Methoden der Mehrkörperdynamiksimulation als Grundlage realitätsnaher Virtueller Welten

Von der Fakultät für Elektrotechnik und Informationstechnik
der Rheinisch-Westfälischen Technischen Hochschule Aachen
genehmigte Dissertation zur Erlangung des akademischen Grades eines
Doktors der Ingenieurwissenschaften (Dr.-Ing.)

vorgelegt von

Diplom-Informatiker Thomas Josef Jung
aus Rudersdorf im Kreis Siegen, Nordrhein-Westfalen

Berichter: Universitätsprofessor Dr.-Ing. Jürgen Roßmann
Universitätsprofessor Dr.-Ing. Jürgen Gausemeier

Tag der mündlichen Prüfung: 5. Oktober 2011

D 82 (Diss. RWTH Aachen University, 2011)

Danksagung

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Mensch-Maschine-Interaktion (MMI) der Rheinisch-Westfälischen Technischen Hochschule (RWTH) Aachen.

Herrn Professor Dr.-Ing. Jürgen Roßmann, dem Leiter des Instituts, gilt mein besonderer Dank für die stetige Förderung der Arbeit, viele wertvolle inspirierende Impulse und die Schaffung ausgezeichneter Rahmenbedingungen, die in hohem Maße zum Gelingen der Arbeit beigetragen haben.

Herrn Professor Dr.-Ing. Jürgen Gausemeier danke ich für das der Arbeit entgegengebrachte Interesse und die freundliche Übernahme des Korreferats.

Allen meinen Kollegen am Institut für Mensch-Maschine-Interaktion gebührt Dank für die immer gute Zusammenarbeit und das freundschaftliche Arbeitsklima.

Meiner lieben Frau Steffi danke ich für ihre unbedingte Unterstützung, für das meiner Arbeit entgegengebrachte große Interesse trotz vieler Entbehrungen und die mir daraus erwachsene Motivation.

Aachen, im Oktober 2011

Thomas Josef Jung

Inhaltsverzeichnis

1. Einleitung **9**
- 1.1. Motivation . . . 9
- 1.2. Aufbau der Arbeit . . . 13

2. Verfahren der Simulation von Mehrkörpersystemen: Stand der Technik **15**
- 2.1. Teilaufgaben einer Mehrkörperdynamiksimulation . . . 16
- 2.2. Steuerung des zeitlichen Ablaufs . . . 18
  - 2.2.1. Variable Schrittweiten mit Zeitschritt-Unterteilungsverfahren . . . 19
  - 2.2.2. Variable Schrittweiten mit dem Einseiten-Verfahren . . . 19
  - 2.2.3. Variable Schrittweiten mit dem Timewarp-Algorithmus . . . 20
  - 2.2.4. Zeitliche Steuerungen mit konstanter Schrittweite . . . 20
- 2.3. Kollisionserkennung und Bestimmung von Kontaktpunkten . . . 22
  - 2.3.1. Verfahren in der Grobphase . . . 23
  - 2.3.2. Verfahren in der Detailphase . . . 27
- 2.4. Die Kollisionsbehandlung . . . 34
  - 2.4.1. Grundlagen . . . 35
  - 2.4.2. Newtons Stoßgesetz . . . 37
  - 2.4.3. Poissons Hypothese . . . 37
  - 2.4.4. Stronges Hypothese . . . 37
  - 2.4.5. Einbindung der Kollisionsbehandlung in den Simulationsablauf . . . 38
- 2.5. Verfahren zur Einhaltung von Zwangsbedingungen . . . 39
  - 2.5.1. Grundlagen: Zwangsbedingungen und Zwangskräfte . . . 39
  - 2.5.2. Penalty Verfahren . . . 44
  - 2.5.3. Sequenzielle, impulsbasierte Verfahren . . . 45
  - 2.5.4. Verfahren der positionsbasierten Projektion . . . 50
  - 2.5.5. Verfahren mit Lagrange-Multiplikatoren . . . 51
  - 2.5.6. Verfahren in reduzierten Koordinaten . . . 64
- 2.6. Animation der Bewegungsgleichungen . . . 67
- 2.7. Verfahren zur Fehlerkorrektur . . . 70
  - 2.7.1. Stabilisierung von Zwangsbedingungen . . . 70
  - 2.7.2. Fehlerkorrektur durch Projektion . . . 72

3. Inverse Dynamik mit Lagrange-Multiplikatoren 75
3.1. Einheitliche Formulierung von Bewegungsgleichungen und Zwangsbedingungen . . . 76
3.1.1. Die Bewegungsgleichungen nach Newton und Euler . . . 76
3.1.2. Integration bilateraler geschwindigkeitsbezogener Zwangsbedingungen . . . 79
3.1.3. Integration unilateraler Zwangsbedingungen - reibungsfreie Kontakte . . . 81
3.1.4. Integration unilateraler Zwangsbedingungen - Berücksichtigung Coulombscher Kontaktreibung . . . 82
3.1.5. Stabilisierung von Zwangsbedingungen . . . 87
3.2. Geschwindigkeitsbezogene Zwangsbedingungen für klassische Gelenke 89
3.2.1. Bestimmung von Position und Orientierung von Gelenken zur Simulationslaufzeit . . . 89
3.2.2. Einheitliche Schreibweise für Zwangsbedingungen . . . 90
3.2.3. Das Kugelgelenk . . . 92
3.2.4. Das Drehgelenk . . . 93
3.2.5. Das Kardan- oder Kreuzgelenk . . . 96
3.2.6. Das Lineargelenk . . . 98
3.2.7. Gelenkanschläge oder -limits . . . 101
3.2.8. Realisierung von Motoren durch Zwangsbedingungen . . . 103
3.3. Geschwindigkeitsbezogene Zwangsbedingungen für Kontakte und Kontaktreibung . . . 105
3.3.1. Formulierung von Kontaktnormalenzwangsbedingungen . . . 105
3.3.2. Formulierung von Kontaktreibungszwangsbedingungen . . . 107
3.4. Zwangsbedingungen für erweiterte mechanische Zusammenhänge . . . 108
3.4.1. Das Schraubengelenk . . . 108
3.4.2. Differenzialbedingungen und Differenzialgetriebe . . . 111
3.4.3. Robuste Realisierung von Feder-Dämpfer-Systemen . . . 114
3.4.4. Realisierung einer Radaufhängung durch ein integriertes Gelenk . 118
3.5. Eingesetzte Verfahren zur Lösung des Komplementaritätsproblems . . . 122
3.5.1. Das projizierte Gauß-Seidel Iterationsverfahren . . . 123
3.5.2. Eine effiziente und problemorientierte Implementierung der Dantzig Routine . . . 125
3.5.3. Vergleich und Bewertung der eingesetzten Lösungsverfahren . . . 134

4. Kontaktgraphenanalyse zur Optimierung von Verfahren und Modellierung 141
4.1. Dynamische Rekonfiguration des Mehrkörpersystems . . . 143
4.2. Automatisierte Bestimmung von Kollisionsgruppen . . . 145
4.3. Bildung von Simulationsclustern . . . 146
4.4. Die Kontaktpfadlänge als Grundlage für Stoßfortpflanzung und Heuristiken 147
4.4.1. Optimierte Adaption des Prinzips der Stoßfortpflanzung . . . 148

4.4.2. Die Kontaktpfadlänge als Entscheidungskriterium in Aktivierungsheuristiken . . . 152
4.5. Softwaretechnische Realisierung . . . 153
4.6. Bewertung der Verfahren der Kontaktgraphenanalyse . . . 155

5. Implementierung in einem VR-System 159
5.1. Struktur der Implementierung . . . 159
5.2. Der zeitliche Ablauf der Simulation . . . 160
5.3. Modellierung dynamikbasierter Simulationen . . . 162
5.3.1. Modellierung eines Starrkörpers . . . 164
5.3.2. Modellierung von Gelenken . . . 165
5.3.3. Modellierung von Antrieben . . . 168
5.3.4. Modellierung von Differenzialgetrieben . . . 168
5.3.5. Modellierung dynamikbezogener Sensoren . . . 172
5.4. Objektorientierte Softwarestrukturen des Systems . . . 174
5.4.1. Klassenstruktur für die Kollisionserkennung . . . 174
5.4.2. Klassenstruktur zur Einbringung von Zwangsbedingungen . . . 176
5.5. Laufzeitoptimale Ausführung notwendiger Matrixmultiplikationen . . . 179

6. Anwendungen 183
6.1. Harvestersimulatoren für die Maschinenführerausbildung . . . 183
6.1.1. Anwendungsspezifische Anforderungen . . . 185
6.1.2. Anwendungsspezifische Verfahren . . . 186
6.1.3. Anwendung generischer Verfahren und Modellierung . . . 190
6.1.4. Erfahrungen und Ergebnisse . . . 193
6.2. Forwardersimulatoren für die Maschinenführerausbildung . . . 197
6.2.1. Anwendungsspezifische Anforderungen . . . 197
6.2.2. Anwendungsspezifische Verfahren . . . 198
6.2.3. Anwendung generischer Verfahren und Modellierung . . . 200
6.2.4. Erfahrungen und Ergebnisse . . . 200
6.3. Radlader- und Schüttgutsimulation . . . 202
6.3.1. Anwendungsspezifische Anforderungen . . . 203
6.3.2. Anwendungsspezifische Verfahren . . . 203
6.3.3. Anwendung generischer Verfahren und Modellierung . . . 206
6.3.4. Erfahrungen und Ergebnisse . . . 207
6.4. Entwicklung eines Virtuellen Testbeds zur Evaluierung von Laufrobotern . . . 208
6.4.1. Anwendungsspezifische Anforderungen . . . 209
6.4.2. Anwendungsspezifische Verfahren . . . 210
6.4.3. Anwendung generischer Verfahren und Modellierung . . . 219
6.4.4. Experimente im Virtuellen Testbed . . . 219
6.4.5. Erfahrungen und Ergebnisse . . . 222
6.5. Weitere realisierte Anwendungen . . . 222
6.5.1. Steuerungsentwicklung in der Weltraumrobotik . . . 223

6.5.2. Fahrzeuganimation in einem 3D-Geoinformationssystem (3D-GIS) 225

7. Zusammenfassung der Ergebnisse und Ausblick 229
7.1. Zusammenfassung der Ergebnisse 229
7.2. Ausblick 231

A. Mathematische und physikalische Grundlagen 233
A.1. Eigenschaften des Trägheitstensors 233
A.1.1. Koordinatentransformation des Trägheitstensors 234
A.1.2. Akkumulation von Trägheitstensoren 235
A.2. Das Coulombsche Reibungsmodell 236
A.3. Kreuzproduktmatrizen 237
A.4. LDLT-Zerlegung positiv semi-definiter Matrizen 237
A.5. Das Quaternion als Orientierungsdarstellung 238
A.5.1. Quaternionenmultiplikation 238
A.5.2. Konjugiertes Quaternion 239
A.5.3. Rotation eines Vektors 239
A.5.4. Inverse Rotation 239
A.5.5. Zusammenhang Quaternion und Winkelgeschwindigkeit 239

B. Ein analytisches Beispiel: Der SCARA-Roboter 241

C. Jacobimatrix der Zwangsbedingungen und in der Robotik 249

Kapitel 1.

Einleitung

1.1. Motivation

Virtuelle Realität (VR) und virtuelle Welten erhalten Einzug in immer mehr Bereiche von Forschung und Industrie. In kommerziellen Anwendungen werden sie bis heute vorrangig als Visualisierungs- und Präsentationsmedium eingesetzt. Sie dienen dazu, neue Produkte oder Anlagen lange vor ihrer Realisierung erfahren und im Team diskutieren zu können. Das dynamische Verhalten virtueller Gegenstände wird häufig durch Steuerungen nur phänomenologisch nachgebildet, Simulation nur zur detaillierten Betrachtung einzelner Aspekte bzw. einzelner „Zielobjekte“ eingesetzt.

Die Entwicklung immer leistungsfähigerer Hard- und Software erlaubt heute, neben dem eigentlichen Zielobjekt auch seinen späteren Einsatzbereich in Form komplexer Umgebungen zu simulieren. Erst hierdurch können realitätsnahe Anwendungsszenarien und damit realitätsnahe Anforderungsprofile in der Simulation erzeugt werden. Autonome mobile Roboter sind ein gutes Beispiel hierfür. Sie sind ein intensiv untersuchtes Thema, weil ihnen ein sehr breites Anwendungsspektrum vorausgesagt wird: Als Service- und Explorationsroboter in der Weltraumforschung, als Transportsystem in der innerbetrieblichen Logistik, als Rettungskräfte in Katastrophenszenarien oder als autonome fliegende Plattformen für Überwachungsaufgaben oder für die luftgestützte Bilddatengewinnung. In diesen Anwendungen ist offensichtlich, dass der Einsatz von Simulation zur Planung, Entwicklung und Erprobung solcher Systeme nur dann erfolgversprechend sein kann, wenn neben dem untersuchten System auch die zugehörigen Anwendungsszenarien realitätsnah abgebildet werden. Diese beinhalten jedoch typischerweise große Ungewissheiten. Vollständige formale Beschreibungen sind kaum möglich. Erst ein ganzheitlicher Simulationsansatz, der alle Teile eines möglichen An-

wendungsszenarios simuliert, ohne dabei nur deren Auswirkungen auf das untersuchte System in Form bekannter Effekte vorherzusagen, kann hier sinnvoll zur Simulation eingesetzt werden.

Ein anderer wichtiger Anwendungsbereich von Virtueller Realität sind Maschinensimulatoren für Ausbildung und Training von Maschinenführern. Während der Einsatz von Simulatoren in prominenteren Bereichen wie der Pilotenausbildung seit langem üblich ist, rentiert sich ihr Einsatz aufgrund immer komplexer und damit teurer werdender Arbeitsmaschinen und gleichzeitig sinkender Kosten und steigender Leistung von VR-Technologie auch in weniger exklusiven Arbeitsbereichen. So werden Simulatoren seit einigen Jahren sehr erfolgreich für Ausbildung und Training von Forstmaschinenführern eingesetzt. Hiervon angesteckt entsteht auch im Baumaschinenbereich langsam ein entsprechendes Interesse.

Solange die Realisierung solcher Simulationsanwendungen aufgrund fehlender Leistungsfähigkeit der Rechner noch nicht auf Grundlage von physikalischen Modellen möglich war, wurden sie rein kinematisch, zustands- und steuerungsbasiert umgesetzt. Rein kinematisch bedeutet, dass die Simulationsmodelle keine physikalischen Größen wie Massen und Kräfte berücksichtigen, sondern lediglich das Bewegungsverhalten phänomenologisch in Form von Beschleunigungen, Geschwindigkeiten und Positionen nachempfinden. Zustands- und steuerungsbasiert bedeutet, der Apfel löst sich nicht deshalb vom Baum, weil sein Stiel nicht mehr stark genug ist, sein Gewicht zu tragen, sondern weil eine Logik voraussagt, dass er im Herbst abfallen wird. Und er fällt nicht deshalb zu Boden, weil er durch seine Gewichtskraft beschleunigt wird, sondern weil eine zweite Logik besagt, dass er sich im Zustand „Fallend“ mit konstanter Beschleunigung bis zum Boden bewegen wird.

Während sich eine phänomenologische Simulation des Apfelbeispiels noch gut und wahrscheinlich sogar einfacher umsetzen lässt, als eine Simulation auf Grundlage eines physikalischen Modells, steigt der Aufwand für die phänomenologische Realisierung komplexerer Umgebungen überproportional. Gleichzeitig sinkt unweigerlich die Realitätsnähe. Außerdem muss jede Anwendung sehr individuell programmiert werden, Wiederverwendung vorhandenen Codes ist nur in geringem Umfang und für einfache Zusammenhänge möglich.

Die Nutzung physikalischer Modelle als Grundlage für die Anwendungsentwicklung liegt daher nahe. Das Ideal dieser Arbeit sieht vor, dass Anwendungen nicht länger *programmiert*, sondern *modelliert* werden. Anstatt das Bewegungsverhalten des Apfels

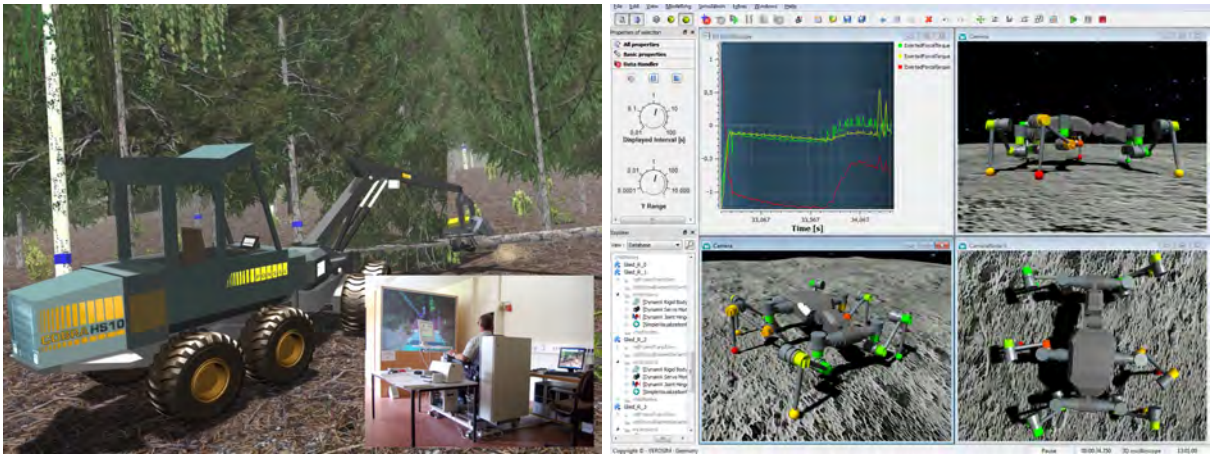


Abbildung 1.1.: Anwendungen von VR-Technologie am Institut für Mensch-Maschine-Interaktion der RWTH Aachen. Links: Forstmaschinensimulator für Ausbildung und Training. Rechts: Virtuelles Testbed für lunare Explorationsroboter.

zu programmieren, werden seine physikalischen Eigenschaften modelliert. Eine Physiksimulation sorgt durch Umsetzung physikalischer Gesetzmäßigkeiten dann für das realitätsnahe Bewegungsverhalten. Ein Simulationssystem kann so als Grundlage für eine Vielzahl von Anwendungen dienen. Von diesem Ideal ist die Realität noch weit entfernt, denn in der Simulationstechnik kommen, je nach fokussierter Anwendung, viele unterschiedliche Modelle für dynamisches Bewegungsverhalten zum Einsatz.

Abbildung 1.1 zeigt beispielhaft zwei der Anwendungen, die am Institut für Mensch-Maschine-Interaktion (MMI) [4] der RWTH Aachen entwickelt werden. Für diese und viele weitere scheint das Modell der Mehr-Starrkörper-Dynamiksimulation jedoch besonders geeignet, eine zentrale Rolle im Reigen der Simulationsmodelle einzunehmen. Ob autonome Roboter, industrielle Automatisierung oder Arbeitsmaschinensimulatoren: In all diesen Bereichen kann die Starrkörperdynamik für einen großen Anteil der dynamischen Effekte eine gute Approximation der Realität bieten. Daher erscheint es sinnvoll, den Nutzen eines Mehrkörperdynamiksimulationssystems als Grundlage unterschiedlichster Anwendungen zu evaluieren.

Veröffentlichungen zum Thema Mehrkörperdynamiksimulation bis in die jüngste Zeit zeigen, dass dieses Thema Gegenstand aktueller Forschung ist. Grundlagen der Starrkörperdynamik werden heute zwar teilweise schon im Grundstudium behandelt. Es handelt sich jedoch um ein stark idealisiertes Modell - jeder reale Körper wird sich unter Krafteinwirkung nicht nur bewegen, sondern auch verformen. Da das Starrkörpermo-

dell diese Verformungen nicht nachbildet, kommt es bei der numerischen Simulation zu Unstetigkeiten, die große Stabilitätsprobleme der Simulationsverfahren nach sich ziehen können.

Sowohl auf kommerzieller wie auf nicht kommerzieller Ebene ist bereits eine große Anzahl von Softwarebibliotheken für die Mehrkörperdynamiksimulation verfügbar. Schon die Tatsache, dass das Starrkörpermodell ein idealisiertes Modell ist, bringt es jedoch mit sich, dass keine der verfügbaren Lösungen für jeden Anwendungsbereich optimal geeignet ist. Die Probleme, die durch die Abweichung zwischen Realität und Starrkörpermodell entstehen, werden von jeder Lösung anders behandelt. Als Entwickler eines Simulationssystems nicht für eine, sondern für eine Menge von Anwendungen ist es jedoch unabdingbar, jedes Detail einer komplexen Simulationsanwendung „in der eigenen Hand“ zu haben, d.h. jedes Verhalten der Simulation verstehen und ggf. anpassen zu können. Greift man für eine zentrale und gleichzeitig so komplexe Softwarekomponente wie eine Mehrkörperdynamiksimulation auf eine bestehende, externe Bibliothek zurück, besteht die große Gefahr, dass aufgrund des fehlenden Verständnisses für innere Vorgänge Anpassungen nicht möglich sind, selbst wenn Zugriff auf den Quellcode besteht.

Deshalb stand am Anfang dieser Arbeit auch die Motivation, eine eigene Simulationsbibliothek für die Mehrkörperdynamik zu entwickeln und dabei aus den Unzulänglichkeiten bekannter Konzepte zu lernen. Zur Evaluation soll ihr Einsatz und ihre Erprobung in realen Anwendungen dienen. Realitätsnahe, praktische Anwendungen stehen deshalb im Vordergrund, weil nur sie geeignet sind, die wirklich notwendigen Anforderungen an ein Simulationssystem zu definieren.

Wichtige Grundlage für die erfolgreiche Realisierung eines Systems mit großem Anwendungsspektrum ist die Integration der Mehrkörperdynamiksimulation in das am MMI mitentwickelte Robotersimulations- und Virtual-Reality-System (VR-System) VEROSIM®. Durch diese Integration kann der Fokus ganz auf der eigentlichen Aufgabe liegen, denn VEROSIM® bietet leistungsfähige Komponenten für die Datenhaltung, die Visualisierung, den Datenaustausch und die Benutzerführung. Es wird stetig auch im Sinne der im Verlauf dieser Arbeit gestellten Anforderungen weiterentwickelt.

1.2. Aufbau der Arbeit

Im Anschluss an das einleitende Kapitel liefert Kapitel 2 einen umfassenden Überblick über den Stand der Technik in all den Themengebieten, die die Mehrkörperdynamiksimulation betreffen. Hierzu werden zunächst Einzelaufgaben während der Simulation identifiziert und anschließend zu jedem Punkt die in der Literatur beschriebenen Verfahren diskutiert. Ein Schwerpunkt liegt auf dem Problem der Bestimmung der inversen Dynamik[1], da dies unbestritten eine der komplexesten Aufgaben im Gesamtthemenbereich darstellt.

