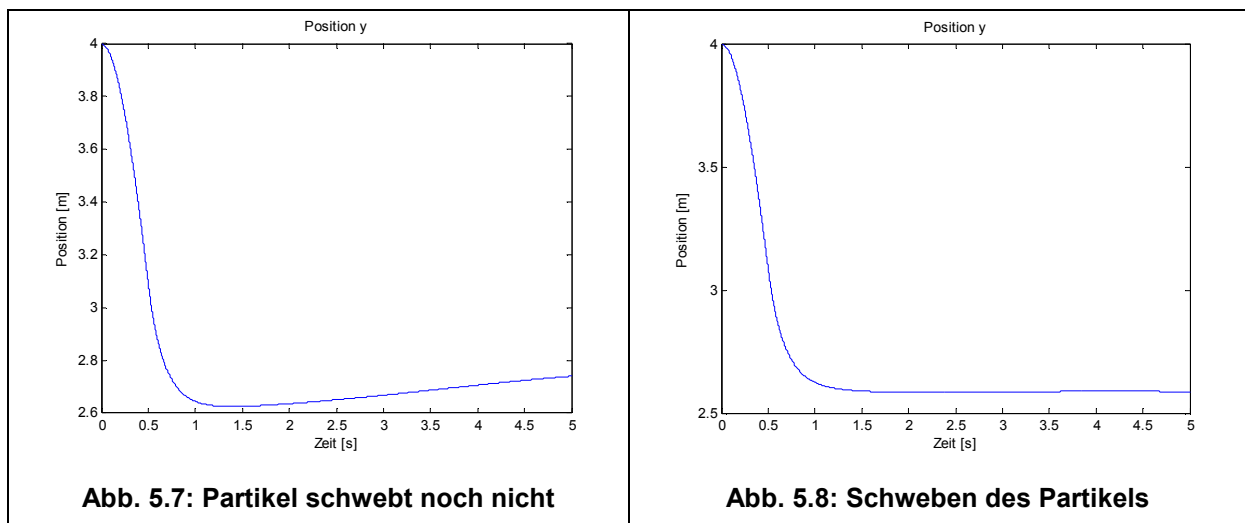


Abb. 5.6: Strömungsfeld



Trotzdem wird diese Simulation für die weiteren Berechnungen in diesem Beispiel verwendet. Die Strömungsdaten werden nun in den Projektordner kopiert und die Berechnung mit dem entwickelten Programm gestartet.

Wie in Abb. 5.7 zu sehen ist, fällt die Kugel zunächst aufgrund der wirkenden Schwerkraft hinunter. Nach 0,5 s hat sich die Strömung im Berechnungsgebiet eingestellt und die Kugel wird abgebremst, bis sie bei ca. 1,5 s zur Ruhe kommt. Danach wird sie durch die Strömung wieder nach oben befördert.



Um das Partikel doch noch zum Schweben zu bringen, wird der Radius der Kugel vergrößert. Hierbei wurden einige Versuche mit unterschiedlichen Radien ausprobiert. Bei einem Radius von 0,0204 m stellte sich, wie in Abb. 5.8 zu sehen ist ein Schweben des Partikels ein.



## 6 Zusammenfassung

### 6.1 Fazit

Im Rahmen dieser Master-Thesis ist es gelungen ein Programm zu entwickeln, dass zur Berechnung von Partikelbewegungen und -kollisionen genutzt werden kann. Dabei wurden viele Problemstellungen die zunächst auftraten, erfolgreich bewältigt. Dies war vor allem die die Implementierung von Partikeleinlass und -auslass, sowie die Kollisionserkennung von Kugel-Kugel und Kugel-Wand Kontakten. Für die Kollisionserkennung wurden im Laufe dieser Arbeit mehrere Kontakterkennungsalgorithmen, z.B. das Neighboring Cells Verfahren implementiert. Die im praktischen Test aber nicht so effizient, wie das jetzt verwendete Verfahren waren.