Im Verlauf dieser Arbeit hat sich ein Verfahren zur Bestimmung der inversen Dynamik auf Grundlage von Lagrange-Multiplikatoren als geeignet erwiesen und wurde deshalb als Grundlage vieler Anwendungen realisiert. Das dritte Kapitel beschreibt die wesentlichen Details dieses Verfahrens. Hierzu gehören vor allem die mathematischen Grundlagen, die Formulierung und Berücksichtigung klassischer und erweiterter Zwangsbedingungen sowie die Beschreibung und Untersuchung numerischer Lösungsverfahren für das resultierende mathematische Problem.

Kontaktgraphen liefern eine abstrakte Sicht auf ein Simulationsmodell und dienen als Grundlage für heuristische Verfahren zur Optimierung der vorgestellten Simulationsverfahren. Die Idee des Kontaktgraphen wurde im Laufe des Einsatzes der Realisierung vieler Anwendungen verfeinert und als ein eigenes Werkzeug mit vielen Anwendungen umgesetzt. Die Konzepte und Werkzeuge und ihre Realisierungen behandelt Kapitel 4.

Kapitel 5 behandelt praxisnahe Aspekte der Realisierung des Simulationssystems, die vor allem für die spätere Weiterentwicklung eine wertvolle Informationsquelle darstellen. Aber auch neue Konzepte zur Modellierung von Anwendungen mit VEROSIM® auf Basis des realisierten Mehrkörperdynamiksimulationssystems werden hier beschrieben.

Kapitel 6 schließlich bildet neben Kapitel 3 den zweiten Schwerpunkt der Arbeit. Hier werden die Anwendungen und ihre Entwicklung vorgestellt, die auf dem im Rahmen dieser Arbeit entstandenen Mehrkörperdynamiksimulationssystem basieren. Zur deren Realisierung sind auch mit dem hier vorgestellten Mehrkörperdynamiksimulationssystem anwendungsindividuelle Entwicklungen notwendig. Im Verlauf dieses Kapitels wird jedoch deutlich, dass der Anteil solcher anwendungsindividuellen Entwicklungsarbeiten durch die gemeinsame Grundlage der Dynamiksimulation gering gehalten werden

[1] Die inverse Dynamik beantwortet die Frage nach den notwendigen Kräften und Momenten zur Erreichung eines geforderten, realitätsnahen Bewegungsverhaltens

kann. Außerdem werden am konkreten Beispiel von Bodenmechanik- und Schüttgutsimulation Konzepte vorgestellt, die die Verknüpfung anderer Verfahren der Dynamiksimulation mit der Mehrkörperdynamiksimulation erlauben.

Das letzte Kapitel schließlich fasst die Ergebnisse der Arbeit zusammen. Hierzu gehören die systematische Entwicklung eines vielseitigen, effizienten und robusten Mehrkörperdynamiksimulationssystems, Verfahren zu dessen Optimierung im Hinblick auf ein breites Anwendungsspektrum, seine geschickte Einbindung in ein modernes Robotersimulations- und Virtual-Reality-System, die detaillierte Analyse wesentlicher Lösungsverfahren für das formulierte mathematische Problem der inversen Dynamik und moderne Konzepte und Verfahren zur Verbindung von Mehrkörperdynamiksimulation mit anderen Simulationsverfahren. Zum Abschluss erfolgt ein Ausblick auf zukünftige Entwicklungen.

Kapitel 2.

Verfahren der Simulation von Mehrkörpersystemen: Stand der Technik

Obwohl der Begriff Mehrkörperdynamiksimulation prinzipiell auch die Interaktionen nicht starrer Körper abdeckt, wird er doch meist mit rein starren Körpern assoziiert. Dies liegt u.a. an der Tatsache, dass das Modell des starren Körpers gut verstanden und vollständig formal beschrieben ist. Aus Sicht der Anwendungserstellung bietet es außerdem einen ausgezeichneten Kompromiss zwischen dem Aufwand, der für die Modellierung einer dynamikbasierten Simulation notwendig ist und dem Nutzen, den man als Entwickler in Form einer realitätsnahen Simulation erhält. Außerdem sind auch Starrkörpersimulationen nicht „absolut starr“: In gewissen Grenzen lassen sich definierte flexible Elemente realisieren, z.B. als Verbindungsglieder zwischen einzelnen Körpern. Weil sich das Mehr-*Starr*körpermodell als geeignete Grundlage für viele der im Rahmen dieser Arbeit untersuchten Anwendungen erwiesen hat, wird hier von *starren* Mehrkörpersystemen ausgegangen.

Wer heute anfängt, sich dem Thema Mehrkörperdynamiksimulation zu widmen, identifiziert im Wesentlichen zwei Gruppen von Interessenten: Ein großer Teil der Literatur betrachtet das Thema sehr grundlagenorientiert. Hier hat sich die Wendung „Starrkörperdynamik mit Gelenken, Kontakten und Reibung“ (engl.: *Rigid Body Dynamics with Joints, Contacts and Friction*) zu einem Standard-Problem der angewandten Physik entwickelt, für das auf analytischem und häufig sehr formalem Wege neue mathematische Formulierungen und Lösungen gesucht und beschrieben werden. Auf der anderen Seite gibt es eine stetig wachsende Gemeinde von Entwicklern, die dem Thema sehr anwen-

dungsorientiert begegnet. Sie kommen aus der Praxis, haben konkrete Anwendungen und daraus resultierende Anforderungen und diskutieren häufig auf Basis einer der vielen frei verfügbaren Implementierungen wie Bullet [1] oder der Open Dynamics Engine (ODE) [121].

Dieses Kapitel beleuchtet die praktischen und theoretischen Aspekte der Mehrkörperdynamiksimulation ausgehend von der Problemstellung der Anwendungsentwicklung. Da sich bei Einsatz einer der bestehenden Bibliotheken häufig herausstellt, dass eine hiermit erzeugte Simulation nicht das erwartete Verhalten zeigt, kann dieses Kapitel auch in diesem Fall eine nützliche Informationsquelle sein.

Die bis heute bekannten und verbreiteten Verfahren und die zugehörigen Begriffe und Formulierungen der Mehrkörperdynamiksimulation werden geordnet und erläutert. Da diese Arbeit eine ganzheitliche Sicht über das Themengebiet liefern möchte, werden zunächst die einzelnen Aufgaben, die mit diesem Kontext assoziiert werden, separiert und beschrieben. Die dann folgenden Unterkapitel greifen die in der Literatur verwendeten Begriffe aus dem Bereich Mehrkörperdynamiksimulation auf und beschreiben jeweils, welche Verfahren zur Bearbeitung der Teilaufgaben existieren und wie sie funktionieren.

2.1. Teilaufgaben einer Mehrkörperdynamiksimulation

Die im Folgenden aufgeführten Teilaufgaben der Mehrkörperdynamiksimulation werden nicht in jedem Simulationsverfahren separiert behandelt. Dennoch bieten sie eine geeignete thematische Gliederung der nachfolgend beschriebenen Verfahren.

1. ***Die Steuerung des zeitlichen Ablaufs***: Die zeitliche Steuerung ist aus softwaretechnischer Sicht die zentrale Komponente im Gesamtsystem. Sie bestimmt die zeitlichen Schrittweiten, ggf. unter Zuhilfenahme der anderen Komponenten und definiert darüber hinaus die Abarbeitungsreihenfolge der anderen Komponenten.

2. ***Die Kollisionserkennung und Kontaktpunktbestimmung***: Die Kollisionserkennung überprüft die beteiligten Körper auf Kollisionen oder gegenseitige Durchdringung. Für die physikalisch basierte Simulation ist neben der reinen Feststellung einer Kollision die Bestimmung von Kontaktpunkten mit jeweils einer Position, einer Kontaktnormalen und ggf. der Durchdringungstiefe notwendig. Zwischen zwei Körpern können dabei auch mehr als ein einzelner Kontaktpunkt notwendig sein,

wenn z.B. eine stabile Ruhelage von aufeinander gestapelten Körpern erreicht werden soll.

3. ***Die Kollisionsbehandlung***: Stoßen zwei Körper mit einer nicht verschwindenden relativen Geschwindigkeit aufeinander, sorgt die Kollisionsbehandlung für ein Auseinanderdriften der Körper im Fortgang der Simulation. Dies geschieht, indem Kollisionskräfte bestimmt und aufgewendet werden. Im Gegensatz zur Kollision wird der *ruhende Kontakt* durch eine *Zwangsbedingung* berücksichtigt.

4. ***Berücksichtigung von Zwangsbedingungen***: Zwangsbedingungen schränken die möglichen Freiheitsgrade des Mehrkörpersystems ein. Sie werden u.a. durch Kontakte und Gelenke impliziert. Die Berücksichtigung von Zwangsbedingungen ist eine der komplexesten Aufgaben im Kontext der Mehrkörperdynamiksimulation und bildet daher einen Schwerpunkt in diesem Kapitel.

   Je nach eingesetztem Verfahren spricht man bei dieser Aufgabe auch von der Bestimmung der *inversen Dynamik*. Diese Formulierung rührt daher, dass ausgehend von einem gewünschten bzw. realitätsnahen Verhalten des Mehrkörpersystems z.B. an einem Gelenk die dafür nötigen Kräfte und Momente bestimmt werden. Die *Vorwärtsdynamik* im Gegensatz hierzu bestimmt das Bewegungsverhalten des Mehrkörpersystems, wenn alle Kräfte und Momente bekannt sind, und entspricht damit im Wesentlichen der Animation der Bewegungsgleichungen (s.u.). Diese Komponente wird im Folgenden mit dem englischen Fachbegriff *Constraint-Solver* bezeichnet.

5. ***Animation der Bewegungsgleichungen***: Sind die Bewegungsgleichungen sowie alle wirkenden Kräfte und Momente des Mehrkörpersystems bekannt, folgt die Animation, d.h. die numerische Integration der Bewegungsgleichungen. Hieraus resultieren neue Geschwindigkeiten, Positionen und Orientierungen der Körper im System. Diese Komponente wird im Folgenden mit dem englischen Fachbegriff *Motion-Solver* bezeichnet.

6. ***Die Fehlerkorrektur***: Während der Bestimmung der inversen Dynamik können Positions- und Orientierungszwangsbedingungen nicht immer direkt berücksichtigt werden, sondern nur durch entsprechende Derivate, d.h. Zwangsbedingungen mit Bezug auf Geschwindigkeiten oder Beschleunigungen. Kleine Fehler während der Simulation, hervorgerufen u.a. durch die numerische Integration, führen zu

Verletzungen der ursprünglichen Zwangsbedingungen, die durch eine Fehlerkorrektur beseitigt werden müssen.

2.2. Steuerung des zeitlichen Ablaufs

Die numerische Dynamiksimulation erfordert die Diskretisierung der Zeit. Die Zeitsteuerung ist damit die zentrale Komponente der zeitdiskreten Dynamiksimulation. Sie überwacht den Fortschritt der Simulationszeit und steuert die anderen Komponenten der Simulation, die Kollisionserkennung und -behandlung, den Constraint-Solver, den Motion-Solver und die Fehlerkorrektur an.

Im Kontext realzeitfähiger oder interaktiver Systeme ist eine möglichst große Zeitschrittweite wünschenswert. Sie wird durch unterschiedliche Aspekte begrenzt: Werden im Mehrkörpersystem Kontakte und Kollisionen berücksichtigt, sollte ein Zeitschritt möglichst immer nur bis zum nächsten Erscheinen oder Verschwinden eines Kontaktes andauern. Ab diesem Zeitpunkt verändert sich die Konstellation auftretender Kräfte und Momente. In Mehrkörpersystemen mit Maximalkoordinatenansatz (vgl. Kapitel 2.5.1 auf Seite 43) müssen nicht nur in Kontakten, sondern auch in Lagern und Gelenken explizit Zwangskräfte bestimmt werden. Da diese im Allgemeinen nicht konstant sind, sondern in Abhängigkeit von Positionen und Geschwindigkeiten im Mehrkörpersystem stark variieren, muss die Zeitschrittweite klein genug sein, um die gerade wirkenden Zwangskräfte stetig bestimmen und berücksichtigen zu können. Zuletzt muss der unvermeidliche Fehler numerischer Integrationsverfahren berücksichtigt werden, der mit der Größe der Zeitschrittweite ansteigt.

Grundsätzlich unterscheidet man die Zeitsteuerungsverfahren danach, ob sie mit variablen oder mit konstanten Zeitschrittweiten arbeiten. Zeitsteuerungen mit variablen Schrittweiten nutzen die Kollisionserkennung und den Motion-Solver, um ausgehend von einem gültigen Zustand der Mehrkörpersystems den nächsten Zeitpunkt festzustellen, zu dem eine neue Kollision auftritt. Der Motion-Solver führt die Bewegungssimulation bis zu diesem Zeitpunkt aus. Dort werden die Kontaktpunkte bestimmt, so dass die durch sie verursachten Kräfte für den nächsten Zeitschritt berücksichtigt werden können.

Die Alternative zur Bestimmung einer Zeitschrittweite zur Laufzeit der Simulation sind konstante Schrittweiten. Hier wird gänzlich auf die Bestimmung einer optimalen Zeitschrittweite verzichtet und stattdessen eine konstante Schrittweite $h = \Delta t$ angesetzt.

2.2.1. Variable Schrittweiten mit Zeitschritt-Unterteilungsverfahren

Ein einfacher Ansatz, den Zeitpunkt t_C der nächsten Kollision zu bestimmen, besteht in einem Unterteilungs- (engl.: *Backtracking*-)-Verfahren [9, 17, 89]. Ausgehend von einem Zeitpunkt t_0 wird die Zeit um einen konstanten Wert h fortgeschrieben und der Motion-Solver animiert die Bewegungsgleichungen bis zum neuen Zeitpunkt $t_n = t_0 + h$. Stellt die Kollisionserkennung im neuen Zeitpunkt t_n Durchdringungen der Körper fest, lag der echte Kollisionszeitpunkt vor t_n. Dann muss die Simulation zum Zeitpunkt t_0 zurückkehren und einen kleineren Zeitschritt versuchen. Bei Einsatz eines Bisektionsverfahrens wählt man $t_n = t_0 + h/2$ und wiederholt den Vorgang. Stellt die Kollisionserkennung keine oder nur geringe Durchdringungen fest, so ist der Zeitpunkt der nächsten Kollision ausreichend genau gefunden.

2.2.2. Variable Schrittweiten mit dem Einseiten-Verfahren

Beim Einseiten-Verfahren [91, 88] (engl.: *Conservative Advancement*) werden zunächst alle potentiellen Kollisionspaarungen nach ihrem voraussichtlichen Kollisionszeitpunkt (engl.: *Time of Impact, TOI*) in einer Liste sortiert, so dass das Paar, dessen Kollision als nächstes auftritt, an erster Position steht. Die Schätzung des Kollisionszeitpunktes geschieht dabei *konservativ*, der geschätzte Zeitpunkt liegt also in jedem Fall vor dem tatsächlichen Zeitpunkt. Die Bewegungsgleichungen können nun sicher, d.h. ohne, dass eine Kollision berücksichtigt werden müsste, bis zu diesem Zeitpunkt animiert werden.

Im Anschluss wird auf Grundlage des Lin/Canny „Closest-Features-Algorithmus“ [79] der Abstand des ersten Kollisionspaares bestimmt. Liegt er noch über einem Grenzwert ϵ, wird eine neue konservative Schätzung des Kollisionszeitpunktes bestimmt. Das gerade betrachtete Paar wird ggf. neu in die Liste der Kollisionskandidaten einsortiert. Jetzt können die Bewegungsgleichungen aller Körper bis zum neuen nächsten Kollisionszeitpunkt animiert werden.

Liegt der Abstand unter dem Grenzwert ϵ, wird eine Kollision deklariert und durch Aufbringen der aus ihr resultierenden Kräfte behandelt. Die geschätzten Kollisionszeitpunkte aller betroffenen Kollisionskandidaten müssen nun erneut bestimmt werden, da sie sich durch die eingebrachten Kräfte verändert haben können.

2.2.3. Variable Schrittweiten mit dem Timewarp-Algorithmus

Eine weitere Variante der zeitlichen Steuerung beschreibt Mirtich in [90]. Er wendet Jeffersons Timewarp-Algorithmus [68] auf die Simulation von Starrkörpern an. Ziel ist es, eine hohe Parallelisierbarkeit der Simulation einzelner Körper zu erreichen: Jeder einzelne Starrkörper und jedes gelenkgekoppelte System wird in einem eigenen Prozess simuliert. Der Algorithmus minimiert die Anzahl notwendiger Synchronisierungen zwischen den Prozessen. Viele Körper können unabhängig von den anderen simuliert werden. Nur wenn tatsächlich Kollisionen oder Kontakte zwischen ihnen auftreten, müssen die nebenläufigen Prozesse synchronisiert werden. In den einzelnen Prozessen sind u.U. sehr große Zeitschrittweiten möglich.

Nachteilig ist: Tritt eine Kollision auf, müssen die beteiligten Prozesse u.U. um längere Zeitabschnitte „zurückgedreht“ werden. Dieses Verfahren scheint nur für sehr spezielle Anwendungen einsetzbar zu sein: Der Vorteil der Parallelität wirkt nur in Konfigurationen, die viele unabhängige Körpergruppen besitzen und die darüber hinaus nur selten interagieren. Andernfalls wiegt der Nachteil des notwendigen „Zurückdrehens“ den Vorteil der Parallelität schnell auf.

2.2.4. Zeitliche Steuerungen mit konstanter Schrittweite

Bei einer Diskretisierung mit konstanter Schrittweite spricht man von „*Time-Stepping*“-Verfahren. Ausgehend von einem gültigen Zustand der Mehrkörpersimulation zum Zeitpunkt t wird die Simulationszeit um einen konstanten Wert $h = \Delta t$ fortgeschrieben. Offensichtlicher Vorteil dieses Vorgehens ist das vollständige Entfallen der aufwändigen Bestimmung der Zeitschrittweite. Im Gegenzug führt die feste Schrittweite auch zu Problemen: So werden bei weitgehend unveränderten Bedingungen möglicherweise unnötig kleine und viele Zeitschritte durchgeführt. Weiterhin können relativ große Durchdringungen in Kontaktpunkten auftreten. Abbildung 2.1 verdeutlicht dieses Problem: Während zum Zeitpunkt t noch keine Kollision stattfindet und entsprechend auch nicht berücksichtigt werden kann, durchdringen sich beide Körper zum Zeitpunkt $t + h$ bereits deutlich.

Eine Möglichkeit, solche Durchdringungen auch bei konstanten Schrittweiten zu verhindern, besteht in der Implementierung eines *Backtracking*-Algorithmus [126, 43]. Zu Beginn eines neuen Zeitschrittes zum Zeitpunkt t wird zunächst die Kollisionserkennung durchgeführt. Alle gefundenen Kontakte werden in einer Menge S_C vermerkt. Die

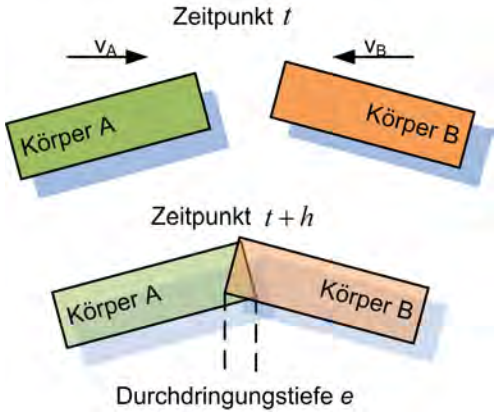


Abbildung 2.1.: 2D Darstellung der Problems der Durchdringung bei konstanter Zeitschrittweite

resultierenden Kontaktkräfte werden in das System eingebracht und die Bewegungsgleichungen unter Berücksichtigung der Kontaktkräfte über den Zeitschritt $h = \Delta t$ hinweg animiert. Am Ende des Zeitschrittes, im Zeitpunkt $t + h$, wird erneut eine Kollisionserkennung durchgeführt. Treten keine neuen Kontakte auf, kann mit dem nächsten Zeitschritt begonnen werden. Sind jedoch neue Kontakte hinzugekommen, die im Zeitpunkt t noch nicht auftraten, werden diese der Menge der Kontakte S_C hinzugefügt und der Animationsschritt von $t \rightarrow t+h$ wird erneut durchgeführt, dieses Mal auch unter Berücksichtigung der zukünftig auftretenden Kontakte. Das Ablaufdiagramm in Abbildung 2.2 verdeutlicht den Vorgang. Auf diesem Weg können Durchdringungen zwar verhindert werden, durch die Berücksichtigung von erst in Zukunft auftretenden Kontakten stellen sich allerdings andere, ungewollte Effekte ein, z.B. das „Schweben“ von Körpern in geringer Höhe oder das Abprallen *kurz vor* einer Wand wie von Geisterhand.

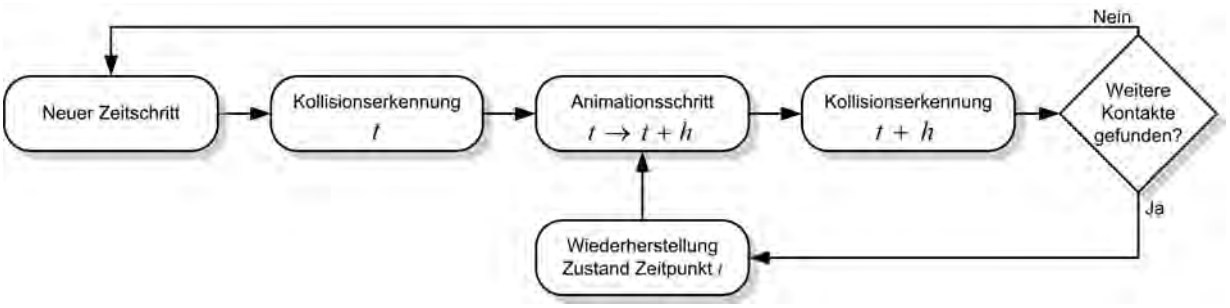


Abbildung 2.2.: Verfahren zur Vermeidung von Durchdringungen durch Detektion zukünftiger Kontakte bei konstanter Zeitschrittweite

Auch lässt sich über das Laufzeitverhalten dieses Algorithmus nur schwer eine Aussage treffen. Zwar ist sicher, dass er terminieren wird, da die Anzahl der Kontakte in jedem Fall endlich ist. Wie häufig er jedoch iterieren muss, kann sehr stark variieren. Darüber hinaus muss gerade die rechenzeitintensive Kollisionserkennung mehrfach pro Simulationsschritt ausgeführt werden, was einen negativen Einfluss auf das Gesamtlaufzeitverhalten hat.