Eine Schwierigkeit zu Beginn der Arbeit war, dass die Thematik der Discrete Element Method nahezu unbekannt war und somit am Anfang viele Fehler gemacht wurden. Somit traten bei den ersten Implementierungsversuchen immer wieder Fehler auf, die Anfangs nicht zu erklären waren. Später stellte sich dann heraus, dass die meisten Fehler entstanden, weil die Zeitschrittweite z.B. mit  $10^{-4}$  s in den meisten Fällen immer noch zu grob gewählt war. Die falsche Wahl des Zeitschritts sorgte z.B. dafür, dass Dämpfung im System vorlag, obwohl diese noch gar nicht implementiert war (Numerische Dämpfung).

Während der Erstellung des Programms wurde versucht die Berechnung von sämtlichen Kräften und Momenten mit Vektoren zu durchzuführen, damit bei einer eventuellen Implementierung des Programms in 3D nicht alles neu geschrieben werden muss.

Als sehr nützlich bei der Implementierung des Programms hat sich noch die Profiler Funktion in Matlab herausgestellt. Diese gibt zum Ende der Berechnung an, welche Rechenschritte und Funktionen welchen Anteil an der Gesamtrechnzeit hatten. So wurde schnell klar, welche Funktionen umgeschrieben werden müssen, um das Programm effizienter zu machen.



## 6.2 Ausblick

In diesem Kapitel soll ein kurzer Ausblick über weitere Ausbaumöglichkeiten dieses Programms gegeben werden.

Da zurzeit nur eine einseitige Kopplung zwischen Strömung und Partikeln stattfindet (Strömung beeinflusst die Partikel, aber das Partikel beeinflusst nicht die Strömung), wäre es für die Berechnung von Mehrphasenströmungen sinnvoll eine vollständige Kopplung zwischen CFD und DEM zu realisieren. Die Reaktionskräfte der Partikel auf das Fluid könnten hierzu über Quellterme in die Navier-Stokes-Gleichung und in die Kontinuitätsgleichung eingebunden werden.

Zur Effizienzsteigerung bei sehr vielen Partikeln, die in Kontakt stehen, könnten die for-Schleifen noch ersetzt werden, indem auch hier mit Matrizen gerechnet wird. Dies ist allerdings sehr umständlich, da die Kräfte dann nicht mehr vektoriell, sondern mittels Betrag und Winkel berechnet werden müssten. Außerdem könnte versucht werden, dass Programm in C++ zu implementieren, da aufgrund der Matlab Oberfläche bestimmt auch Rechenkapazität verloren geht.

Für die Berechnung sehr feiner Pulver müssen zusätzliche Kräfte, die im Mikrobereich wirken, in das Modell eingefügt werden. Dies sind z.B. die Van-der-Waals Kraft oder die elektrostatische Kraft nach Coulomb. Zusätzlich wäre es sinnvoll eine Randbedingung „Moving Wall“ zu implementieren, um Scherversuche nachmodellieren zu können.

Um die Benutzerfreundlichkeit zu erhöhen, wäre es sinnvoll eine GUI zu erstellen, in der alle Berechnungsparameter usw. über Menüs eingegeben werden könnten. Auch könnte die Geometrie des Berechnungsgebietes direkt aus den Randgeometrien der Strömungssimulation gelesen werden.



## Literaturverzeichnis

Quellen aus Fachbüchern, Fachzeitschriften und Vorlesungsskripten

- [Lit1] Allen, M. P.; Tildesley, D. J.: *Computer simulation of liquids*, Fachbuch, Clarendon Press, Oxford 1987
- [Lit2] Clayton, C.; Sommerfeld, M.; Tsuji, Y.: *Multiphase Flows with Droplets and Particles*, Fachbuch, CRC Press, New York, 1997
- [Lit3] Eck, C.; Garcke, H.; Knabner, P.: *Mathematische Modellierung*, Fachbuch, Springer-Verlag, Berlin 2008.
- [Lit4] Elata, D.; Berryman, J. G.: *Contact force-displacement laws and the mechanical behavior of random packs of identical spheres*, Artikel in Fachzeitschrift, *Mechanics of Materials*, S. 229-240, 1996
- [Lit5] Fujita, M.; Yamaguchi, Y.: *Multiscale simulation method for self-organization of nanoparticles in dense suspension*, Artikel in Fachzeitschrift, *Journal of Computational Physics*, S. 108-120, 2007
- [Lit6] Griebel, M.; Knapek, S.; Zumbusch, G.; Caglar, A.: *Numerische Simulation in der Moleküldynamik*, Fachbuch, Springer-Verlag, Berlin 2004.
- [Lit7] Gross, D.; Hauger, W.; Schröder, J.; Wall, W. A.: *Technische Mechanik 1: Statik*, Fachbuch, Springer-Verlag, Berlin 2008
- [Lit8] Gross, D.; Hauger, W.; Schröder, J.; Wall, W. A.: *Technische Mechanik 2: Elastostatik*, Fachbuch, Springer-Verlag, Berlin 2009
- [Lit9] Gross, D.; Hauger, W.; Schröder, J.; Wall, W. A.: *Technische Mechanik 3: Kinetik*, Fachbuch, Springer-Verlag, Berlin 2010



- [Lit10] Hertz, H.: *Ueber die Berührung fester elastischer Körper*, Artikel in Fachzeitschrift, Journal für die reine und angewandte Mathematik, S. 156-171, 1881
- [Lit11] Ihlenburg, F.: *Mathematische Methoden*, Vorlesungsskript, HAW Hamburg, Hamburg 2009
- [Lit12] Johnson, K. L.: *Contact force-displacement laws and the mechanical behavior of random packs of identical spheres*, Artikel in Fachzeitschrift, Mechanics of Materials, S. 229-240, 1996
- [Lit13] Kok, G.: *Strömungslehre 1*, Vorlesungsskript, FH OOW Emden, Emden 2006
- [Lit14] Kruggel-Emden, H.; Rickelt, S.; Wirtz, S.; Scherer, V.: *A study on the validity of the multi-sphere Discrete Element Method*, Artikel in Fachzeitschrift, Powder Technology, S. 153-165, 2008
- [Lit15] Marshall, J. S.: *Discrete-element modeling of particulate aerosol flows*, Artikel in Fachzeitschrift, Journal of Computational Physics, S. 1541-1561, 2009
- [Lit16] Molerus, O.: *Schüttgutmechanik. Grundlagen und Anwendungen in der Verfahrenstechnik*, Fachbuch, Springer-Verlag, Berlin 1985
- [Lit17] Munjiza, A.: *The Combined Finite-Discrete Element Method*, Fachbuch, John Wiley & Sons, Chichester 2004
- [Lit18] Tomas, J. (2009). *Produkteigenschaften ultrafeiner Partikel - Mikromechanik, Fließ- und Kompressionsverhalten kohäsiver Pulver*, Artikel in Fachbuch, Sächsische Akademie der Wissenschaften zu Leipzig, Leipzig 2009
- [Lit19] Volfson, D.; Tsimring, L. S.; Aranson, I. S.: *Partially fluidized shear granular flows: Continuum theory and molecular dynamics simulation*, Artikel in Fachzeitschrift, Physical Review, S. 1-15, 2003
- [Lit20] Wassgren, C.: *DEM Modeling*. Vorlesungsskript, Purdue University, West Lafayette



[Lit21] Wulf, P.: *Antrag zum Projekt SimPneuTrans*, Hamburg 2009

[Lit22] Wulf, P.: *Computational Fluid Dynamics*, HAW Hamburg, Hamburg 2010

[Lit23] Zhang, S., Kubawara, S., Suzuki, T., Kawano, Y., Morita, K., & Fukuda, K.:  
*Simulation of solid-fluid mixture flow using moving particle methods*, Journal of  
Computational Physics, S. 2552-2565, 2009