Neben dem Problem der Durchdringungen verursachen Zeitschritte konstanter Länge eine weitere Ungenauigkeit der Simulation: Der Zeitpunkt jeder Kollision wird nach hinten - bzw. beim Backtracking nach vorne - in der Zeit verschoben, da Kontakte nur zu den äquidistanten, diskreten Zeitpunkten der Simulation detektiert und berücksichtigt werden können. Ebenso wie die unvermeidlichen Fehler durch numerische Integrationsverfahren konvergieren beide Fehler jedoch mit kleiner werdendem $h = \Delta t$ gegen Null. Für Anwendungen mit hohem Anspruch an die Genauigkeit sind diese Effekte aber zu berücksichtigen.

2.3. Kollisionserkennung und Bestimmung von Kontaktpunkten

Das Erkennen von Kollisionen zwischen geometrischen Elementen ist eine zentrale Aufgabe für die Dynamiksimulation. Sie ist besonders rechenintensiv und liegt damit auf dem kritischen Pfad im Hinblick auf das Laufzeitverhalten einer Simulation. Es werden Kollisionserkennung und Kontaktpunktbestimmung unterschieden: Ergebnis der Kollisionserkennung ist die Information, ob zwei Körper miteinander kollidieren und ggf. welchen minimalen Abstand sie voneinander haben. Für viele Anwendungen der Virtuellen Realität, z.B. die Bahnplanung von Industrierobotern, ist dies eine hinreichende Information [102, 117]. Für die physikalische Simulation, die u.a. der *Kollisions-* und *Kontaktbehandlung* dient, ist zusätzlich die Bestimmung einer genauen Kontaktgeometrie erforderlich: Hierzu gehören mindestens ein Paar Kontaktpunkte, an denen Kontaktkräfte wirken, die zugehörigen Kontaktnormalen, welche die Richtung der Kontaktkräfte definieren sowie ggf. die Durchdringungstiefe.

Ein naiver Ansatz zur Erkennung aller Kollisionen zwischen n Körpern überprüft alle möglichen Paarungen von Körpern detailliert auf Kollision, was $n(n-1)/2$ Vergleiche erfordert. Um dieses quadratische Laufzeitverhalten zu umgehen, besteht eine Kollisi-

onserkennung typischerweise aus unterschiedlichen Phasen, durch deren Einsatz eine signifikante Reduzierung der Gesamtlaufzeit erreicht wird. Eine sinnfällige Unterteilung besteht aus drei Stufen:

1. ***Grobphase***: Im ersten Schritt, der Grobphase (engl.: *Broad Phase*), wird die Anzahl potenziell kollidierender Körperpaare stark reduziert. Dies verringert die Anzahl notwendiger Vergleiche in den nachfolgenden Phasen.

2. ***Mittelphase***: Der Begriff Mittelphase (engl.: *Mid-Phase*) der Kollisionserkennung ist nicht so weit verbreitet wie die Begriffe Grob- und Detailphase (s.u.) und zudem unterschiedlich belegt. Daher sollen die unterschiedlichen Deutungen an dieser Stelle nur kurz benannt werden. So wird manchmal die Identifikation der von der Kollision betroffenen Primitiven in den *BVH*-Verfahren[1] der Detailphase als Mittelphase bezeichnet. An anderer Stelle wird die Zerlegung von nicht konvexen Polyedern in eine Menge konvexer Polyeder, die einige Verfahren der Detailphase erfordern, als Mittelphase bezeichnet.

3. ***Detailphase***: In der Detailphase (engl.: *Narrow Phase*) schließlich wird die exakte Kollisionserkennung und Kontaktpunktbestimmung durchgeführt.

Ein phasenübergreifendes Werkzeug in der Kollisionserkennung ist die Einführung primitiver Hüllkörper (engl.: *Bounding Volumes*). Hier kommen meist achsenparallele Boxen (engl.: *Axis-Aligned Bounding Box, AABB*), orientierte Boxen (engl.: *Oriented Bounding Box, OBB*) oder Kugeln zum Einsatz. Ein solcher Hüllkörper umschließt vollständig eine gegebene Geometrie. Primitive Hüllkörper verbinden zwei wesentliche Eigenschaften: Kollisionen zwischen zwei Instanzen ihrer Art lassen sich sehr schnell ausschließen und ihre Ausmaße lassen sich mit vertretbarem Mehraufwand bestimmen. Zwischen zwei achsenparallelen Boxen z.B. lässt sich eine Kollision mit maximal sechs skalaren Vergleichen ausschließen, im günstigsten Fall sogar mit einem einzigen.

2.3.1. Verfahren in der Grobphase

Im Bereich der interaktiven Dynamiksimulation kommen während der Grobphase vor allem zwei Verfahren zum Einsatz: Zum einen unterschiedliche Varianten der Raumunterteilung [78, 84] (engl.: *Space Partitioning* oder *Spatial Subdivision*), zum anderen das sogenannte „Sweep-and-Prune“ [10, 36] (früher auch „Sort and Sweep“ genannt).

[1] „Bounding Volume Hierarchies“, vgl. Kapitel 2.3.2 auf Seite 29

Raumunterteilungs-Verfahren

Die einfachste Variante der Raumunterteilung teilt den Gesamtraum einer Szene durch ein Netz mit festgelegter Zellgröße in mehrere Unterräume auf (engl.: *Uniform Subdivision*) [48], vgl. Abbildung 2.3. Jeder Körper der Szene wird allen Unterräumen zugeordnet, die er bzw. sein primitiver Hüllkörper (z.B. AABB) schneidet. Je nach Anwendung wird die Zerlegung in allen drei Raumrichtungen (Bsp.: Asteroidenfeld) oder nur in der Ebene (Bsp.: Billardtisch) durchgeführt. Die festgelegte Zellengröße hat den Vorteil, dass eine einfache Zuordnung von Körpern zu den von ihnen berührten Zellen möglich ist. Ändert ein Körper seine Position, muss nur seine eigene Zellenzuordnung aktualisiert werden. Daher bedeutet die Neuzuordnung nur geringen Mehraufwand. Auf der anderen Seite ist das Finden einer optimalen Zellengröße ein nicht triviales Optimierungsproblem: Eine zu kleine Zellengröße resultiert in einer großen Anzahl Zellen, von denen viele ganz leer sind oder nur einen Körper beinhalten. Zu große Zellen hingegen werden von vielen Körpern berührt, so dass der Vorteil der Partitionierung verloren geht.

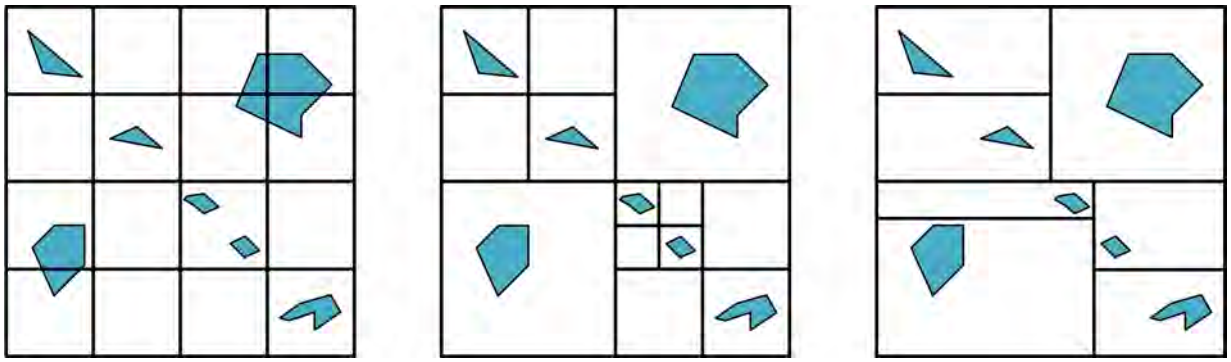


Abbildung 2.3.: Verfahren der Raumunterteilung, von links: Gleichmäßiges Gitter, hierarchisches Gitter, binäre Raumunterteilung

Ausgeklügelte Varianten verwenden hierarchische Unterteilungen: Das Verfahren beginnt mit einer groben Unterteilung der gesamten Szene. Enthält eine Zelle sehr viele Körper, wird die einzelne Zelle erneut unterteilt usw.. Durch dieses Vorgehen erhält man eine homogene Aufteilung von Körpern auf Unterräume. Die resultierende Datenstruktur sind *Quad*- (Unterteilung nur in der Ebene) oder *Octrees* (Unterteilung in allen drei Raumrichtungen) [48]. Nachteil ist jedoch, dass die Unterteilung des Gesamtraumes während der Simulation aktualisiert werden muss, sobald sich die Verteilung der Körper im Raum verändert. Auch das Problem mit u.U. vielen leeren Zellen umgeht man

hiermit nicht.

Eine Alternative zur Unterteilung des Raumes in Unterräume mit konstanter Größe sind binäre Unterteilungsverfahren [33, 96] (engl: *Binary Space Partitioning, BSP*). Solche Verfahren unterteilen den Gesamtraum rekursiv mit Trennebenen in je zwei Unterräume, so dass in beiden Unterräumen die gleiche oder ähnliche Anzahl an Körpern enthalten ist. Die Partitionierung der Szene wird durch einen binären Suchbaum repräsentiert. Diese Verfahren erreichen zwar eine homogene Verteilung der Körper auf die Unterräume, jedoch gilt auch hier, dass die Änderung der Position eines Körpers zu einem unausgeglichenen Suchbaum führt, dessen erneuter Ausgleich u.U. sehr rechenintensiv ist. In [83] werden die „halbausgleichenden Binärbäume“ (engl.: *Semi-adjusting BSP-trees*) vorgestellt. Mittels fünf spezieller Operatoren können lokal unausgeglichene Suchbäume effizient wieder ausgeglichen werden. Eine weitere spezielle Variante der binären Raumunterteilung sind kd-Bäume [24, 122]. Analog zu k-DOPs[2] bei den primitiven Hüllkörpern werden bei kd-Bäumen zur Raumunterteilung Ebenen mit einer von k diskreten Orientierungen genutzt, die jedoch frei im Raum ausgerichtet sein können.

Sweep-and-Prune

Erstmalig noch unter dem Namen „Sort and Sweep“ von Baraff [10] vorgestellt und erweitert und implementiert in der Bibliothek *I-Collide* [36], ist es bis heute das meist genutzte Verfahren in der Grobphase der Kollisionserkennung. Es arbeitet wie folgt: Die achsenparallelen Hüllkörperboxen (AABBs) aller Körper werden auf eine dedizierte Raumachse projiziert, so dass jede AABB durch ein Intervall repräsentiert wird. Die Intervallgrenzen werden in einer sortierten Liste L abgelegt. Im Folgenden wird zusätzlich eine Menge O von Körpern und eine Menge P von Kollisionspaar-Kandidaten benötigt. Die Grobphase beginnt mit einem Durchlauf durch die sortierte Intervallliste: Trifft der Algorithmus beim Durchlauf auf den Intervallanfang von Körper i, werden alle Paarungen ($K_i, K_n \in O$) bestehend aus Körper i und allen anderen Körpern der Menge O gebildet und in der Menge P abgelegt. Trifft er auf das Intervallende von Körper i, so wird K_i aus der Menge O entfernt. Ist der Listendurchlauf beendet, enthält die Menge P alle Körper-Paarungen, die potenziell kollidieren und im Detail untersucht werden müssen. Abbildung 2.4 zeigt eine grafische Darstellung des Verfahrens.

Sweep-and-Prune nutzt besonders gut den Effekt der zeitlichen Kohärenz (engl.: *Temporal Coherence*) in einer Simulationsszene: Häufig ist die Annahme gültig, dass

[2] DOP: Discrete Oriented Polytope. Siehe Kapitel 2.3.2, Seite 29

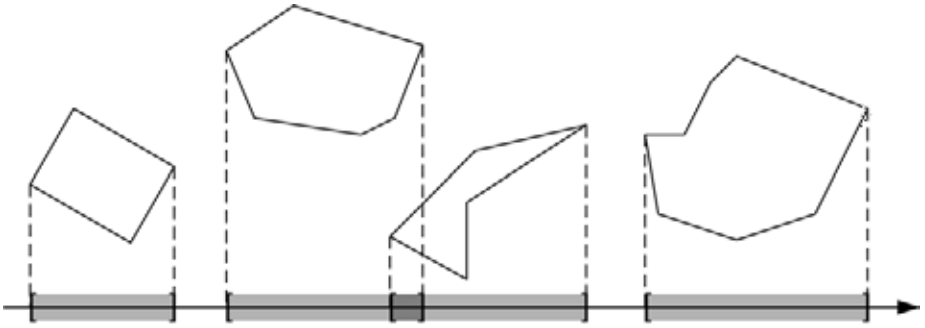


Abbildung 2.4.: Sweep-and-Prune: Die Ausdehnungen der Körper werden auf eine dedizierte Raumachse projiziert. Ein Durchlauf über die sortierten Intervallgrenzen liefert die potenziell kollidierenden Paarungen.

sich die Positionen der Körper zwischen zwei aufeinanderfolgenden Simulationsschritten nur geringfügig ändern. Entsprechend müssen auch nur relativ wenige Aktualisierungen der Intervalllisten vorgenommen werden. Die Intervalllisten müssen darüber hinaus nicht vollständig neu sortiert werden. Hat ein Objekt seine Position verändert, muss es lediglich aus der sortierten Liste entfernt und neu eingefügt werden. Dieser Vorgang ist mit konstanter Zeitkomplexität möglich.

Das Verfahren kann weiter verfeinert werden, indem zwei oder drei Achsen, typischerweise jeweils orthogonal zueinander, zur Projektion der AABBs genutzt werden. *I-Collide* [36] verwaltet zusätzlich zu den drei Intervalllisten je eine boolesche Tabelle, die für jede mögliche Paarung den Status der Überlappung in der betreffenden Richtung vorhält. Hat sich in der Szene ein Körper bewegt, wird er neu in die drei Intervalllisten einsortiert und der Status der betroffenen Paarungen in den drei booleschen Tabellen aktualisiert. Nachfolgende Anfragen können jetzt schnell beantwortet werden.

Obige Ausführungen sind nur ein erster Überblick über die heute gebräuchlichen Verfahren für die Grobphase der Kollisionserkennung. Sie erreichen für eine Vielzahl von Anwendungen der Virtuellen Realität eine völlig ausreichende Leistung, Rocha [101] liefert eine entsprechende Analyse. Tracy et al. [132] kombinieren die Verfahren der Raumunterteilung und des Sweep-and-Prune miteinander und erreichen auf diesem Weg für sehr große Probleme (100.000 Objekte, davon 10.000 bewegt) noch einmal leichte Leistungsgewinne. Ein noch recht junger, vielversprechender Ansatz für noch höhere Leistungswerte ist die Nutzung von massiv parallelen Architekturen mit Streaming Prozessoren, wie sie auf modernen Grafikkarten zur Verfügung stehen. Heyn et al. [60] und Mazhar [84] nutzen eine parallelisierte Version der Raumunterteilung zur effizi-

enten Kollisionserkennung zwischen Millionen von Kugeln. Da ihr Verfahren auf Kugeln als Kollisionskörper beschränkt ist, könnte es beim Einsatz von Kugeln als Hüllkörper in speziellen Anwendungen zur Realisierung einer hochperformanten Grobphase genutzt werden.

2.3.2. Verfahren in der Detailphase

Aus Sicht der Anwendung von Kollisionserkennung in der Dynamiksimulation werden an die Detailphase die höchsten Anforderungen gestellt. Zu einem einzelnen Kontaktpunkt zwischen zwei Körpern ist für die physikalische Simulation die Bestimmung folgender Kontaktpunktattribute notwendig:

- Die Kontaktpunktposition auf jedem der Körper. An diesen Positionen greifen die Kräfte an, die eine weitere Durchdringung der Körper verhindern, eine bestehende Durchdringung korrigieren oder durch Kontaktreibung hervorgerufen werden.
- Die Kollisions- oder Kontaktnormale. Die Kontaktnormale beschreibt die Richtung der Krafteinwirkung, die der Kontaktpunkt hervorruft.
- Die Durchdringungstiefe: Liegt im Kontakt bereits eine Durchdringung vor, besteht in Richtung der Kontaktnormalen die maximale Durchdringungstiefe zwischen den Körpern.

Um eine stabile Ruhelage gewährleisten zu können, müssen zwischen zwei Körpern ggf. mehrere Kontaktpunkte gleichzeitig gefunden werden können. Man spricht in diesem Fall von der Bestimmung einer *Kontaktgeometrie*.

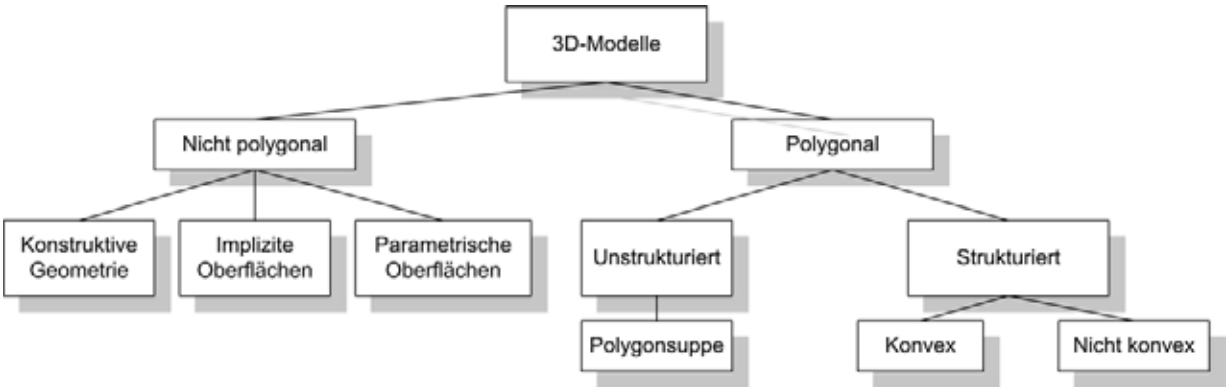


Abbildung 2.5.: Kategorisierung von 3D-Modellen [80]

Die in der Detailphase verwendeten Verfahren sind stark abhängig von den eingesetzten Geometrierepräsentation: Von Lin und Gottschalk [80] stammt die in Abbildung 2.5 dargestellte Kategorisierung. Auf der ersten Ebene werden polygonale und nicht polygonale Geometrien unterschieden. Der Großteil der Literatur im Zusammenhang mit Kollisionserkennung im Bereich Virtueller Realität befasst sich mit polygonalen Modellen. Einerseits ist die polygonale Darstellung eine Grundform, in die alle anderen Darstellungsformen leicht transformiert werden können, andererseits ist die Mehrheit der heute eingesetzten Grafikhard- und -software auf diese Darstellungsvariante hin optimiert, was die Nutzung dieser ohnehin vorhandenen Datenstruktur auch für die physikalische Simulation nahelegt.

Konstruktive Festkörpergeometrie - Primitive geometrische Körper

Im Bereich der nicht-polygonalen Varianten hat vor allem die konstruktive Festkörpergeometrie (engl.: *Constructive Solid Geometry, CSG*) eine nennenswerte Verbreitung bei physikalisch basierter Simulation in Virtueller Realität: Viele Softwarebibliotheken unterstützen geometrische Primitive und Kompositionen hieraus als Kollisionskörper. Typische Repräsentanten solcher Primitiven sind Kugeln, Quader und Kugelzylinder (engl.: *Capped Cylinder* oder *Cylsphere*). Die Kollisionserkennung mit solchen Primitiven ist schnell und exakt und erlaubt „echt runde“ Formen im Gegensatz zur Approximation z.B. einer Kugel durch einen Polyeder. Außerdem ist die genaue Bestimmung aller Kontaktattribute und einer Kontaktgeometrie möglich. Liegt z.B. ein Zylinder eben auf einer Box, so ist es relativ einfach möglich, zwei Kontaktpunkte zu bestimmen, die eine robuste Kontaktgeometrie definieren.

Die Approximation von realen Gegenständen durch Primitive oder Kompositionen aus Primitiven kann jedoch sehr ungenau sein oder aber zur genauen Approximation sind so viele Primitive notwendig, dass der Vorteil der hohen Geschwindigkeit verloren geht. Aus Sicht der Implementierung kommen zwei weitere Nachteile hinzu: Zum einen müssen bei Hinzufügen eines neuen Geometrietyps neue Algorithmen zur Kontaktpunktbestimmung zwischen allen neu entstehenden möglichen Paarungen von Geometrietypen definiert werden. Die Anzahl notwendiger Routinen wächst somit quadratisch mit der Anzahl der unterstützten Primitiven. Zum anderen sind die Algorithmen zur Kontaktpunktbestimmung an geometrischen Primitiven durchaus nicht trivial, vor allem wenn man weitere Geometrietypen als die oben genannten berücksichtigen möchte.

Die in diesem Kapitel noch betrachteten *Sphere-Trees* (vgl. Seite 32) könnte man

ebenfalls als eine automatisierte Variante von konstruktiver Festkörpergeometrie bezeichnen. Hier werden beliebige geometrische Formen durch Mengen von Kugeln approximiert.

Parametrische Flächen

Parametrische Flächen liegen häufig in Form von Höhenfeldern (engl.: *Height Fields*) vor. Bei einem Höhenfeld handelt es sich um eine einfache Abbildung einer Koordinate in der Ebene $(x, y) \in \mathbb{R}^2$ auf einen Punkt einer Oberfläche $(x, y, z) \in \mathbb{R}^3$ bzw. auf einen Höhenwert (2D-Raster mit Höhenwert). Praktisch alle verfügbaren Bibliotheken für die Dynamiksimulation im VR-Umfeld unterstützen diesen Modelltyp. Häufig werden solche Höhenfelder in einem Vorverarbeitungsschritt in Dreiecksnetze (engl.: *Triangle Mesh*) umgewandelt und dann als polygonales Modell behandelt. Selbst wenn dies nicht in einem Vorverarbeitungsschritt geschieht, werden die „Zwischenräume“ zwischen den Rasterpunkten in der Regel linear interpoliert.

Unstrukturierte, polygonale Modelle - Bounding Volume Hierarchies

Bei den polygonalen Geometrien werden strukturierte und unstrukturierte („Polygonsuppen“[3]) Modelle unterschieden. Für unstrukturierte Geometrien kommen vor allem Hierarchische Datenstrukturen aus Hüllkörpern (engl.: *Bounding Volume Hierarchies, BVH*) zum Einsatz. Diese Datenstrukturen dienen dazu, die potenziell von einer Kollision betroffenen Primitiven - zumeist Dreiecke - der beiden Polygonsuppen möglichst schnell zu identifizieren. Dies geschieht analog zur Partitionierung des Raumes in der Grobphase mithilfe einer Partitionierung der einzelnen Geometrien durch hierarchische Unterteilungsverfahren. Bekannte Vertreter solcher Verfahren sind die *OBB-Trees* (*Oriented Bounding Box Trees*) [55], *Boxtrees* [144], *AABB-Trees* [26], *k-DOP-Trees*[4] [75] und *Sphere-Trees* [62].

Aus Sicht der physikalischen Simulation sind BVH-Verfahren nur von begrenztem Nutzen: Wird eine Geometrie, die tatsächlich die Oberfläche eines abgeschlossenen Körpers darstellt, nur als Polygonsuppe, d.h. ohne Information über die zusammenhängende Oberfläche interpretiert, ist die exakte Bestimmung der Kontaktattribute bei einer

[3] Eine Polygonsuppe ist eine unstrukturierte Menge von Polygonen. Sie enthält keine Information darüber, ob die beschriebene Geometrie eine Punktmenge einhüllt.

[4] Ein k-DOP ist ein Hüllkörper, das aus k Ebenen mit diskreten Orientierungen gebildet wird. Eine *Axis Aligned Bounding Box* (AABB) ist ein 6-DOP. Abbildung 2.7 liefert zwei einfache Beispiele.

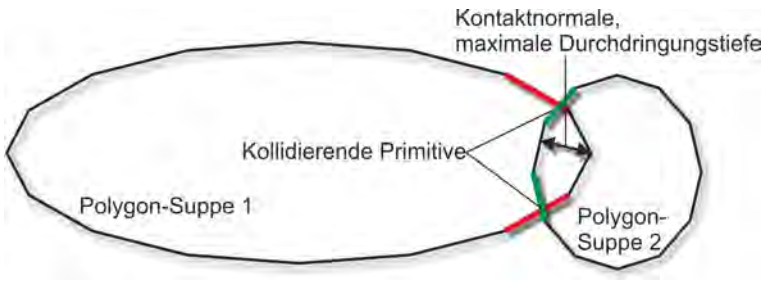


Abbildung 2.6.: Ergebnis einer Mittelphasen-Kollisionserkennung sind die Paare kollidierender Primitiven. Das Beispiel zeigt, dass diese Information für die physikalische Kollisionsbehandlung nicht hinreichend ist.

Kollision beider Körper nicht möglich. Abbildung 2.6 verdeutlicht das Problem: Die Kontakte und zugehörigen Attribute zwischen den direkt kollidierenden Polygonen erlauben noch keine Rückschlüsse auf die tatsächliche Kontaktsituation zwischen den beiden Körpern.

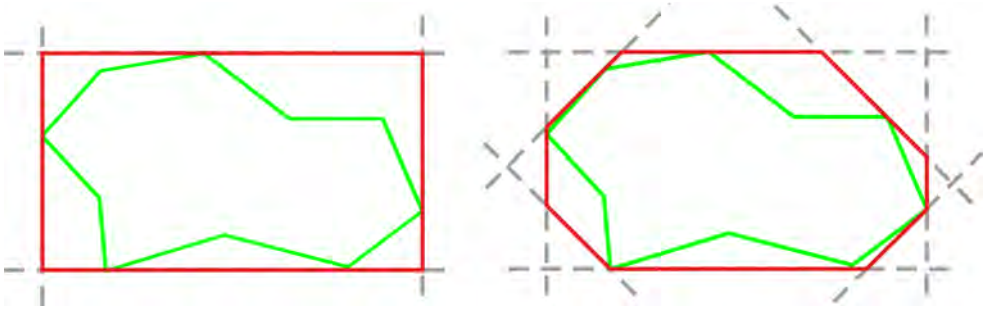


Abbildung 2.7.: k-DOPs in 2D: Links ein 4-DOP, das einer *Axis Aligned Bounding Box* in der Ebene entspricht, rechts ein 8-DOP

Strukturierte, konvexe, polygonale Modelle - Feature-Tracking Algorithmen

Viele Verfahren innerhalb dieser Gruppe sind Weiterentwicklungen des „Closest-Features“-Algorithmus von Lin und Canny [79, 78]. Die *Features* eines Polyeders sind seine Knoten, Kanten und Flächen. Der Algorithmus nutzt den Effekt der zeitlichen Kohärenz: In zwei aufeinander folgenden Zeitschritten bewegt sich das Paar der nächstliegenden Features zweier Polyeder nur geringfügig relativ zueinander. Das nutzt der Algorithmus aus, indem er vom zuletzt nächstliegenden Paar ausgeht und dann entlang der Oberflächen der Polyeder das aktuell nächstliegende Paar „anläuft“. Die Konvexität der Polyeder stellt sicher, dass das lokal gefundene nächstliegende Paar auch global das

nächstliegende ist. Unterschreitet die Distanz der Features des nächstliegenden Paares einen Schwellwert, wird eine Kollision detektiert. Der „Closest-Features“-Algorithmus kann noch nicht mit Durchdringungen umgehen und ist daher für Anwendungen in der Mehrkörpersimulation in Virtueller Realität nicht geeignet. Mirtichs V-Clip [87] ist eine entsprechende Weiterentwicklung. V-Clip verfolgt ebenfalls die nächstliegenden Features, berücksichtigt jedoch auch mögliche Durchdringungen. Zu Gunsten der Stabilität verzichtet er hingegen auf die Fähigkeit, solche Fälle zu identifizieren, in denen die nächstliegenden Features Fläche und Fläche oder Fläche und Kante sind. Durch dieses Vorgehen liefert er eine etwas ungünstigere Ausgangssituation zur Bestimmung einer genauen Kontaktgeometrie.

Konvexe Modelle - Simplex-basierte Algorithmen

Weitere nennenswerte Verfahren in der Detailphase der Kollisionserkennung sind die Simplex-basierten[5] Verfahren auf Grundlage des GJK-Algorithmus (**G**ilbert-**J**ohnson-**K**eerthi) [53]. Ausgereifte Implementierungen wurden vor allem durch van den Bergen [27, 28] präsentiert. Die Bibliothek SOLID [25] basiert auf diesem Verfahren und auch die Dynamiksimulationsbibliothek Bullet [1] bietet eine entsprechende Implementierung. Grundlegende Idee des GJK-Algorithmus ist folgender Zusammenhang: Enthält die Minkowski-Differenz zweier konvexer Hüllen (z.B. konvexe Polyeder) den Ursprung des Koordinatensystems, durchdringen sich beide Hüllen. Die Minkowski-Differenz $A+(-B)$ zweier konvexer Hüllen A und B ist die Menge der Vektoren $\vec{c} = \vec{a} + (-\vec{b})$ mit $\vec{a} \in A$ und $\vec{b} \in B$. Abbildung 2.8 verdeutlicht diesen Zusammenhang an einem einfachen Beispiel.

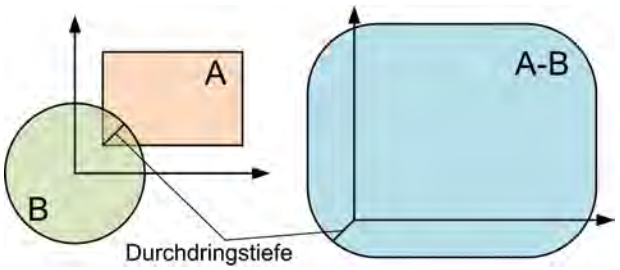


Abbildung 2.8.: Grundlage für den GJK-Algorithmus: Enthält die Minkowski-Differenz $A+(-B)$ der konvexen Hüllen A und B den Ursprung, kollidieren beide Hüllen.

[5] Ein Simplex ist ein n-dimensionales Polytop bestehend aus $n+1$ Eckpunkten. Ein 2D-Simplex ist ein Dreieck, ein 3D-Simplex ein Tetrahedron.

Der Algorithmus arbeitet iterativ. Es werden nacheinander Simplexe aus Punkten der Minkowski-Summe gebildet, die sich dem Ursprung so weit annähern, bis schließlich keine weitere Annäherung mehr möglich ist. Die Distanz des zuletzt gefundenen Simplex zum Ursprung entspricht gleichzeitig dem Abstand bzw. - falls der Ursprung Teil des letzten Simplex ist - der Durchdringungstiefe der beiden Polyeder. Besonders interessant an GJK ist, dass er nicht auf Polyeder beschränkt ist. Er unterstützt alle konvexen Geometrien, für die sich eine sogenannte *Support-Mapping*-Funktion definieren lässt. Eine Support-Mapping-Funktion eines konvexen Objektes A ist eine Funktion $s_A : \mathbb{R}^3 \rightarrow \mathbb{R}^3$, die einen beliebigen Vektor $\vec{v} \in \mathbb{R}^3$ auf einen Punkt des Objektes abbildet:

$$s_A(\vec{v}) \in A, \text{ so dass gilt: } \quad \vec{v} \cdot s_A(\vec{v}) = max\,(\vec{v} \cdot \vec{x} : \vec{x} \in A) \tag{2.1}$$

Sie liefert damit einen Punkt des Objektes, der in Richtung $\vec{v}$ am weitesten vom Ursprung entfernt ist.

Dadurch erlaubt GJK auch die Kollisionserkennung zwischen Geometrien unterschiedlichen Typs, z.B. zwischen einem primitiven Zylinder und einem konvexen Polyeder. Diese enorme Flexibilität gegenüber anderen, auf einen Geometrietyp beschränkten Algorithmen macht ihn für die Anwendung in einem vielseitigen Simulationssystem besonders interessant. Van den Bergen [27] demonstriert, wie Support Mapping Funktionen für die wichtigsten Geometrie-Primitiven, nämlich Kugel, Box, Zylinder und Kegel, definiert werden müssen.

Strukturierte, nicht konvexe polygonale Modelle

Für die Kontaktpunktbestimmung an allgemeinen (konvex und nicht konvex), strukturierten, polygonalen Modellen finden zwei approximierende Verfahren Anwendung: *Signed Distance Maps* [43, 57, 69] und *Sphere-Trees* [62, 63, 98, 29, 30]. *Signed Distance Maps* sind dreidimensionale Gitterraster, die einen Polyeder vollständig umschließen. In jeder Zelle des Rasters wird der Abstand zur Oberfläche des Polyeders abgespeichert. Entsprechend gibt das Vorzeichen des Wertes Auskunft darüber, ob die Zelle innerhalb oder außerhalb des Polyeders liegt. Zusätzlich wird für jeden Polyeder ein Satz von Testpunkten auf der Oberfläche benötigt, dies können z.B. seine Eckpunkte sein.

Zur Kollisionserkennung kann nun für die Testpunkte eines Polyeders ihr Abstand zur

Oberfläche des anderen Polyeders aus dessen *Distance Map* abgelesen werden. Liegt ein Punkt eines Polyeders innerhalb des anderen, kann er als Kontaktpunkt genutzt werden, der Gradient der *Distance Map* dient dann als Kontaktnormale [43]. Alternativ kann auch eine Strahlverfolgung (engl.: *Raytracing*) entlang des Gradienten eingesetzt werden, um den genauen Schnittpunkt an der Oberfläche sowie die exakte Oberflächennormale zu bestimmen [57].

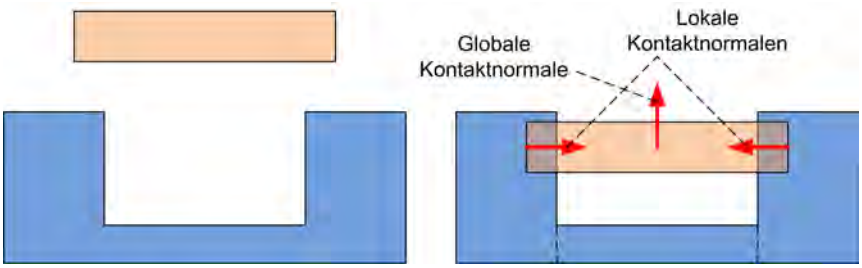


Abbildung 2.9.: Beispiel für das Problem der Bestimmung globaler Kontaktattribute an nicht konvexen Polyedern durch Zerlegung in konvexe Komponenten

Die schon für Polygonsuppen eingesetzten *Sphere-Trees* lassen sich auch in der approximierenden Kontaktpunktbestimmung für strukturierte Modelle einsetzen [29, 30, 62, 63]. Wie bei hierarchischen Datenstrukturen aus Hüllkörpern in der Mittelphase (vgl. Seite 29) wird eine Hierarchie aus Ersatzgeometrien - Kugeln - aufgebaut, welche die eigentliche Geometrie mit zunehmender Tiefe immer besser annähert. Die Wurzel der Hierarchie ist dabei eine die Geometrie vollständig einschließende Kugel. Während die BVH-Verfahren bei Polygonsuppen jedoch nur dazu dienen, die betroffenen Primitiven (Polygone) der Polyeder zu identifizieren, werden die Kugeln der Sphere-Trees zusätzlich direkt zur Kontaktpunktbestimmung herangezogen [98]. Besonders interessant an Sphere-Tree-Verfahren ist die Möglichkeit, die Genauigkeit der Approximation in Abhängigkeit von der verfügbaren Rechenzeit variieren zu können, derart eingesetzt z.B. durch Weller und Zachmann [141, 140]. Der Algorithmus beginnt mit den Wurzelelementen eines Kollisionspaares, den beiden einhüllenden Kugeln. Findet hier keine Kollision statt, bricht der Algorithmus ab. Findet eine Kollision statt und ist noch Rechenzeit verfügbar, werden die Kugeln der nächst tieferen Ebene beider Hierarchien auf Kollision überprüft. Steht keine weitere Rechenzeit zur Verfügung, werden die aktuell gefundenen kollidierenden Kugeln zur Kontaktpunktbestimmung genutzt. Diese Fähigkeit macht sie für Anwendungen mit haptischer Interaktion, bei denen eine besonders kleine Zeitschrittweite (typisch: 1 - 2 ms) nicht überschritten werden darf, besonders interessant.

Weiterhin besteht die Möglichkeit, Algorithmen für konvexe Modelle auch bei nicht konvexen Modellen einzusetzen, indem diese in einem Vorverarbeitungsschritt in konvexe Teilmodelle zerlegt werden. Einfache Anwendung z.B. des GJK-Algorithmus auf einen Satz konvexer Teilmodelle bringt jedoch weitere Probleme mit sich. Ein simples Beispiel zeigt Abbildung 2.9: Fällt der Quader auf das nicht konvexe Objekt unten, so dass dieses stark durchdrungen wird, liefern die lokalen Kontaktattribute an den betroffenen konvexen Teilobjekten keine geeignete Approximation an die globalen Kontaktattribute Durchdringungstiefe und Kontaktnormale. Wirkten während der Simulation Kontaktkräfte entlang der lokal bestimmten Kontaktnormalen, könnte die Durchdringung nicht korrigiert werden. Kim et al. [74] beschreiben ein Verfahren, um dieses Problem näherungsweise und effizient unter Nutzung der paarweisen Minkowski-Summen und Rasterisierungsmechanismen einer Grafikkarte zu lösen.

2.4. Die Kollisionsbehandlung

Gemeinhin werden im Starrkörpermodell *ruhende Kontakte* und *Kollisionen* unterschieden: Während ein Kontakt lediglich die (weitere) Durchdringung zweier Körper verhindert, sorgt eine Kollision für einen Abstoßungseffekt zwischen beiden Körpern. In der Realität resultieren solche Effekte aus Verformungen der beteiligten Körper. Das Starrkörpermodell kann den Effekt daher offensichtlich nur phänomenologisch nachbilden. Da keine Verformung stattfinden kann, wird eine infinitesimal kurze Berührungszeit $\delta t \to 0$ angenommen. Würde man mit Kräften im Newtonschen Sinne arbeiten, müsste sich der Verlauf einer wirkenden Kollisionskraft entsprechend einem Dirac-Impuls (auch: Delta-Distribution, siehe z.B. [31], Seiten 637f.) annähern. Um dieses numerisch mindestens problematische Verhalten zu umgehen, arbeitet man bei Kollisionen direkt mit dem Impuls, der durch die Einwirkung einer Kraft über einen infinitesimalen Zeitschritt hinweg auftritt. Ein Impuls ruft demnach eine nicht stetige, instantane Änderung der Geschwindigkeiten der Körper auf, zwischen denen er wirkt. Im folgenden Unterkapitel werden zunächst einige Grundlagen zur Arbeit mit Impulsen anstelle von Kräften erläutert.

2.4.1. Grundlagen

Der Impuls

Wirkt eine Kraft $f(t)$ über einen Zeitraum t_0 bis t_1 hinweg, so resultiert hieraus der Impuls p als das zeitliche Integral über die Kraft:

$$\vec{p} = \int_{t_0}^{t_1} \vec{f}(t)\mathrm{d}t \tag{2.2}$$

Aufwenden eines Impulses auf einen Starrkörper

Abbildung 2.10 zeigt die schematische Darstellung einer Kollision zweier Starrkörper. Hierin sind $\vec{x}_i$ die jeweiligen Schwerpunkte, $\vec{\omega}_i$ die Winkelgeschwindigkeiten, $\vec{r}_i$ die Vektoren vom jeweiligen Schwerpunkt zum Kontaktpunkt, $\vec{n}$ die gemeinsame Kontaktnormale, m_i die Massen und $\mathbf{\Theta}_i$ die jeweiligen Trägheitstensoren mit Bezug auf die Schwerpunkte. Alle Vektoren und Trägheitstensoren sind im ortsfesten Weltkoordinatensystem (WK) formuliert.

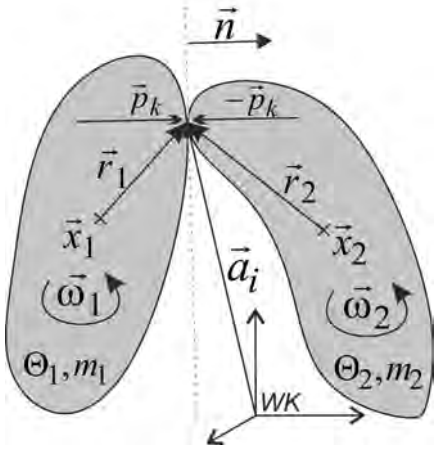


Abbildung 2.10.: Darstellung der für die Kollisionsbehandlung relevanten Größen

Für diese Situation gelten folgende Zusammenhänge. Wirkt eine *Kollisionskraft* $\vec{f}_k$ im Kontaktpunkt auf den Körper i, so gilt für diese Kraft und das resultierende Versatzmoment $\vec{\tau}_k$ (vgl. auch Bewegungsgleichungen nach Newton und Euler, Kapitel 3.1.1 ab Seite 76):

$$\begin{aligned} \vec{f}_k &= m_i \ddot{\vec{x}}_i \\ \vec{\tau}_k = \vec{r}_i \times \vec{f}_k &= \mathbf{\Theta}_i \dot{\vec{\omega}}_i + \vec{\omega}_i \times \mathbf{\Theta}_i \vec{\omega}_i \end{aligned} \tag{2.3}$$

Von Mirtich [88] stammt die folgende Herleitung der Vorschrift für das Aufbringen eines Kollisionsimpulses. Der letzte Term der zweiten Zeile in Gleichung 2.3 ist der gyroskopische Term. Er wird für die infinitesimale Dauer der Kollision vernachlässigt. Außerdem werden $\mathbf{\Theta}_i, m_i$ und $\vec{r}_i$ für die Dauer der Kollision als konstant angenommen. Mit diesen Annahmen können obige Gleichungen wie folgt über die Zeit integriert werden:

$$\begin{aligned} \vec{p}_k &= m_i \Delta\dot{\vec{x}}_i \\ \vec{r}_i \times \vec{p}_k &= \mathbf{\Theta}_i \Delta\vec{\omega}_i \end{aligned} \tag{2.4}$$

Hierin ist $\vec{p}_k = \int_{t_0}^{t_0+h} \vec{f}(\tau)\mathrm{d}\tau$ der Kollisionsimpuls, der über den Zeitraum h hinweg auf den Körper i übertragen wird und hier eine entsprechende Änderung der Schwerpunktsgeschwindigkeit $\Delta\dot{\vec{x}}$ und der Winkelgeschwindigkeit $\Delta\vec{\omega}$ hervorruft. Umgestellt nach $\Delta\dot{\vec{x}}$ und $\Delta\dot{\vec{\omega}}$ erhält man somit eine Vorschrift für die Änderung von Geschwindigkeit und Winkelgeschwindigkeit eines Starrkörpers durch das Aufbringen eines Kollisions*impulses* $\vec{p}_k$ anstelle einer Kollisions*kraft*:

$$\begin{aligned} \Delta\dot{\vec{x}}_i &= \frac{1}{m_i}\vec{p}_k \\ \Delta\vec{\omega}_i &= \mathbf{\Theta}_i^{-1}\left(\vec{r}_i \times \vec{p}_k\right) \end{aligned} \tag{2.5}$$

Die Geschwindigkeit $\dot{\vec{a}}_i$ eines Punktes $\vec{a}_i$ auf Körper i ergibt sich aus:

$$\dot{\vec{a}}_i = \dot{\vec{x}}_i + \vec{\omega}_i \times \vec{r}_i \tag{2.6}$$

Setzt man die Zusammenhänge aus Gleichung 2.5 in Gleichung 2.6 ein, so erhält man für die Änderung der Geschwindigkeit des Kontaktpunktes $\Delta\dot{\vec{a}}_i$ bei Aufwendung eines Impulses $\vec{p}_k$ in diesem Punkt:

$$\begin{aligned} \Delta\dot{\vec{a}}_i &= \Delta\dot{\vec{x}}_i + \Delta\vec{\omega}_i \times \vec{r}_i \\ &= \frac{1}{m_i}\vec{p}_k + \left(\mathbf{\Theta}_i^{-1}\left(\vec{r}_i \times \vec{p}_k\right)\right) \times \vec{r}_i \\ &= \left(\frac{1}{m_i}\mathbf{E} - \mathbf{r}_i^\star \mathbf{\Theta}_i^{-1} \mathbf{r}_i^\star\right)\vec{p}_k \end{aligned} \tag{2.7}$$

Hierin sind $\mathbf{E} \in \mathbb{R}^{3\times3}$ die Einheitsmatrix und $\mathbf{r}_i^\star$ die Kreuzproduktmatrix (siehe Anhang A.3) zum Vektor $\vec{r}_i$. Somit besteht ein Zusammenhang zwischen einer gewünschten Änderung der Geschwindigkeit eines Körperpunktes und eines dafür notwendigen, auf-

zubringenden Impulses im selben Punkt. Dieser Zusammenhang dient als Grundlage für die Anwendung der nachfolgend beschriebenen Kollisionsmodelle.