Quellen aus dem Internet:

[Int1] <http://www.wikipedia.de>

[Int2] [https://twiki.auscope.org/wiki/pub/EarthSim/ExistingInfrastructure/  
CollapsingBlock3.png](https://twiki.auscope.org/wiki/pub/EarthSim/ExistingInfrastructure/CollapsingBlock3.png)

[Int3] <http://geo.hmg.inpg.fr/frederic/pictures/impacts/perforation.jpg>

[Int4] <http://ditwww.epfl.ch/SIC/SA/publications/SCR99/scr11-page25a.jpeg>

[Int7] [http://www.che.ufl.edu/images/Curtis\\_Index\\_1.jpg](http://www.che.ufl.edu/images/Curtis_Index_1.jpg)

[Int6] <http://www.ibnm.uni-hannover.de/typo3temp/pics/81ee61d4dd.jpg>

[Int7] [http://www.process.vogel.de/mechanische\\_verfahrenstechnik/  
schuettguttechnik/articles/107232/](http://www.process.vogel.de/mechanische_verfahrenstechnik/schuettguttechnik/articles/107232/)

[int8] [http://www.ems-i.com/gmshelp/Interpolation/Interpolation\\_Schemes/  
Inverse\\_Distance\\_Weighted/Shepards\\_Method.htm](http://www.ems-i.com/gmshelp/Interpolation/Interpolation_Schemes/Inverse_Distance_Weighted/Shepards_Method.htm)

[Int9] <http://www.mail-archive.com/paraview@paraview.org/msg01149.html>



## **Erklärung über eigenständige Erstellung der Arbeit**

Ich versichere hiermit, dass ich die vorliegende Master-Thesis selbstständig verfasst, weder ganz noch in Teilen als Prüfungsleistung vorgelegt und keine anderen als die angegebenen Hilfsmittel benutzt habe. Sämtliche Stellen der Arbeit, die benutzten Werken im Wortlaut oder dem Sinn nach entnommen sind, habe ich durch Quellenangaben kenntlich gemacht. Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen und dergleichen, sowie für Quellen aus dem Internet.

Hamburg, den 30.06.2011

---

Unterschrift des Studenten



## Anhang

Im Anhang ist der Programmcode der im Zuge dieser Master-Thesis erstellten Programme enthalten. Außerdem gibt es eine CD-ROM, auf der die Programme und die Beispielrechnungen, sowie eine kurze Anleitung und ein Video einer Partikelsimulation enthalten sind.

### Inhaltsverzeichnis

Hauptprogramm Discrete_Element_Modeler.m....	<b>Fehler! Textmarke nicht definiert.</b>
Unterprogramm Write_Startvalues.m .....	<b>Fehler! Textmarke nicht definiert.</b>
Unterprogramm Read_Startvalues.m .....	<b>Fehler! Textmarke nicht definiert.</b>
Unterprogramm Write_Boundary.m .....	<b>Fehler! Textmarke nicht definiert.</b>
Unterprogramm Read_Boundary.m.....	<b>Fehler! Textmarke nicht definiert.</b>
Unterprogramm Write_Conditions.m .....	<b>Fehler! Textmarke nicht definiert.</b>
Unterprogramm Read_Conditions.m .....	<b>Fehler! Textmarke nicht definiert.</b>
Unterprogramm Import_CFD.m .....	<b>Fehler! Textmarke nicht definiert.</b>
Funktion Calculate_Forces.m .....	<b>Fehler! Textmarke nicht definiert.</b>
Funktion kreuz.m .....	<b>Fehler! Textmarke nicht definiert.</b>
Unterprogramm Delete_Data.m.....	<b>Fehler! Textmarke nicht definiert.</b>
Unterprogramm Convert_ml2vtk.m.....	<b>Fehler! Textmarke nicht definiert.</b>
CD-ROM.....	<b>Fehler! Textmarke nicht definiert.</b>