Drei wesentliche Modelle für die Kollisionsbehandlung zwischen Starrkörpern sind bekannt. Besonders verbreitet ist *Newtons Stoßgesetz*. Zwei im Kontext von Anwendungen der virtuellen Realität deutlich seltener eingesetzte Modelle sind die *Hypothese von Poisson* und die *Hypothese von Stronge*.

2.4.2. Newtons Stoßgesetz

Newtons Stoßgesetz trifft eine Aussage über die relative Geschwindigkeit im Kontaktpunkt entlang der Kontaktnormalen vor (v^v_{rel}) und nach (v^n_{rel}) dem Zusammenstoß:

$$v^n_{\mathrm{rel}} = -\varepsilon \cdot v^v_{\mathrm{rel}} \tag{2.8}$$

Hierin ist ε der Elastizitätskoeffizient der betrachteten Körperpaarung (engl.: *Coefficient of Restitution*) mit Beträgen typischerweise zwischen 0 und 1. Aufgrund seiner Einfachheit ist es das meist angewandte Modell zur Kollisionsauflösung im Anwendungsbereich Virtuelle Realität, so u.a. bei Bender [17], Giang et al. [52], Müller et al. [93] und Guendelman et al. [57].

2.4.3. Poissons Hypothese

Poissons Hypothese [54] setzt den Impuls $\vec{p}_e$, der notwendig ist, um die relative Geschwindigkeit im Kontaktpunkt zu eliminieren, in ein Verhältnis zum Gesamtimpuls der Kollision $\vec{p}$:

$$\vec{p} = \vec{p}_e \cdot (1 + \varepsilon) \tag{2.9}$$

Dieses Modell wird u.a. von Bender [17] zur Kollisionsbehandlung eingesetzt.

2.4.4. Stronges Hypothese

Werden bei der Kollisionsbehandlung zusätzlich Reibungseffekte berücksichtigt, kann die Anwendung der Kollisionsmodelle nach Newton und Poisson zur Verletzung der Energieerhaltung führen [88], da die kinetische Energie im Gesamtsystem ansteigt.

Stronge [129] entwickelt daher ein Modell, dass explizit die Einhaltung der Energieerhaltung berücksichtigt und sogar immer dissipativ ist (engl.: *Stronge's internal dissipation hypothesis*). Dissipativ bedeutet in diesem Zusammenhang, das System verliert kinetische Energie. In Stronges Modell wird die Arbeit W_D, die während der Dekompressionsphase durch den Kollisionsimpuls in Kontaktnormalenrichtung aufgebracht wird, in ein Verhältnis zur Arbeit W_C während der Kompressionsphase gestellt. Arbeit W ist definiert als Produkt aus Kraft f und zurückgelegtem Weg x:

$$W = f \cdot x \tag{2.10}$$

Damit lautet die Hypothese von Stronge:

$$W_D = (1 - \varepsilon^2) \cdot W_C \tag{2.11}$$

Dieses Modell wird von Mirtich [88] während der Kollisionsbehandlung unterstützt.

Die Elastizitätskoeffizienten ε ergeben sich aus Materialeigenschaften der kollidierenden Körper. Bei einer Paarung mit unterschiedlichen Werten nutzen einige Autoren das Minimum beider Werte, so z.B. Moore und Wilhelms [95] sowie Guendelman [57]. Bender [17] wählt das Produkt beider Koeffizienten.

2.4.5. Einbindung der Kollisionsbehandlung in den Simulationsablauf

Die direkte Anwendung eines Impulses auf einen Körper ruft eine nicht stetige Geschwindigkeitsänderung hervor. Die Behandlung von Kollisionen unterscheidet sich demnach grundlegend von der Behandlung ruhender Kontakte, in denen Kontaktkräfte über nicht-infinitesimale Zeiträume hinweg wirken können. Daher setzen die meisten Autoren in ihren Systemen einen eigenen Schritt zur Kollisionsbehandlung vor die Behandlung ruhender Kontakte. Kontakte und Kollisionen werden dazu meist anhand der auftretenden relativen Kontaktnormalengeschwindigkeit unterschieden, so z.B. bei Mirtich [88], Son et al. [123], Bender [17], Hahn [59] und Kavan [73]. Mirtich [86] und Son et al. [123] nennen dieses Vorgehen „hybrides Verfahren".

Guendelman et al. [57] können in ihrem Verfahren zwar auf Geschwindigkeitsgrenzwerte als Unterscheidungskriterium verzichten, auch hier geschieht die Kollisionsbehandlung aber in einem eigenen Schritt.

2.5. Verfahren zur Einhaltung von Zwangsbedingungen

2.5.1. Grundlagen: Zwangsbedingungen und Zwangskräfte

Zwangsbedingungen werden durch Gleichungen oder Ungleichungen beschrieben, die die Freiheitsgrade der Körper im System einschränken. Sie werden u.a. durch Gelenke und Kontakte impliziert. Zwangsbedingungen rufen *Zwangskräfte* hervor, die für ihre Einhaltung sorgen. Das 2D-Beispiel im grauen Kasten soll dem Verständnis dienen. Es wird im Verlauf dieses Kapitels mehrfach herangezogen. In Anhang B wird als zusätzliches, etwas komplexeres Beispiel ein 2D-Modell eines SCARA-Roboters behandelt.

Es existieren verschiedene Unterscheidungskriterien für Zwangsbedingungen. Eine Zwangsbedingung ist *holonom*, wenn sie sich in der Form einer Gleichung $C(\vec{x}, t) = 0$ darstellen lässt. Hierin beschreibt $\vec{x}$ Positionen und Orientierungen des Mehrkörpersystems, t die Zeit. Durch Gelenke implizierte Zwangsbedingungen sind typischerweise holonom.

Im Gegensatz dazu sind u.a. Kontaktzwangsbedingungen durch Ungleichungen darstellbar und damit *anholonom*. Am Beispiel des Massepunktes aus dem 2D-Beispiel hieße das: Soll sichergestellt werden, dass der Massepunkt nicht unter die Gerade $x_2 = 0$ fällt, so lautet die zugehörige Kontaktzwangsbedingung: $C(\vec{x}, t) : x_2 \geq 0$. Diese Zwangsbedingung ist außerdem *unilateral*, weil sie durch eine Ungleichung dargestellt wird. Hieraus resultiert auch, dass eine zugehörige Zwangskraft, die für die Einhaltung dieser Zwangsbedingung sorgt, *nur in eine Richtung* wirkt, im Beispiel nur nach oben.

Holonome Zwangsbedingungen sind immer auch *bilateral* (engl.: *equality constraints*), d.h. die aus ihnen resultierenden Zwangskräfte wirken in positiver wie in negativer Richtung. Bilaterale Zwangsbedingungen können jedoch auch anholonom sein. Dies gilt u.a. für solche Zwangsbedingungen, die sich explizit auf Geschwindigkeiten oder Beschleunigungen des Mehrkörpersystems beziehen. Ein Beispiel hierfür sind Differenzialzwangsbedingungen, die in Kapitel 3.4.2 ab Seite 111 behandelt werden. Unilaterale Zwangsbedingungen (engl.: *Complementarity Constraints*) sind immer auch anholonom. Die Zwangsbedingung aus dem 2D-Beispiel ist demnach bilateral und holonom.

2D-Beispiel

Abbildung 2.11 zeigt ein zweidimensionales Beispiel. Der Massepunkt (keine Ausdehnung, keine Orientierung) m hat zunächst zwei Freiheitsgrade, er kann sich in der Ebene translatorisch in x_1- wie in x_2-Richtung bewegen. Seine Bewegung unterliegt dabei äußeren Kräften $\vec{f}_{\text{ext}}$. Durch eine Zwangsbedingung soll sichergestellt werden, dass er sich nur entlang der dargestellten Geraden bewegen kann, d.h. er verliert einen Freiheitsgrad. Für seine Koordinaten x_1, x_2 muss gelten: $x_2(t) = 2 \cdot x_1(t)$. Zur Einhaltung dieser Zwangsbedingung muss ggf. eine Zwangskraft $\vec{f}_z$ wirken. Die zugehörige Zwangsbedingung lautet:

$$C(\vec{x}, t) = 2 \cdot x_1(t) - x_2(t) = 0 \quad | \quad \forall t \tag{2.12}$$

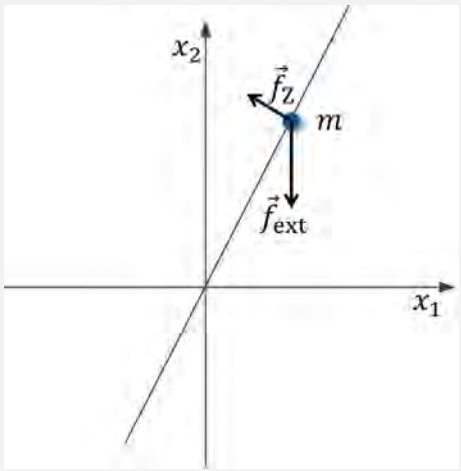


Abbildung 2.11.: 2D-Beispiel für Zwangsbedingung und Zwangskraft: Der Massepunkt darf sich nur entlang der Geraden $x_2 = 2x_1$ bewegen.

Je nach eingesetztem Verfahren zur Einhaltung von Zwangsbedingungen kann es notwendig sein, anstelle einer positions- und orientierungsbezogenen Zwangsbedingung geschwindigkeits- oder beschleunigungsbezogene Zwangsbedingungen zu for-

mulieren. Diese erhält man durch Ableiten nach der Zeit. Formal bedeutet das:

$$\begin{aligned} & \frac{\mathrm{d}}{\mathrm{d}t}\left(C(\vec{x}(t),t)\right) && = 0 \\ \Rightarrow & \frac{\partial C}{\partial \vec{x}}\dot{\vec{x}} + \frac{\partial C}{\partial t} && = 0 \\ \Rightarrow & \mathbf{J}\dot{\vec{x}} + \frac{\partial C}{\partial t} && = 0 \end{aligned} \tag{2.13}$$

Hierin ist $\mathbf{J} = \frac{\partial C}{\partial x}$ die *Jacobimatrix der Zwangsbedingung*[6]. Sie wird vor allem bei den Verfahren zur Bestimmung der inversen Dynamik mit Lagrange-Multiplikatoren in Kapitel 2.5.5 eine zentrale Rolle spielen. Unterliegt ein System $m \geq 1$ Zwangsbedingungen, ergibt sich durch deren Ableitungen eine Jacobimatrix der Zwangsbedingungen mit m Zeilen.

Für die zweite symbolische Ableitung erhält man:

$$\begin{aligned} & \frac{\mathrm{d}^2}{\mathrm{d}t^2}\left(C(\vec{x}(t),t)\right) && = 0 \\ \Rightarrow\ & \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial C}{\partial \vec{x}}\dot{\vec{x}} + \frac{\partial C}{\partial t}\right) && = 0 \\ \Rightarrow\ & \frac{\partial C}{\partial \vec{x}}\ddot{\vec{x}} + \frac{\partial^2 C}{\partial \vec{x}^2}\dot{\vec{x}}^2 + 2\frac{\partial^2 C}{\partial t \partial \vec{x}}\dot{\vec{x}} + \frac{\partial^2 C}{\partial t^2} && = 0 \\ \Rightarrow\ & \mathbf{J}\ddot{\vec{x}} + \frac{\partial^2 C}{\partial \vec{x}^2}\dot{\vec{x}}^2 + 2\frac{\partial^2 C}{\partial t \partial \vec{x}}\dot{\vec{x}} + \frac{\partial^2 C}{\partial t^2} && = 0 \end{aligned} \tag{2.14}$$

Anwendung auf das 2D-Beispiel

$$\begin{aligned} & \frac{\mathrm{d}}{\mathrm{d}t}\left(C(\vec{x}(t),t)\right) && = 0 \\ \Rightarrow\ & 2\cdot\dot{x}_1 - \dot{x}_2 && = 0 \end{aligned} \tag{2.15}$$

[6] Anhang C stellt einen Zusammenhang her zwischen Jacobimatrix der Zwangsbedingungen und der aus der Robotik bekannten Jacobimatrix der Koordinatentransformationen.

In Matrix-Vektor-Notation erhält man für die erste Ableitung:

$$\underbrace{\begin{pmatrix} 2 & -1 \end{pmatrix}}_{\mathbf{J}} \cdot \underbrace{\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix}}_{\dot{\vec{x}}} \underbrace{+0}_{\frac{\partial C}{\partial t}} = 0 \tag{2.16}$$

Für die zweite Ableitung erhält man:

$$\begin{aligned} & \frac{\mathrm{d}^2}{\mathrm{d}t^2}\left(2x_1 - x_2\right) &&= 0 \\ \Rightarrow\ & \frac{\mathrm{d}}{\mathrm{d}t}\left(2\dot{x}_1 - \dot{x}_2\right) &&= 0 \\ \Rightarrow\ & 2\ddot{x}_1 - \ddot{x}_2 &&= 0 \end{aligned} \tag{2.17}$$

$$\underbrace{\begin{pmatrix} 2 & -1 \end{pmatrix}}_{\mathbf{J}} \cdot \underbrace{\begin{pmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{pmatrix}}_{\ddot{\vec{x}}} \underbrace{+0}_{\frac{\partial^2 C}{\partial \vec{x}^2}\dot{\vec{x}}^2 + 2\frac{\partial^2 C}{\partial t \partial \vec{x}}\dot{\vec{x}} + \frac{\partial^2 C}{\partial t^2}} = 0 \tag{2.18}$$

Am 2D-Beispiel wird intuitiv klar, dass die Zwangskraft $\vec{f_{\mathrm{Z}}}$ orthogonal zur gültigen Bewegungsrichtung des Masseteilchens wirken muss: Damit durch die Zwangskraft keine Energie in das System eingebracht wird, darf sie keine Leistung $P = \vec{f_{\mathrm{Z}}}\dot{\vec{x}}$ erbringen. Damit dieses Produkt Null wird, müssen Zwangskraft und zulässige Bewegungsrichtung senkrecht zueinander verlaufen.

Das allgemeingültige Prinzip hinter dieser Intuition ist das *d'Alembert-Prinzip* (siehe z.B. [76]), die Erweiterung des *Prinzips der virtuellen Arbeit* auf dynamische Systeme. Um dieses zu erläutern, müssen zunächst einige Begriffe aus der klassischen Mechanik eingeführt werden.

Virtuelle Verrückungen Virtuelle Verrückungen $\delta\vec{x}$ sind infinitesimale Verschiebungen in den Koordinaten des Mehrkörpersystems. Sie haben drei wesentliche Eigenschaften:

1. Sie sind infinitesimal klein

2. Sie sind zeitlos ($\delta t = 0$). Hieraus folgt auch, dass die Zwangsbedingungen während einer virtuellen Verrückung unverändert bleiben.

3. Sie erfüllen die Zwangsbedingungen, die zum Ausgangszeitpunkt t vorliegen.

Virtuelle Arbeit ist die Arbeit, die durch Kräfte bei virtuellen Verrückungen erbracht wird.
Das d'Alembert-Prinzip besagt: Die Summe der durch Zwangskräfte erbrachten virtuellen Arbeit verschwindet.

Damit das d'Alembert-Prinzip gilt, muss die durch Zwangskräfte erbrachte Leistung bei virtuellen Verrückungen ($P = \vec{f}_Z^T \dot{\vec{x}}$) verschwinden. Setzt man für die Zwangskräfte $\vec{f}_Z$ den Term $\mathbf{J}^T \vec{\lambda}$ ein, so erhält man:

$$\begin{aligned} P = & \vec{f}_Z^T & \dot{\vec{x}} = 0 \\ & \left(\mathbf{J}^T \vec{\lambda}\right)^T & \dot{\vec{x}} = 0 \\ & \vec{\lambda}^T \mathbf{J} & \dot{\vec{x}} = 0 \end{aligned} \tag{2.19}$$

Berücksichtigt man in Gleichung 2.13 die Eigenschaften der virtuellen Verrückung, während derer die Zwangsbedingung unverändert bleibt ($\frac{\partial C}{\partial t} = 0$!), so gilt in der letzten Zeile in $\mathbf{J}\dot{\vec{x}} = \vec{0}$. Damit ist Gleichung 2.19 und damit das d'Alembert-Prinzip erfüllt, wenn für die Zwangskräfte gilt:

$$\vec{f}_Z = \mathbf{J}^T \vec{\lambda} \tag{2.20}$$

Man kann sagen: Die Richtungen der Zwangskräfte im Mehrkörpersystem werden durch die Zwangsbedingungen selbst vorgegeben. Lediglich ihre jeweiligen vorzeichenbehafteten Beträge $\vec{\lambda}$ sind unbekannt.

Maximal- vs. Minimalkoordinaten Grundsätzlich gibt es zwei Herangehensweisen, um Zwangsbedingungen bei der Bewegungssimulation von Mehrkörpersystemen zu berücksichtigen (vgl. Abbildung 2.12): In Minimalkoordinatenansätzen werden Bewegungsgleichungen für das Mehrkörpersystem entwickelt, die nur noch genau die Freiheitsgrade besitzen, die nicht durch Zwangsbedingungen eingeschränkt sind. Bei Animation dieser Bewegungsgleichungen werden die Zwangsbedingungen implizit einge-

halten. Zur Simulationslaufzeit müssen keine Zwangskräfte bestimmt werden, eine besonders rechenzeitintensive Aufgabe der Simulation entfällt somit.

In Maximalkoordinatenansätzen hingegen wird jeder Körper des Mehrkörpersystems zunächst in allen sechs Freiheitsgraden betrachtet. Zusätzlich zu externen Kräften wie der Gravitation werden explizit die wirkenden Zwangskräfte bestimmt, die für die Einhaltung der Zwangsbedingungen sorgen. Dies erfordert zwar einen erheblichen Rechenaufwand zur Simulationslaufzeit, liefert mit den Beträgen der wirkenden Zwangskräften aber auch eine interessante zusätzliche Information neben dem Bewegungsverhalten des Mehrkörpersystems. In Anwendungen der Virtuellen Realität haben Maximalkoordinatenansätze eine deutliche Dominanz - vermutlich vor allem, weil sie flexibler an eine Vielzahl unterschiedlicher Anwendungen angepasst werden können.

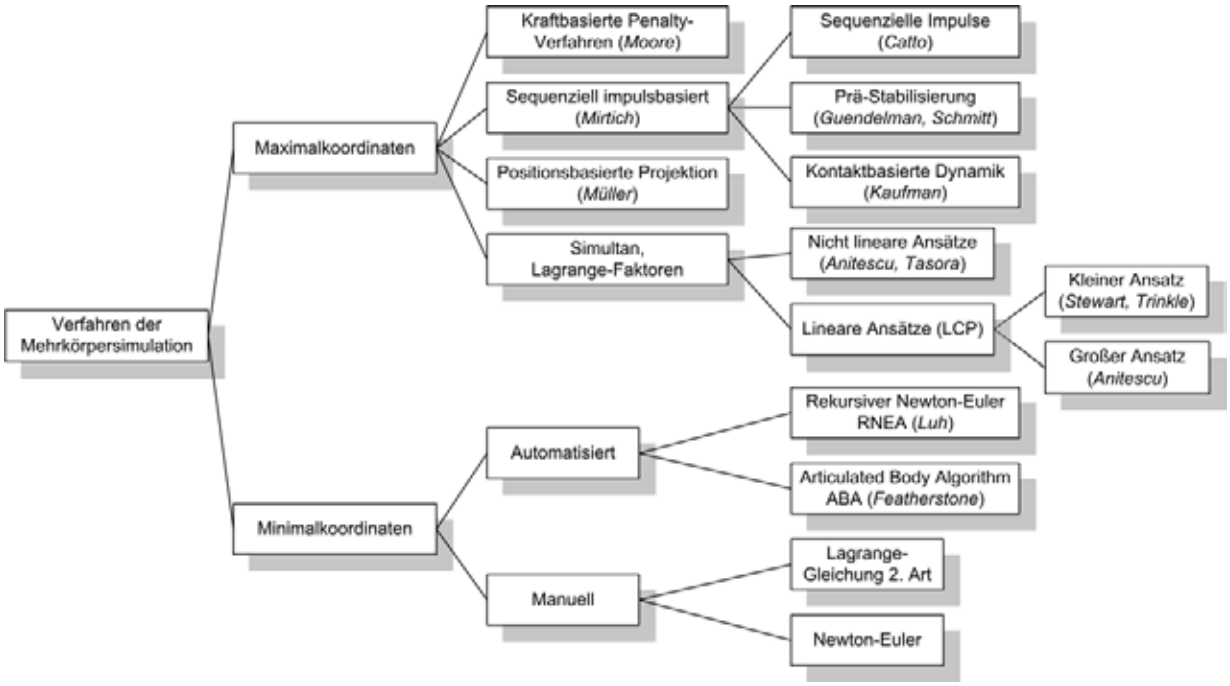


Abbildung 2.12.: Gliederung der Verfahren zur Einhaltung von Zwangsbedingungen. In Kursivschrift ggf. die Autoren mit wesentlichen Beiträgen zu den jeweiligen Verfahren

2.5.2. Penalty Verfahren

Grundidee der Penalty Verfahren ist es, Verletzungen von Zwangsbedingungen lokal durch „strafende“ Kräfte zu korrigieren. Die Richtung der korrigierenden Kräfte ergibt

sich, wie oben erläutert, aus den Zwangsbedingungen selbst, am Beispiel Kontaktpunkt entspricht sie der Kontaktnormalen. Ihr Betrag wird typischerweise aus einem einfachen Federmodell gewonnen, z.B. bei Moore und Wilhelms [95] und bei Yamane et al. [142]. Die resultierenden Kräfte werden in den Bewegungsgleichungen berücksichtigt und sorgen so für eine näherungsweise Einhaltung der Zwangsbedingungen. Diese Verfahren sind äußerst einfach zu implementieren und besitzen durch ihre lokale Herangehensweise ein lineares Laufzeitverhalten in Bezug auf die Anzahl der Zwangsbedingungen.

Bekannt sind jedoch auch die Schwächen dieser Verfahren. Zu schwache Federn führen dazu, dass die Verletzungen der Zwangsbedingungen nicht ausreichend genau bzw. zu langsam korrigiert werden. Sollen hingegen beinahe starre Kollisionen simuliert werden, müssen sehr steife Federn bzw. sehr große Federkonstanten Anwendung finden. Diese resultieren aber in zu starken, „sprunghaften“ Gegenreaktionen. Aus mathematischer Sicht entstehen diese durch die numerische Integration von steifen Differentialgleichungen, die nur mit sehr kleinen Zeitschritten behandelt werden können. Diese lassen den potenziellen Geschwindigkeitsvorteil der Verfahren schnell dahinschmelzen. Für einfache Anwendungen werden solche Verfahren bis heute dennoch eingesetzt. Um die Schwächen einfacher Federmodelle zu beseitigen, nutzt Drumwright [42] zusätzlich zum proportionalen Federkraftanteil differenzielle und integrale Terme in Abhängigkeit vom Fehlerterm $C(\vec{x}, t)$, der Größe der Verletzung der positions- bzw. orientierungsbezogenen Zwangsbedingung. Die Bestimmung der Korrekturkräfte erfolgt durch einen PID-Regler, der die Verletzung von Zwangsbedingungen durch das Einstellen korrigierender Kräfte ausgleicht. Wagner [135] liefert eine formale Beschreibung dieser Verfahren im kontinuierlichen Zeitbereich.

Das Grundproblem bleibt jedoch erhalten: Ein einmal parametrierter PID-Regler kann nur ungenügend auf stark variierende Kontaktsituationen reagieren. Müssen Zwangskräfte zwischen Körpern bestimmt werden, die um Größenordnungen abweichende Massen haben von denen, auf die der Regler hin optimiert wurde, wird er den Fehler nur langsam oder mit starkem Überschwingen ausgleichen können. Als universelles Simulationsverfahren für ein breites Anwendungsspektrum in der VR finden solche Verfahren heute daher praktisch keine Verwendung mehr.

2.5.3. Sequenzielle, impulsbasierte Verfahren

Der Begriff impulsbasierte Dynamiksimulation ist recht unscharf, da sich zum heutigen Zeitpunkt ein Großteil der Ansätze zur Behandlung von Zwangsbedingungen unter die-

sem Begriff fassen lässt. Zunächst bedeutet impulsbasiert nur, dass zur Erfüllung von Zwangsbedingungen während der numerischen Simulation keine zeitkontinuierlichen Zwangskräfte, sondern zeitdiskrete *Zwangsimpulse* eingesetzt werden. Ein Zwangsimpuls $\vec{p}_Z$ kann man dabei als das zeitliche Integral der Zwangskraft $\vec{f}_Z$ über einen diskreten Zeitschritt verstehen:

$$\vec{p}_Z(t_0 + h) = \int_{t_0}^{t_0+h} \vec{f}_Z(\tau)\mathrm{d}\tau \tag{2.21}$$

Der Übergang von kraftbasierter zu impulsbasierter Dynamiksimulation erscheint im Kontext der zeitdiskreten Simulation im Computersystem naheliegend, da das Konzept der zeitkontinuierlichen Kraft im zeitdiskreten Simulationssystem zu Problemen führt. Die Schwierigkeiten der Penalty-Verfahren machen dies deutlich.

Obwohl zwischen den impulsbasierten Verfahren heute große Ähnlichkeiten bestehen, haben sie sehr unterschiedliche Ursprünge und hatten zunächst nur wenig miteinander gemein. Einerseits wurden die Verfahren mit Lagrange-Multiplikatoren von einem zunächst kraftbasierten durch Stewart und Trinkle [126, 127] und Anitescu und Potra [6] in einen implizit impulsbasierten Ansatz weiterentwickelt, dies wird in Kapitel 2.5.5 detailliert behandelt. Gleichzeitig, jedoch unabhängig davon, entwickelte Mirtich [91, 88] einen Weg, Zwangsimpulse zur Auflösung einzelner Kollisionen zu bestimmen und auf die Körper des Systems aufzubringen, siehe hierzu auch Kapitel 2.4.1 ab Seite 35. Auf dieser Grundlage entstanden viele weitere, in Bezug auf mehrere Kontakte bzw. Zwangsbedingungen sequenziell arbeitende Verfahren, die Thema des nachfolgenden Kapitels sind.

Ursprung der impulsbasierten Verfahren

Den Beginn der Entwicklung der sequenziellen, impulsbasierten Verfahren markiert Mirtich [91, 88] zunächst nur für Kontakte. Er stellt dar, wie ein Kollisionsimpuls bestimmt und angewendet werden muss, der an den jeweiligen Kontaktpunkten auf den kollidierenden Körpern eine geforderte Änderung der relativen Geschwindigkeit in diesen Punkten hervorruft. Sein Verfahren arbeitet mit variabler Zeitschrittweite, so dass Kollisionsimpulse exakt zum individuellen Zeitpunkt jeder Kollision bestimmt und aufgebracht werden können. Unter der Annahme, dass Starrkörperkollisionen eine infinitesimale Zeitdauer haben, kann jede Kollision unabhängig vom restlichen System behandelt werden. Zur Behandlung von bleibenden Kontakten nutzt er Serien von Mikrokollisionen: Im

Falle von aufeinander ruhenden Körpern wird eine geringe relative Sollgeschwindigkeit bestimmt, die einen Körper ein kleines Stückchen anhebt, bevor er durch die Gravitation erneut auf den unteren stößt. Diese Serien von Mikrokollisionen resultieren in einem vibrierenden Verhalten des oberen Körpers. Laut Mirtich lässt sich dieses Vibrieren aber mit einer Amplitude realisieren, die kleiner als ein Pixel auf dem darstellenden Bildschirm ist, so dass die Bewegung nicht wahrnehmbar ist. Komplexere Situationen, z.B. Stapel mit vielen übereinanderliegenden Ebenen, lassen sich mit diesem Verfahren jedoch nicht stabil realisieren.

Das Prästabilisierungsverfahren

Der nächste Schritt der sequenziellen, impulsbasierten Ansätzen wird gleichzeitig von Schmitt [118] und Guendelman et al. [57] vorgestellt. Beide stellen eine neue Methode vor, um die Simulation ruhender Kontakte zu verbessern. Ausgehend von einem zum Zeitpunkt t_0 detektierten Kontaktpunkt wird zunächst die Größe des positionsbezogenen Fehlers (Durchdringungstiefe) in diesem Kontaktpunkt zum nächsten Zeitpunkt $C(\vec{x}(t_0 + h), t_0 + h)$ berechnet, wenn Geschwindigkeiten und Positionen ohne Berücksichtigung von Kontakten aufintegriert werden. Dann wird ein Impuls bestimmt, der die Geschwindigkeiten der beteiligten Körper im Zeitpunkt t_0 so verändert, dass der positionsbezogene Fehler im nächsten Zeitschritt näherungsweise behoben sein wird. Bei mehr als einem Kontaktpunkt läuft der Algorithmus über alle Kontakte und passt die aufgebrachten Impulse iterativ an, solange bis die Fehler in allen Kontaktpunkten im nächsten Zeitschritt unter eine vorher festgelegte Fehlerschranke gefallen sind. Erst dann wird der Zeitschritt tatsächlich ausgeführt. An seinem Ende erreicht das Mehrkörpersystem einen Zustand, in dem alle Kontaktzwangsbedingungen erfüllt sind.

Schmitt [118] behandelt in seinem Verfahren neben Kontakten und Kollisionen auch Gelenke und sogar Systeme von Gelenken. Es wird beschrieben, wie ein System mit Gelenken mit demselben iterativen Ansatz gelöst werden kann wie eines mit ausschließlich ruhenden Kontakten. Abbildung 2.13 verdeutlicht das Vorgehen für eine einzelne Kontakt- oder Gelenksituation. Darüber hinaus zeigt er, wie die Korrekturimpulse zur Einhaltung holonomer Zwangsbedingungen, also z.B. für klassische Gelenkzwangsbedingungen, in einem linearen Gleichungssystem simultan bestimmt werden können. Kontaktbedingungen sind wegen ihrer anholonomen Eigenschaften vom simultanen Vorgehen jedoch ausgeschlossen und müssen weiterhin iterativ behandelt werden. Das Verfahren von Schmitt wird von Bender et al. [19, 20, 22, 17] weiterent-

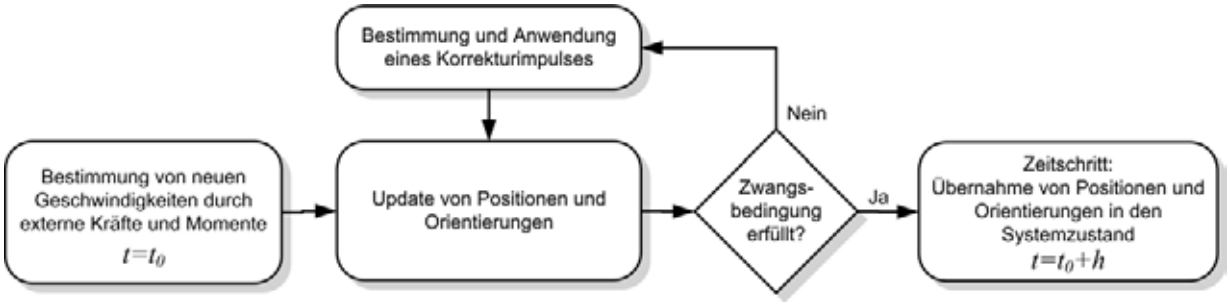


Abbildung 2.13.: Prästabilisierungsverfahren nach Schmitt und Bender sowie Guendelman und Weinstein zur Behandlung eines einzelnen Gelenkes oder Kontaktes

wickelt und detailliert dokumentiert.

Den Arbeiten von Schmitt und den hierauf aufbauenden ist gemein, dass sie bei der Bestimmung von Korrekturimpulsen an Gelenken und Kontakten ein lineares Bewegungsverhalten der Punkte eines Starrkörpers annehmen. Hierdurch gelangen sie zu einem exakt lösbaren linearen Gleichungssystem, dessen Lösung die gesuchten Korrekturimpulse liefert. Diese Annahme gilt für einen beliebigen Punkt auf dem Starrkörper im Allgemeinen jedoch nicht, daher kann immer nur eine Näherung des gesuchten Korrekturimpulses bestimmt werden. Das Verfahren wird iterativ angewendet, um die exakte Lösung zu approximieren.

Weinstein et al. [137, 138] stellen an dieser Stelle ein nichtlineares System auf und erreichen so eine bessere Näherung des Korrekturimpulses. Im Gegenzug kann das nichtlineare System allerdings nicht direkt, sondern seinerseits nur näherungsweise durch ein Newton-Raphson Iterationsverfahren gelöst werden. Weinstein behandelt dieses Verfahren im Detail in ihrer Dissertation [139].

Die Bezeichnung „Prästabilisierung“ für diese Verfahrensgruppe stammt von Guendelman et al. [57] (engl.: *Pre-Stabilization*) - zweifellos in Anlehnung an das Problem der Constraint-Stabilisierung, das in Kapitel 2.7 behandelt wird.

Das Verfahren der Sequenziellen Impulse

Unter dem Begriff „Sequenzielle Impulse“ (engl.: *Sequential Impulses*) wird von Catto [32] ein weiteres Verfahren beschrieben, das in der Open Source Bibliothek *Bullet* [1] als eines von mehreren Verfahren zur Bestimmung der inversen Dynamik realisiert ist. Tatsächlich handelt es sich bei diesem Ansatz um eine serialisierte Version der

weit verbreiteten, geschwindigkeitsbasierten Lagrange-Multiplikatoren-Ansätze (s.u.). Er verläuft wie folgt (vgl. Abbildung 2.14): Im Ausgangszeitpunkt t_0 werden zunächst die Geschwindigkeiten der Körper für den neuen Zeitpunkt $t + h$ durch externe Kräfte und Drehmomente, wie die Gravitation, aktualisiert ($\dot{\vec{x}}(t + h)$). Dann wird der Fehler der Zwangsbedingungen auf der Ebene dieser Geschwindigkeiten $\mathbf{J}\dot{\vec{x}}(t + h)$ bestimmt. Im Anschluss wird ein Korrekturimpuls bestimmt und angewendet, der den Fehler auf der Ebene der Geschwindigkeiten ausgleicht ($\mathbf{J}\dot{\vec{x}}(t + h) \rightarrow \vec{0}$) .

Am Beispiel eines Kugelgelenks bedeutet das: Der Punkt des Körpers, an dem die Zwangsimpulse eines Gelenkes angreifen, wird *Ankerpunkt* genannt. Der Korrekturimpuls muss demnach die Geschwindigkeiten der Körper derart anpasst, so dass die translatorischen Geschwindigkeiten der Ankerpunkte der beiden Körper identisch werden. Daraus resultieren neue Geschwindigkeiten der Körper, mit denen als letzter Schritt das Update von Positionen und Orientierungen durchgeführt wird. Enthält das System mehrere Gelenke oder Kontakte, korrigiert der Algorithmus diese nacheinander. Dieser Vorgang muss iterativ wiederholt werden, da die Korrektur an einem Gelenk oder Kontakt den Fehler an einem anderen wieder vergrößern kann. Er wird solange wiederholt, bis der Fehler, hier also die Geschwindigkeitsdifferenz, in allen Gelenken und Kontakten unter ein gegebenes Fehlermaß fällt.

Abbildung 2.14.: Verfahren mit Sequenziellen Impulsen nach Catto [32] zur Behandlung eines einzelnen Gelenkes oder Kontaktpunktes

Genau wie die geschwindigkeitsbasierten Lagrange-Multiplikatoren-Verfahren hat auch das Verfahren mit Sequenziellen Impulsen nach Catto das Problem der notwendigen Constraint-Stabilisierung, da das Verfahren nicht direkt für die Einhaltung der ursprünglichen, positionsbezogenen Zwangsbedingung $C(\vec{x}, t) = 0$ sorgt. In seiner Präsentation beschreibt er daher einen Impuls mit korrigierender Tendenz (engl.: *Bias-Impulse*), betragsmäßig proportional zur Größe des Fehlers der positionsbezogenen Zwangsbedingung, der dem Fehler entgegenwirken soll. Dieses Vorgehen ist analog zur Stabilisierung von Zwangsbedingungen in den Lagrange-Multiplikatoren Verfahren, siehe dazu Kapitel 2.7.1.

Das Verfahren rein kontaktbasierter Dynamik

Das Verfahren der rein kontaktbasierten Dynamik von Kaufman et al. [72] unterstützt keine generellen Zwangsbedingungen, wie sie durch Gelenke impliziert werden, sondern ausschließlich Kontakte und Kollisionen. Damit ist es für viele Anwendungen der Virtuellen Realität nicht geeignet. Es zeichnet sich vor allem durch seine hohe Performanz aus, die aus linearem Laufzeitverhalten sowohl in Abhängigkeit von der Anzahl der simulierten Körper als auch von der Anzahl der Kontakte resultiert. Iglberger und Rüde erweitern dieses Verfahren in eine parallelisierte Version [64, 65], mit der sie Kontaktdynamik zwischen bis zu 1.000.000.000 (eine Milliarde!) Kugeln und nicht kugelförmigen Partikeln [66] auf speziellen Hochleistungscomputern simulieren.

2.5.4. Verfahren der positionsbasierten Projektion

Das Vorgehen der positionsbasierten Projektion nach Müller et al. [93, 94, 92] behandelt nicht direkt Starrkörper, sondern Partikelsysteme. Da man jedoch aus Partikeln, die untereinander z.B. durch Abstands-Zwangsbedingungen in Relation gesetzt sind, ebenfalls Körper mit dreidimensionaler Ausdehnung erzeugen kann, soll es hier kurz beschrieben werden.

Das Verfahren verläuft wie folgt: Ausgehend von einem Zustand des Systems zum Zeitpunkt t_0, in dem alle positionsbezogenen Zwangsbedingungen erfüllt sind, wird zunächst ein Zeitschritt der Länge h und damit eine Aktualisierung der Geschwindigkeit $\vec{v}_i$ und der Position $\vec{x}_i$ jedes Partikels unter dem Einfluss der auf ihn wirkenden externen Kräfte $\vec{f}_{i,\text{ext}}$ durchgeführt:

$$\begin{aligned} \vec{v}_i(t_0 + h) &= \vec{v}_i(t_0) + h\frac{\vec{f}_{i,\text{ext}}(t_0)}{m} \\ \vec{x}_i(t_0 + h) &= \vec{x}_i(t_0) + h\vec{v}_i(t_0 + h) \end{aligned} \tag{2.22}$$

Dies entspricht einem semi-impliziten Euler-Integrationsschritt, siehe dazu auch 2.6. Um Zwangsbedingungen der Form $C(\vec{x}) = 0$ berücksichtigen zu können, folgt für jede Zwangsbedingung der Schritt der Projektion: Ausgehend von den neuen Positionen $\vec{x}(t_0 + h)$ (Konkatenation der $\vec{x}_i$), die eine Zwangsbedingung $C_j(\vec{x}(t_0 + h))$ verletzen, wird eine Verrückung $\Delta\vec{x}_j$ gesucht, so dass die Zwangsbedingung wieder erfüllt ist:

$C(\vec{x} + \Delta\vec{x}_j) = 0$. Er macht folgende Annahme:

$$\begin{aligned} C_j(\vec{x} + \Delta\vec{x}_j) &\approx C_j(\vec{x}) + \frac{\partial C_j(\vec{x})}{\partial \vec{x}} \cdot \Delta\vec{x}_j = 0 \\ C_j(\vec{x}) + \mathbf{J}_j \Delta\vec{x}_j &= 0 \end{aligned} \tag{2.23}$$

Aus dem d'Alembert-Prinzip leitet er ab, dass $\Delta\vec{x}_j$ orthogonal zur einschränkenden Richtung der Zwangsbedingung verlaufen muss, skaliert mit einem Multiplikator λ_j:

$$\Delta\vec{x}_j = \mathbf{J}_j^T \lambda_j \tag{2.24}$$

Zusammenführung der Gleichungen 2.23 und 2.24 führt auf ein nichtlineares Gleichungssystem, dass mit einem Newton-Raphson-Iterationsverfahren gelöst wird. Das Verfahren behandelt alle Zwangsbedingungen nacheinander und iteriert mehrfach, bis alle Zwangsbedingungen $C_j(\vec{x}(t+h))$ ausreichend genau erfüllt sind.

Aufgrund seiner partikelorientierten Natur ist das Verfahren nur bedingt für die Simulation von Starrkörpern geeignet, muss ein einzelner Starrkörper doch aus Referenzpunkten und zwischen ihnen geltenden Zwangsbedingungen konstruiert werden. Im Gegenzug scheint es ein vielversprechender Ansatz für die Simulation nicht starrer Körper zu sein.

2.5.5. Verfahren mit Lagrange-Multiplikatoren

Grundlage der Verfahren mit Lagrange-Multiplikatoren ist das d'Alembert-Prinzip, aus dem für die Zwangskräfte $\vec{f_Z}$ folgt:

$$\vec{f_{\mathrm{Z}}} = \mathbf{J}^T \vec{\lambda} \tag{2.25}$$

Hierin sind $\vec{\lambda}$ die *Lagrange-Multiplikatoren*. Sie entsprechen den vorzeichenbehafteten Beträgen der Zwangskräfte $\vec{f_{\mathrm{Z}}}$, deren Richtungen bereits durch die transponierte Jacobimatrix der Zwangsbedingungen $\mathbf{J}^T$ definiert sind. Eine detaillierte Herleitung dieses Zusammenhangs liefert Kapitel 2.5.1.

Die Bewegungsgleichungen nach Newton und Euler eines Mehrkörpersystems aus n Körpern in (kartesischen) Maximalkoordinaten unter Einfluss der externen und der

Zwangskräfte lassen sich in Matrix-Vektor-Notation wie folgt formulieren:

$$\begin{aligned} \dot{\vec{p}} &= \vec{f}_{\text{ext}} + \vec{f}_{\text{Z}} \\ \Rightarrow \frac{\mathrm{d}}{\mathrm{d}t}\left(\mathbf{M}\cdot\dot{\vec{x}}\right) &= \vec{f}_{\text{ext}} + \vec{f}_{\text{Z}} \\ \Rightarrow \dot{\mathbf{M}}\cdot\dot{\vec{x}} + \mathbf{M}\cdot\ddot{\vec{x}} &= \vec{f}_{\text{ext}} + \vec{f}_{\text{Z}} \\ \Rightarrow \mathbf{M}\cdot\ddot{\vec{x}} &= \vec{f}_{\text{Z}} + \underbrace{\vec{f}_{\text{ext}} - \dot{\mathbf{M}}\cdot\dot{\vec{x}}}_{\vec{f}} \end{aligned} \tag{2.26}$$

Hierin sind:

$\dot{\vec{p}} \in \mathbb{R}^{6n}$	Die zeitliche Änderung von Impuls und Drehimpuls,
$\dot{\vec{x}} \in \mathbb{R}^{6n}$	die Schwerpunkts- und Winkelgeschwindigkeitsvektoren der n Körper des Systems,
$\mathbf{M}, \dot{\mathbf{M}} \in \mathbb{R}^{6n\times 6n}$	die Gesamtmassenmatrix des Mehrkörpersystems (siehe auch Gleichung 2.40) und ihre zeitliche Ableitung und
$\vec{f}_{\text{ext}}, \vec{f}_{\text{Z}} \in \mathbb{R}^{6n}$	die externen Kräfte und Momente und die Zwangskräfte und -momente.

Hinweis: Der Term $\vec{f}_{\text{ext}} - \dot{\mathbf{M}}\dot{\vec{x}}$ bestehend aus externen Kräften und Momenten und den gyroskopischen Momenten wird zu Gunsten der Übersichtlichkeit für den weiteren Verlauf zusammengefasst zum Term $\vec{f}$ der wirkenden Kräfte. Eine detaillierte Beschreibung und Herleitung der Terme in Gleichung 2.26 liefert Kapitel 3.1.1 ab Seite 76.

Zur Bestimmung der Körperbeschleunigungen $\ddot{\vec{x}}$ in Gleichung 2.26 müssen noch die Zwangskräfte $\vec{f}_{\text{Z}} = \mathbf{J}^T\vec{\lambda}$ und dazu die Einträge des Vektors $\vec{\lambda}$ der Lagrange-Multiplikatoren bestimmt werden. Grundsätzlich bestehen zwei Varianten, Bewegungsgleichungen und Zwangsbedingungen in eine einheitliche Formulierung zu bringen. Diese werden im Folgenden als „großer“ und „kleiner“ Lagrange-Multiplikatoren-Ansatz beschrieben.

Der „große“ Lagrange-Multiplikatoren-Ansatz

Beim „großen“ Ansatz werden Bewegungsgleichungen und Zwangsbedingungen zu einem System „zusammengeschrieben“. Solange nur bilaterale Zwangsbedingungen der Form $\mathbf{J}\ddot{\vec{x}} = \vec{b}$ berücksichtigt werden, wird ein lineares Gleichungssystem (LGS) ohne

Nebenbedingungen in folgender Form definiert:

$$\underbrace{\begin{pmatrix} \mathbf{M} & -\mathbf{J}^T \\ \mathbf{J} & \mathbf{0} \end{pmatrix}}_{\mathbf{A}} \cdot \begin{pmatrix} \ddot{\vec{x}} \\ \vec{\lambda} \end{pmatrix} = \begin{pmatrix} \vec{f} \\ \vec{b} \end{pmatrix} \tag{2.27}$$

Die Lösung des LGS ergibt die gesuchten Beträge der Zwangskräfte $\vec{\lambda}$ und die Beschleunigungen und Drehbeschleunigungen aller Körper des Systems $\ddot{\vec{x}}$.

Anwendung auf das 2D-Beispiel

Das zu bildende System für das 2D-Beispiel hat die Form (vgl. Gleichung 2.18 auf Seite 42):

$$\begin{pmatrix} m & 0 & -2 \\ 0 & m & 1 \\ 2 & -1 & 0 \end{pmatrix} \begin{pmatrix} \ddot{x}_1 \\ \ddot{x}_2 \\ \vec{\lambda} \end{pmatrix} = \begin{pmatrix} f_{\text{ext},1} \\ f_{\text{ext},2} \\ 0 \end{pmatrix} \tag{2.28}$$

Lösung nach den Unbekannten $\ddot{x}_1, \ddot{x}_2, \vec{\lambda}$ liefert:

$$\begin{aligned} \ddot{x}_1 &= \frac{2f_{\text{ext},2} + f_{\text{ext},1}}{5m} \\ \ddot{x}_2 &= \frac{4f_{\text{ext},2} + 2f_{\text{ext},1}}{5m} \\ \vec{\lambda} &= -\frac{2f_{\text{ext},1} - f_{\text{ext},2}}{5} \end{aligned} \tag{2.29}$$

Der Vektor $\vec{\lambda}$ hat in diesem Beispiel die Dimension 1, weil nur eine Zwangsbedingung vorliegt.

Die Systemmatrix $\mathbf{A}$ hat eine spezielle, dünn besetzte Struktur. Baraff [13] beschreibt eine Möglichkeit, die Matrix $\mathbf{A}$ durch Zeilen- und Spalten-Vertauschungen in eine Bandform zu überführen. Hierdurch wird eine Lösung des Systems mit linearer Zeitkomplexität in Bezug auf die Anzahl der Zwangsbedingungen und der Körper möglich. Dieses Methode wird in der Literatur häufig als „Baraff's Verfahren“ bezeichnet. Es ist jedoch wichtig zu berücksichtigen, dass sich das günstige lineare Laufzeitverhalten nur auf solche Zwangsbedingungen bezieht, die folgende Bedingungen erfüllen (in Baraffs Paper:

Primary constraints):

- Die Zwangsbedingung darf sich auf höchstens zwei Körper beziehen - Differenzialbedingungen (siehe Kapitel 3.4.2) sind hiervon ausgeschlossen.
- Sie muss bilateral sein - Kontaktzwangsbedingungen sind demnach ausgeschlossen.
- Die Zwangsbedingung darf keine zyklische Abhängigkeit definieren. Zyklische Abhängigkeiten ergeben sich z.B. bei Kränen. Hier werden Drehgelenke durch parallel dazu verbaute Hydraulikzylinder angetrieben. Für ein Beispiel siehe Abbildung 6.3 auf Seite 189.

Alle Zwangsbedingungen, die mindestens eine dieser Bedingungen nicht erfüllen (bei Baraff: *Auxiliary Constraints*), werden gesondert behandelt. In Bezug auf diese Zwangsbedingungen ist die Zeitkomplexität des Verfahrens $O(n^3)$.

Anitescu et al. [5] zeigen, wie im „großen“ Lagrange-Multiplikatoren-Ansatz unilaterale Kontaktzwangsbedingungen berücksichtigt werden können. Eine Kontaktzwangsbedingung, die die Durchdringung zweier Körper verhindert, hat auf der Ebene der Beschleunigungen die Form: $\mathbf{J}_i\ddot{\vec{x}} - b_i = a_i, a_i \geq 0$. Die Bedingung $a_i \geq 0$ sorgt dafür, dass die Körper sich voneinander entfernen, sich aber nicht weiter annähern können, weil dies Durchdringung bedeuten würde. Hierbei ist es wichtig zu berücksichtigen, dass diese Formulierung von tatsächlich ruhenden Kontakten und nicht von Kollisionen ausgeht, d.h. es wird vorausgesetzt, dass sich die Körper im Ausgangszustand nicht aufeinander zu bewegen. Der Term a_i hat die Dimension einer relativen Beschleunigung entlang der Kontaktnormalen. Kollisionen müssen zuvor an anderer Stelle behandelt werden. Für die Kontaktnormalenkraft gilt, dass sie beide Körper auseinander drücken, sie aber nicht zusammenhalten können darf. Für den zugehörigen Lagrange-Multiplikator gilt daher die Nichtnegativitätsbedingung: $\lambda_{n_i} \geq 0$. Im ruhenden Kontakt können nun zwei Situationen auftreten:

1. Beide Körper bleiben unbewegt in Kontakt oder bewegen sich voneinander weg ($a_i \geq 0$). In diesem Fall kann keine Durchdringung auftreten und die Kontaktzwangskraft verschwindet $\lambda_{n_i} = 0$.
2. Beide Körper werden durch äußere Einflüsse gegeneinander gedrückt und bleiben in Kontakt ($a_i = 0$). In diesem Fall ist die Kontaktzwangskraft positiv $\lambda_{n_i} > 0$ und verhindert die Durchdringung.

Beide Fälle werden durch die mathematische Formulierung der Komplementarität zusammengefasst: Die relative Beschleunigung in Richtung der Kontaktnormalen und die Kontaktnormalenkraft sind komplementär zueinander ($a_i \geq 0, \lambda_{n_i} \geq 0, a_i \cdot \lambda_{n_i} = 0$).

Im Folgenden werden die unterschiedlichen Zwangsbedingungen in je eigenen Jacobimatrizen zusammengefasst. Fasst man die bilateralen Zwangsbedingungen in der Matrix $\mathbf{J}_e$ ('e' wie *equality*) und die unilateralen in der Matrix $\mathbf{J}_n$ ('n' wie Kontakt**n**ormale) zusammen, können Bewegungsgleichungen und alle Zwangsbedingungen in ein gemeinsames System gefasst werden:

$$\begin{pmatrix} \mathbf{M} & -\mathbf{J_e^T} & -\mathbf{J_n^T} \\ \mathbf{J}_e & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_n & \mathbf{0} & \mathbf{0} \end{pmatrix} \cdot \begin{pmatrix} \ddot{\vec{x}} \\ \vec{\lambda_e} \\ \vec{\lambda_n} \end{pmatrix} - \begin{pmatrix} \vec{f} \\ \vec{b_e} \\ \vec{b_n} \end{pmatrix} = \begin{pmatrix} \vec{0} \\ \vec{0} \\ \vec{a}_n \end{pmatrix} \tag{2.30}$$
$$\vec{a}_n \geq \vec{0}, \quad \vec{\lambda_n} \geq \vec{0}, \quad \vec{a}_n^T \cdot \vec{\lambda_n} = 0$$

Hierbei handelt es sich um ein lineares Komplementaritätsproblem (engl.: *Linear Complementary Problem, LCP*) [38, 39], aufgrund der Kombination von Gleichungen und Ungleichungen genau genommen sogar um ein gemischtes LCP (engl.: *Mixed LCP, MLCP*). Die Komplementarität der Vektoren gilt elementweise. Die Anwendung des Skalarproduktes in der Komplementaritätsbedingung ist nur deshalb gültig, weil für jedes einzelne Element der komplementären Vektoren die Nichtnegativitätsbedingung gilt.

Zur Lösung dieser Formulierung wird häufig „Lemkes Algorithmus“ [77] herangezogen, so auch durch Anitescu und Potra [6], Baraff [11], Cline [34], Miller und Christensen [85] sowie durch Trinkle [133].

Übergang zu impulsbasierten Formulierungen Bis hierher wurde das Problem auf der Ebene von Kräften, Momenten und Beschleunigungen formuliert. Baraff [11] zeigt jedoch, dass bei Hinzunahme von Zwangsbedingungen zur Realisierung des Coulombschen Reibungsmodells (vgl. Anhang A.2) die Existenz einer Lösung des Problems nicht garantiert werden kann. Daher formulieren Anitescu et al. [6] anstelle der zeitkontinuierlichen, kraft- und beschleunigungsbasierten Modelle zeitdiskrete, impuls- und geschwindigkeitsbasierte Modelle. Somit gehen sie schon auf der Ebene des mathematischen Modells des physikalischen Vorgangs einen wichtigen Schritt in Richtung der Simulation in zeitdiskreten Computersystemen. Die Idee ist denkbar einfach: Sie ersetzen den Term der Beschleunigungen $\ddot{\vec{x}}$ in der Bewegungsgleichung 2.26 durch

den Differenzenquotienten als Näherung erster Ordnung:

$$\begin{aligned} &\mathbf{M} \cdot \frac{\dot{\vec{x}}(t+h) - \dot{\vec{x}}(t)}{h} - \mathbf{J}^T \cdot \vec{\lambda} = \vec{f}, \quad h : \text{Zeitschrittweite} \\ &\Rightarrow \mathbf{M} \cdot \dot{\vec{x}}(t+h) - \mathbf{J^T} h\lambda = \mathbf{M} \cdot \dot{\vec{x}}(t) + h\vec{f} \end{aligned} \tag{2.31}$$

Entsprechend der Formulierung der Bewegungsgleichungen auf der Ebene von Geschwindigkeiten müssen auch Zwangsbedingungen auf der Ebene von Geschwindigkeiten formuliert werden. In vielen Fällen lassen sich solche, geschwindigkeitsbezogenen Zwangsbedingungen auch für ursprünglich holonome Zwangsbedingungen direkt formulieren, Kapitel 3.2 ab Seite 89 liefert entsprechende Beispiele.

Andernfalls kann diese Form durch einmaliges Ableiten der ursprünglichen Zwangsbedingung $C(\vec{x}, t)$ erreicht werden. Die Jacobimatrix einer Zwangsbedingung $\mathbf{J}$ taucht in der ersten wie in der zweiten Ableitung gleichermaßen auf. Entsprechende Beispiele liefern das 2D-Beispiel (Abschnitt 2.5.1 ab Seite 39) und das SCARA-Roboter-Beispiel im Anhang (Abschnitt B).

Die resultierenden, geschwindigkeitsbezogenen Zwangsbedingungen haben die allgemeine Form $\mathbf{J}\dot{\vec{x}} - b = a, a \geq 0$ (für eine einzelne Zwangsbedingung hat $\mathbf{J}$ nur eine Zeile). Alle Zwangsbedingungen können in folgender Matrix-Vektor-Form zusammengefasst werden:

$$\mathbf{J} \cdot \dot{\vec{x}} - \vec{b} = \vec{a}, \vec{a} \geq 0 \tag{2.32}$$

Fasst man nun Zwangsbedingungen 2.32 und diskretisierte Bewegungsgleichungen 2.31 zusammen, ergibt sich folgendes System:

$$\begin{pmatrix} \mathbf{M} & -\mathbf{J_e^T} & -\mathbf{J_n^T} \\ \mathbf{J}_e & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_n & \mathbf{0} & \mathbf{0} \end{pmatrix} \cdot \begin{pmatrix} \dot{\vec{x}}(t+h) \\ h\vec{\lambda}_e \\ h\vec{\lambda}_n \end{pmatrix} - \begin{pmatrix} \mathbf{M} \cdot \dot{\vec{x}}(t) + h\vec{f} \\ \vec{b}_e \\ \vec{b}_n \end{pmatrix} = \begin{pmatrix} \vec{0} \\ \vec{0} \\ \vec{a}_n \end{pmatrix} \tag{2.33}$$

mit den Komplementaritätsbedingungen:

$$\vec{\lambda}_n \geq \vec{0}, \quad \vec{a}_n \geq \vec{0}, \quad \vec{a}_n^T \cdot \vec{\lambda}_n = 0$$

Durch den Übergang von Beschleunigungen auf Geschwindigkeiten werden nun auch nicht mehr Zwangskräfte, sondern Zwangs*impulse* gesucht. Unmittelbar deutlich wird dies auch an der Tatsache, dass die Lösung des Systems 2.33 nicht mehr die Lagrange-Multiplikatoren und damit die Beträge der Zwangskräfte, sondern deren Produkt mit der

Zeitschrittweite h und damit die Zwangsimpulse liefert, die die Zwangskräfte über die Dauer des Zeitschrittes hinweg aufbringen.

Berücksichtigung von Coulombscher Kontaktreibung Nun fehlt der „großen“ Lagrange-Formulierung noch ein Modell zur Berücksichtigung von Kontaktreibung. Das Coulombsche Modell definiert eine Relation zwischen der Kraft f_n, die in Kontaktnormalenrichtung wirkt und der Reibungskraft (engl.: ***f**riction force*) f_f:

$$f_f \leq \mu f_n \tag{2.34}$$

Die Reibungskraft ist kleiner oder gleich der Normalenkraft multipliziert mit einem Reibungskoeffizienten μ, dessen Wertebereich typischerweise zwischen Null und Eins liegt.

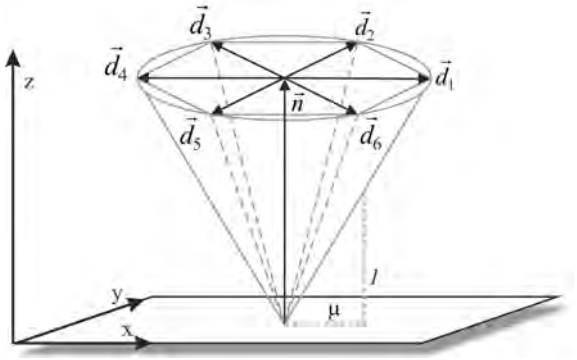


Abbildung 2.15.: Der diskretisierte Reibungskonus nach Stewart [126]

Da die Reibungskraft durch einen Konus mit kreisförmiger Grundfläche begrenzt wird (vgl. Anhang A.2), ergibt sich für entsprechende Zwangsbedingungen ein nichtlinearer Zusammenhang. Um dennoch eine einheitliche Formulierung in einem linearen System zu ermöglichen, verwenden sowohl Stewart und Trinkle [126, 127] als auch Anitescu et al. [6] einen linearisierten Reibungskonus, wie in Abbildung 2.15 dargestellt. Hierzu wird die eigentlich kreisförmige Grundfläche durch einen Polyeder approximiert, im Beispiel durch ein Sechseck. Die relative Kontaktgeschwindigkeit in der Kontaktebene kann dann in den diskreten Richtungen $\vec{d}_i$ durch lineare Geschwindigkeitszwangsbedingungen eingeschränkt werden. Wie Reibungszwangsbedingungen im Detail formuliert werden, behandelt Kapitel 3.3.2 ab Seite 107. Diese Linearisierung hat jedoch einen Fehler zur Folge: Rutschen die Körper im Kontakt aufeinander, müsste die Reibungs-

kraft genau der Richtung der relativen Geschwindigkeit im Kontakt entgegengesetzt wirken. Liegt die Richtung genau auf einem der diskreten Richtungsvektoren $\vec{d}_i$, erfüllt die Approximation genau das Coulombsche Modell und die Reibungskraft wird maximal. Liegt sie genau zwischen zwei diskreten Richtungen, wird die Reibungskraft minimal. Die Größe dieses Fehlers kann durch eine größere Anzahl diskreter Basisvektoren und damit Reibungszwangsbedingungen reduziert werden, allerdings auf Kosten der Vergrößerung der Dimension der Jacobimatrix und damit auch auf Kosten des Laufzeitverhaltens.

Die Relationen zwischen den Zwangskräften im Kontaktpunkt wird auf Zwangsimpulse $\vec{p}_{n_i}, \vec{p}_{f_i}$ übertragen: Der gesamte im i-ten Kontakt wirkende Impuls $\vec{p}_{n_i} + \vec{p}_{f_i}$ im zeitdiskreten System muss innerhalb des linearisierten Reibungskonus liegen. Entsprechend dem d'Alembert-Prinzip gilt für den Kontaktnormalenimpuls im i-ten Kontakt $\vec{p}_{n_i} = \mathbf{J}_{n_i}^T \lambda_{n_i} h$. Werden die Reibungszwangsbedingungen im i-ten Kontakt zusammengefasst zu $\mathbf{J}_{f_i}\dot{\vec{x}} - \vec{b} = \vec{a}$, dann gilt für den Gesamt-Kontaktreibungsimpuls $\vec{p}_{f_i} = \mathbf{J}_{f_i}^T \vec{\lambda}_{f_i} h$. Die Menge der Kontaktimpulse, die innerhalb des linearisierten Reibungskonus liegt, lässt sich demnach wie folgt formulieren:

$$\begin{aligned} &\{\mathbf{J_{n_i}^T} \cdot \lambda_{n_i} + \mathbf{J_{f_i}^T} \cdot \vec{\lambda}_{f_i} \;\mid\; \lambda_{n_i} \geq 0, \vec{\lambda}_{f_i} \geq \vec{0}, \vec{e} \cdot \vec{\lambda}_{f_i} \leq \mu_i \lambda_{n_i}\} \\ \text{mit} \quad &\vec{e} = \begin{pmatrix} 1 & 1 & \dots & 1 \end{pmatrix} \in \mathbb{R}^l \quad (\text{Zeilenvektor, } l : \text{Anzahl der Basisvektoren } \vec{d}_i) \end{aligned} \tag{2.35}$$

Die Formulierung in Gleichung 2.35 deckt bereits den Fall ab, wenn der Normalenimpuls verschwindet: Ist der Normalenimpuls Null, so muss auch der Reibungsimpuls Null sein, sonst wäre die Bedingung $\vec{e} \cdot \vec{\lambda}_{f_i} \leq \mu_i \lambda_{n_i}$ verletzt. Für die vollständige Realisierung des Coulombschen Reibungsmodells inklusive dynamischer und statischer Reibung sind folgende zusätzliche Komplementaritätsbedingungen erforderlich:

$$\begin{aligned} \gamma_i \vec{e}^T + \mathbf{J_{f_i}} \dot{\vec{x}}_i(t+h) \geq \vec{0} \quad &\text{komplementär zu} \quad \vec{\lambda}_{f_i} \geq \vec{0} \\ \mu_i \lambda_{n_i} - \vec{e} \cdot \vec{\lambda}_{f_i} \geq 0 \quad &\text{komplementär zu} \quad \gamma_i \geq 0 \end{aligned} \tag{2.36}$$

In dieser Formulierung ist der zusätzliche Lagrange-Multiplikator γ_i eine Hilfsgröße, die eine Annäherung an die relative tangentiale Geschwindigkeit im i-ten Kontaktpunkt repräsentiert [126].

Mit diesen Bedingungen sind nun auch die Fälle abgedeckt, in denen der Kontaktnormalenimpuls nicht verschwindet:

- Der Ausdruck $\mathbf{J_{f_i}} \dot{\vec{x}}_i(t+h)$ beschreibt die relativen tangentialen Geschwindigkei-

ten in den diskreten Richtungen des Reibungskonus. Im Falle von dynamischer Reibung, wenn ein Rutschen auftritt, enthält er echt positive sowie echt negative Werte, da die Richtungsvektoren stets paarweise entgegengesetzt zueinander definiert sind. Um die erste Größergleich-Bedingung zu erfüllen, muss der „Hilfsmultiplikator“ γ_i echt größer als Null sein. Dann fordert die zweite Komplementaritätsbedingung jedoch, dass der erste Ausdruck der zweiten Komplementaritätsbedingung gleich Null wird, d.h. es gilt $\mu_i \lambda_{n_i} = \vec{e}\vec{\lambda}_{f_i}$: Der Reibungsimpuls liegt auf dem Rand des Reibungskonus.

- Im Falle von statischer Reibung, wenn also keine tangentiale Bewegung im Kontakt auftritt, wird der Term $\mathbf{J_{f_i}}\dot{\vec{x}}_i(t+h) = \vec{0}$. Um die Größergleich-Bedingung der ersten Komplementaritätsbedingung zu erfüllen, muss der Hilfsmultiplikator γ_i gleich Null sein. Dadurch fordert die zweite Komplementaritätsbedingung für den ersten Ausdruck, dass er echt größer Null wird, d.h. es gilt $\vec{e} \cdot \vec{\lambda}_{f_i} < \mu_i \lambda_{n_i}$. Der Gesamtkontaktimpuls bewegt sich also innerhalb des Reibungskonus und nicht auf seinem Rand.

Die beschriebenen Zusammenhänge lassen sich nun für das gesamte Mehrkörpersystem mit mehreren Kontakten und Gelenken in folgendem, gemischtem LCP formulieren:

$$\begin{pmatrix} \mathbf{M} & -\mathbf{J_e^T} & -\mathbf{J_n^T} & -\mathbf{J_f^T} & \mathbf{0} \\ \mathbf{J}_e & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_n & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_f & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{e} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\mu} & -\mathbf{e^T} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \dot{\vec{x}}(t+h) \\ \vec{\lambda}_e \\ \vec{\lambda}_n \\ \vec{\lambda}_f \\ \vec{\gamma} \end{pmatrix} - \begin{pmatrix} \mathbf{M}\dot{\vec{x}}(t) + h\vec{f} \\ \vec{b}_e \\ \vec{b}_n \\ \vec{0} \\ \vec{0} \end{pmatrix} = \begin{pmatrix} \vec{0} \\ \vec{0} \\ \vec{a}_n \\ \vec{a}_f \\ \vec{a}_h \end{pmatrix}$$

mit den Komplementaritätsbedingungen

$$\begin{pmatrix} \vec{a}_n \\ \vec{a}_f \\ \vec{a}_h \end{pmatrix} \geq 0, \begin{pmatrix} \vec{\lambda}_n \\ \vec{\lambda}_f \\ \vec{\gamma} \end{pmatrix} \geq 0, \begin{pmatrix} \vec{a}_n \\ \vec{a}_f \\ \vec{a}_h \end{pmatrix}^T \cdot \begin{pmatrix} \vec{\lambda}_n \\ \vec{\lambda}_f \\ \vec{\gamma} \end{pmatrix} = 0 \tag{2.37}$$

$$\text{mit } \mathbf{e} = \begin{pmatrix} \vec{e}^T & & \\ & \ddots & \\ & & \vec{e}^T \end{pmatrix} \text{ und } \boldsymbol{\mu} = \begin{pmatrix} \mu_1 & & \\ & \ddots & \\ & & \mu_n \end{pmatrix}$$

Hierin ist μ_i der Reibungskoeffizient im i-ten Kontakt.

Hierbei handelt es sich wieder um ein gemischtes lineares Komplementaritätsproblem. Die Formulierung besitzt die positive Eigenschaft, dass das komplexe Problem der Kontaktreibung durch eine bekannte mathematische Formulierung approximiert werden kann. Der Grad der Approximation kann dabei durch die Anzahl der Basisvektoren des linearisierten Reibungskonus beliebig erhöht werden.

Der „kleine“ Lagrange-Multiplikatoren-Ansatz

Lötstedt [81] bringt Bewegungsgleichungen und Zwangsbedingungen auf anderem Wege zusammen. Anstelle der Formulierung eines gemeinsamen Systems wendet er die Zwangsbedingungen direkt auf die Beschleunigungen des Mehrkörpersystems an. Zunächst wird Gleichung 2.26 nach den Beschleunigungen umgestellt:

$$\begin{aligned} \mathbf{M} \cdot \ddot{\vec{x}} &= \vec{f} + \mathbf{J}^T \cdot \vec{\lambda} \\ \Rightarrow \ddot{\vec{x}} &= \mathbf{M}^{-1}\vec{f} + \mathbf{M}^{-1}\mathbf{J}^T \cdot \vec{\lambda} \end{aligned} \tag{2.38}$$

Dann werden die beschleunigungsbezogenen Zwangsbedingungen (wiederum zunächst nur bilaterale) direkt auf den resultierenden Term für $\ddot{\vec{x}}$ angewendet:

$$\begin{aligned} \mathbf{J}_e\ddot{\vec{x}} &= \vec{b}_e \\ \Rightarrow \mathbf{J}_e\mathbf{M}^{-1}\mathbf{J}_e^T \cdot \vec{\lambda}_e + \mathbf{J}_e\mathbf{M}^{-1}\vec{f} &= \vec{b}_e \end{aligned} \tag{2.39}$$

Bei Verwendung ausschließlich bilateraler Zwangsbedingungen gelangt man auf diesem Wege wieder zu einem linearen Gleichungssystem. Der Lösungsvektor enthält bei dieser Formulierung jedoch nur noch die Beträge der Zwangskräfte, nicht mehr die Beschleunigungen der Körper. Die Dimension der quadratischen Systemmatrix $\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$ ist gleich der Anzahl der Zwangsbedingungen.

Zur Invertierbarkeit von M: M ist eine Blockdiagonalmatrix, die entlang ihrer Hauptdiagonalen für jeden Körper dreimal seine Masse m und einmal seinen 3×3-Trägheitstensor $\mathbf{\Theta}$ enthält. Blockdiagonalmatrizen werden blockweise invertiert. Trägheitstensoren sind invertierbar, weil ihre Eigenwerte die Hauptträgheitsmomente (vgl. Anhang A.1) und damit immer positiv sind. Diagonalmatrizen mit skalaren Werten (m_i) sind genau dann invertierbar, wenn diese ungleich Null sind. Für die Inverse ergibt sich

daher:

$$\mathbf{M} = \begin{pmatrix} m_1 \cdot \mathbf{E} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Theta}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ & & \ddots & & \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & m_n \cdot \mathbf{E} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{\Theta}_n \end{pmatrix}, \quad \mathbf{M}^{-1} = \begin{pmatrix} \frac{1}{m_1} \cdot \mathbf{E} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Theta}_1^{-1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ & & \ddots & & \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{1}{m_n} \cdot \mathbf{E} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{\Theta}_n^{-1} \end{pmatrix}$$

mit $\mathbf{E} \in \mathbb{R}^{3\times 3}$ der Einheitsmatrix. (2.40)

Anwendung auf das 2D-Beispiel

Das zu bildende System für das 2D-Beispiel hat die Form (vgl. Gleichung 2.18 auf Seite 42):

$$\begin{aligned} \underbrace{\begin{pmatrix} 2 & -1 \end{pmatrix}}_{\mathbf{J}} \underbrace{\begin{pmatrix} \frac{1}{m} & 0 \\ 0 & \frac{1}{m} \end{pmatrix}}_{\mathbf{M}^{-1}} \underbrace{\begin{pmatrix} 2 \\ -1 \end{pmatrix}}_{\mathbf{J}^T} \vec{\lambda} + \begin{pmatrix} 2 & -1 \end{pmatrix} \begin{pmatrix} \frac{1}{m} & 0 \\ 0 & \frac{1}{m} \end{pmatrix} \begin{pmatrix} f_{\text{ext},1} \\ f_{\text{ext},2} \end{pmatrix} &= 0 \\ \Rightarrow \frac{1}{m} 5\vec{\lambda} + \frac{1}{m}(2f_{\text{ext},1} - f_{\text{ext},2}) &= 0 \end{aligned} \tag{2.41}$$

Lösung nach $\vec{\lambda}$ liefert:

$$\vec{\lambda} = \frac{f_{\text{ext},2} - 2f_{\text{ext},1}}{5} \tag{2.42}$$

Setzt man $\vec{\lambda}$ wiederum in die Bewegungsgleichungen (vgl. Gl. 2.38) ein, erhält man für die Beschleunigungen $\ddot{x}_1, \ddot{x}_2$ dieselben Ausdrücke wie beim „großen“ Ansatz:

$$
\begin{aligned}
\underbrace{\begin{pmatrix}\ddot{x}_1\\ \ddot{x}_2\end{pmatrix}}_{\ddot{\vec{x}}} &= \underbrace{\begin{pmatrix}\frac{1}{m} & 0\\ 0 & \frac{1}{m}\end{pmatrix}}_{\mathbf{M}^{-1}} \left(\underbrace{\begin{pmatrix}f_{\text{ext},1}\\ f_{\text{ext},2}\end{pmatrix}}_{\vec{f}_{\text{ext}}} + \underbrace{\begin{pmatrix}2\\ -1\end{pmatrix}}_{\mathbf{J}^T} \underbrace{\left(\frac{f_{\text{ext},2}-2f_{\text{ext},1}}{5}\right)}_{\vec{\lambda}} \right) \\
\ddot{x}_1 &= \frac{2f_{\text{ext},2}+f_{\text{ext},1}}{5m} \\
\ddot{x}_2 &= \frac{4f_{\text{ext},2}+2f_{\text{ext},1}}{5m}
\end{aligned}
\tag{2.43}
$$

Übergang zu impulsbasierten Formulierungen Stewart und Trinkle [126] führen für den „kleinen“ Ansatz die gleiche Diskretisierung der Bewegungsgleichungen ein wie Anitescu und Potra für den „großen“ [6]. Ersetzt man in Gleichung 2.38 den Term der Beschleunigungen $\ddot{\vec{x}}$ durch einen entsprechenden Differenzenquotienten, gelangt man zu folgender Vorschrift für die Geschwindigkeiten im nächsten Zeitschritt:

$$
\dot{\vec{x}}(t+h) = \dot{\vec{x}}(t) + \mathbf{M}^{-1}\mathbf{J}^T h\vec{\lambda} + \mathbf{M}^{-1}h\vec{f} \tag{2.44}
$$

Die Anwendung bilateraler, geschwindigkeitsbezogener Zwangsbedingungen auf 2.44 führt zu folgendem linearen Gleichungssystem:

$$
\begin{aligned}
\mathbf{J}_e \cdot \dot{\vec{x}}(t+h) &= \vec{b}_e \\
\Rightarrow \mathbf{J}_e\mathbf{M}^{-1}\mathbf{J}_e^T h\vec{\lambda}_e + \mathbf{J}_e\left(\dot{\vec{x}} + \mathbf{M}^{-1}h\vec{f}\right) &= \vec{b}_e
\end{aligned}
\tag{2.45}
$$

Analog zur Einbringung bilateraler Zwangsbedingungen zeigen Stewart und Trinkle [126] sowie Sauer und Schömer [113], wie unilaterale Kontakt-Zwangsbedingungen der Form
$\mathbf{J}_n \cdot \dot{\vec{x}}(t+h) - b_n = a_n, a_n \geq 0$ im „kleinen“ Lagrange-Multiplikatoren-Ansatz berücksichtigt

werden können. Ergebnis ist wiederum ein lineares Komplementaritätsproblem:

$$\begin{aligned} &\mathbf{J}_n\mathbf{M}^{-1}\mathbf{J}_n^T h\vec{\lambda}_n + \mathbf{J}_n\left(\dot{\vec{x}} + \mathbf{M}^{-1}h\vec{f}\right) - \vec{b}_n = \vec{a}_n \\ &\qquad \text{mit der Komplementaritätsbedingung} \\ &\qquad\qquad \vec{\lambda}_n \geq \vec{0},\ \ \vec{a}_n \geq \vec{0},\ \ \vec{\lambda}_n^T\vec{a_n} = 0 \end{aligned} \tag{2.46}$$

Reibungszwangsbedingungen werden wie beim „großen“ Ansatz durch einen linearisierten Reibungskonus berücksichtigt. Bilaterale, Kontakt- und Reibungszwangsbedingungen können zusammengefasst werden, indem schlicht alle Zwangsbedingungen eine gemeinsame Jacobimatrix eingehen:

$$\begin{aligned} &\begin{pmatrix}\mathbf{J}_e\\ \mathbf{J}_n\\ \mathbf{J}_f\end{pmatrix}\mathbf{M}^{-1}\begin{pmatrix}\mathbf{J}_e^T & \mathbf{J}_n^T & \mathbf{J}_f^T\end{pmatrix}h\begin{pmatrix}\vec{\lambda}_e\\ \vec{\lambda}_n\\ \vec{\lambda}_f\end{pmatrix} + \begin{pmatrix}\mathbf{J}_e\\ \mathbf{J}_n\\ \mathbf{J}_f\end{pmatrix}\left[\dot{\vec{x}} + \mathbf{M}^{-1}h\vec{f}\right] - \begin{pmatrix}\vec{b}_e\\ \vec{b}_n\\ \vec{b}_f\end{pmatrix} = \begin{pmatrix}\vec{0}\\ \vec{a}_n\\ \vec{a}_f\end{pmatrix} \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{mit } \begin{pmatrix}\vec{a}_n\\ \vec{a}_f\end{pmatrix} \geq \vec{0} \end{aligned} \tag{2.47}$$

Genau wie beim „großen“ Ansatz sind die zusätzlichen Komplementaritätsbedingungen aus den Gleichungen 2.35 und 2.36 mit dem zusätzlichen Hilfsmultiplikatoren $\vec{\gamma}$ erforderlich. Multipliziert man 2.47 aus und ergänzt die Bedingungen zur Einhaltung des Coloumbschen Reibungsgesetzes, so gelangt man zu:

$$\begin{pmatrix}\mathbf{J}_e\mathbf{M}^{-1}\mathbf{J}_e^T & \mathbf{J}_e\mathbf{M}^{-1}\mathbf{J}_n^T & \mathbf{J}_e\mathbf{M}^{-1}\mathbf{J}_f^T & \mathbf{0}\\ \mathbf{J}_n\mathbf{M}^{-1}\mathbf{J}_e^T & \mathbf{J}_n\mathbf{M}^{-1}\mathbf{J}_n^T & \mathbf{J}_n\mathbf{M}^{-1}\mathbf{J}_f^T & \mathbf{0}\\ \mathbf{J}_f\mathbf{M}^{-1}\mathbf{J}_e^T & \mathbf{J}_f\mathbf{M}^{-1}\mathbf{J}_n^T & \mathbf{J}_f\mathbf{M}^{-1}\mathbf{J}_f^T & \mathbf{E}\\ \mathbf{0} & \boldsymbol{\mu} & -\mathbf{E}^T & \mathbf{0}\end{pmatrix}\begin{pmatrix}\vec{\lambda}_e\\ \vec{\lambda}_n\\ \vec{\lambda}_f\\ \vec{\gamma}\end{pmatrix} + \begin{pmatrix}\mathbf{J}_e\\ \mathbf{J}_n\\ \mathbf{J}_f\\ \mathbf{0}\end{pmatrix}\left[\dot{\vec{x}} + \mathbf{M}^{-1}h\vec{f}\right] - \begin{pmatrix}\vec{b}_e\\ \vec{b}_n\\ \vec{b}_f\\ \vec{0}\end{pmatrix} = \begin{pmatrix}\vec{0}\\ \vec{a}_n\\ \vec{a}_f\\ \vec{a}_h\end{pmatrix} \tag{2.48}$$

mit den Komplementaritätsbedingungen

$$\begin{pmatrix}\vec{a}_n\\ \vec{a}_f\\ \vec{a}_h\end{pmatrix} \geq \vec{0},\quad \begin{pmatrix}\vec{\lambda}_n\\ \vec{\lambda}_f\\ \vec{\gamma}\end{pmatrix} \geq \vec{0},\quad \begin{pmatrix}\vec{a}_n\\ \vec{a}_f\\ \vec{a}_h\end{pmatrix}^T\begin{pmatrix}\vec{\lambda}_n\\ \vec{\lambda}_f\\ \vec{\gamma}\end{pmatrix} = 0 \tag{2.49}$$

Wie in 2.37 handelt es sich hierbei nicht mehr um ein „reines“, sondern um ein gemisch-

tes lineares Komplementaritätsproblem. Zu seiner Lösung werden daher die gleichen Algorithmen wie beim großen Ansatz verwendet.

Nichtlineare Verfahren mit Lagrange-Multiplikatoren

Weit weniger verbreitet sind nichtlineare Ansätze mit Lagrange-Multiplikatoren, zweifelsohne ob ihrer äußerst komplexen mathematischen Formulierung. Ziel solcher Formulierungen ist die Vermeidung der Linearisierung des Reibungskonus.

Anitescu und Tasora beschreiben in [8] und [130] ein Verfahren, in dem sie anstelle eines LCP ein konisches Komplementaritätsproblem (engl.: *Cone Complementarity Problem, CCP*) formulieren und lösen. Ihr Verfahren konzentriert sich auf die Anwendung in der Granulat- oder Schüttgutsimulation und damit auf die Simulation sehr vieler, kugelförmiger Körper. Das Verfahren wird von Tasora et al. [131] parallelisiert und von Heyn et al. [60] in einer Version für die heute weit verbreiteten, massiv parallelen Grafikprozessoren vorgestellt.

Ein Vergleich des nichtlinearen Verfahrens nach Tasora und Anitescu mit den linearen anhand der veröffentlichten Ergebnisse ist schwierig, allerdings sind die präsentierten Laufzeiten für das spezielle Problem der Granulatsimulation mit allen anderen heute verfügbaren Verfahren unerreichbar. Vor allem die Nutzung der enormen theoretischen Rechenleistung moderner, massiv paralleler Streaming-Prozessoren auf Grafikkarten erscheint erfolgversprechend. Ließe sich dieses Verfahren auch auf interaktive Simulationen mit beliebigen Zwangsbedingungen und beliebig geformten Körpern übertragen, bedeutete das einen großen Schritt im Bereich der Mehrkörpersimulationsverfahren.

2.5.6. Verfahren in reduzierten Koordinaten

Eine Zwangsbedingung entfernt genau einen Freiheitsgrad aus einem Mehrkörpersystem. Ein Mehrkörpersystem mit n Körpern und m Zwangsbedingungen besitzt also nur noch $6n - m$ Freiheitsgrade. Ziel der Verfahren in reduzierten Koordinaten ist es, einen Satz von $6n - m$ unabhängigen Koordinaten zu finden, mit deren Hilfe sich die kinematischen und dynamischen Zusammenhänge eines Mehrkörpersystems unter impliziter Berücksichtigung aller Zwangsbedingungen beschreiben lassen.

Bender [17] liefert eine gute Zusammenfassung über die Schritte, die im Allgemeinen notwendig sind, um zu einer Formulierung der Bewegungsgleichungen eines Mehrkörpersystems in reduzierten Koordinaten zu gelangen. Zunächst sind die $6n-m$ unabhän-

gigen Koordinaten $q_i, i \in [1, \ldots, 6n-m]$ zu identifizieren, die den Bewegungszustand des Mehrkörpersystems eindeutig beschreiben. Als nächstes müssen die kartesischen Koordinaten x_i durch diese generalisierten Koordinaten ausgedrückt werden in Form einer Abbildung:

$$x_i = f\left(q_1, \ldots, q_{6n-m}\right) \tag{2.50}$$

Hierbei handelt es sich um das Problem der Vorwärts-Kinematik. Der letzte Schritt ist die eigentliche Bestimmung der Bewegungsgleichungen, die in Form eines Differenzialgleichungssystems zweiter Ordnung kommen:

$$\ddot{\vec{q}} = g\left(\vec{q}, \dot{\vec{q}}, \vec{f}_{ext}\right), \quad \vec{f}_{ext} : \text{Externe Kräfte und Drehmomente} \tag{2.51}$$

Es bestehen zwei grundlegende Ansätze, um die Bewegungsgleichungen zu bestimmen.

Bestimmung von Bewegungsgleichungen nach Newton und Euler in reduzierten Koordinaten durch Freischneiden

Beim Freischneiden betrachtet man zunächst jeden Körper des Systems individuell in allen sechs Freiheitsgraden (im $\mathbb{R}^3$). Die Interaktion mit anderen Körpern des Systems wird durch Zwangskräfte berücksichtigt, die je an den Verbindungspunkten eingebracht werden. Für jeden so freigeschnittenen Körper können unabhängig vom restlichen System die Bewegungsgleichungen formuliert werden. Da die formal eingesetzten Zwangskräfte unbekannt sind, müssen sie durch algebraische Umformung eliminiert werden. Sind alle Zwangskräfte und -momente eliminiert, bleiben die Bewegungsgleichungen in den reduzierten Koordinaten „übrig“.

Bestimmung von Bewegungsgleichungen in reduzierten Koordinaten unter Nutzung des Lagrange-Gleichung 2. Art

Für ein Mehrkörpersystem hat die Lagrange-Gleichung 2. Art [76] die Form:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = Q_i$$

Hierin ist die Lagrange-Funktion $L = E_{\mathrm{kin}} - E_{\mathrm{pot}}$ die Differenz aus kinetischer und potenzieller Energie des gesamten Mehrkörpersystems formuliert in den generalisierten Koordinaten, q_i die i-te generalisierte Koordinate und Q_i die im Sinne der i-ten Koordinate wirkende, eingeprägte (externe) Kraft bzw. das eingeprägte Moment. Ausführen der Ableitungsvorschriften auf der linken Seite der Gleichung führt auf die Bewegungsgleichungen des Mehrkörpersystem in reduzierten Koordinaten. Detailliertere Ausführungen liefern u.a. Jalon und Bayo [67].

Automatisierte Verfahren in reduzierten Koordinaten Zur Anwendung in Computersimulationen sind automatisierte Verfahren gefragt. Featherstone [47, 46] liefert einen guten Überblick über die im Anwendungsbereich Roboterdynamik eingesetzten Verfahren. Eines der prominentesten ist der rekursive Newton-Euler-Algorithmus (engl.: *Recursive Newton Euler Algorithm, RNEA*), den Luh et al. [82] vorstellen. Featherstone [45] präsentiert den „*Articulated Body Algorithm, ABA*“, der sich durch sein lineares Laufzeitverhalten in Bezug auf n verbundene Glieder $O(n)$ auszeichnet.

Beide genannten Algorithmen können nur Mehrkörpersysteme behandeln, die frei von Zyklen sind und ausschließlich holonomen Zwangsbedingungen unterliegen. Es existiert eine große Anzahl weiterer Arbeiten, die sich der Herleitung von Bewegungsgleichungen in reduzierten Koordinaten widmet, die an dieser Stelle nicht weiter behandelt werden. Der interessierte Leser sei als Einstiegspunkt auf die Arbeiten von Featherstone [45, 47, 46] verwiesen. Aufgrund der Tatsache, dass zur Behandlung geschlossener Kreise und Erfüllung anholonomer Zwangsbedingungen Sonderbehandlungen notwendig werden, haben sie sich bis heute für den breiten Einsatz in der interaktiven Virtuellen Realität, in denen häufig sehr dynamische, sich schnell verändernde Konfigurationen zu behandeln sind, nicht durchgesetzt. Für Anwendungen mit gleichbleibenden oder sich nur geringfügig verändernden Zwangsbedingungen, z.B. im Bereich der Robotik, bieten sie jedoch einige Vorteile. Der Verzicht auf die Betrachtung der wirkenden Zwangskräfte und die implizite Annahme der Gültigkeit aller Zwangsbedingungen resultiert durch die von Beginn an kleinere Anzahl an Freiheitsgraden in einem deutlich kleineren mathematischen Problem. Die Bestimmung der wirkenden Zwangskräfte ist nicht notwendig.

2.6. Animation der Bewegungsgleichungen

Die Bewegungsgleichungen nach Newton und Euler lassen sich in Matrix-Vektor-Form durch folgendes Differenzialgleichungssystem zweiter Ordnung darstellen:

$$\ddot{\vec{x}} = \mathbf{M}^{-1}(\vec{x}) \cdot \vec{f}(\vec{x}, t)$$

Dieses System lässt sich durch Einführung der Geschwindigkeiten $\vec{v} = \dot{\vec{x}}$ in ein Differenzialgleichungssystem erster Ordnung überführen:

$$\begin{aligned} \dot{\vec{v}} &= \mathbf{M}^{-1}(\vec{x}) \cdot \vec{f}(\vec{x}, t) \\ \dot{\vec{x}} &= \vec{v} \end{aligned} \tag{2.52}$$

Bei gegebenen Anfangswerten für Position und Orientierungen ($\vec{x}$) und Geschwindigkeiten ($\vec{v}$), z.B.

$$\begin{aligned} \vec{v}(t=0) &= \vec{0} \\ \vec{x}(t=0) &= \vec{0} \end{aligned} \tag{2.53}$$

erhält man ein Anfangswertproblem (AWP). Ziel der Animation der Bewegungsgleichungen bzw. der Lösung des Anfangswertproblems ist die Bestimmung von Positionen, Orientierungen und Geschwindigkeiten zu einem beliebigen, späteren Zeitpunkt $t_n = t + n \cdot h$. Zur Lösung eines AWPs kommen Verfahren der numerischen Integration zum Einsatz. Die drei wesentlichen Kriterien zu ihrer Bewertung sind Genauigkeit bzw. Konsistenz, Stabilität und Berechnungsaufwand. Letzterer ergibt sich aus der Anzahl der Rechenoperationen, die zur Bestimmung des jeweils nächsten Integrationsschrittes notwendig sind. Die ersten beiden hängen eng zusammen und lassen sich formal definieren. Seien im Folgenden $f(x)$ das mathematische Problem mit den Eingabedaten x, hier also die Bewegungsgleichung, $f^*(x)$ das numerische Verfahren und x^* fehlerbehaftete Eingabedaten. Dann gilt für die Begriffe Genauigkeit und Stabilität:

- **Die Genauigkeit bzw. Konsistenz** gibt an, wie gut sich die numerische Lösung des Problems der exakten Lösung annähert. Der Fehler ist definiert als: $e_K = ||f(t) - f^*(t)||$. Die Konsistenz wird durch den Begriff der *Ordnung* eines Verfahrens angegeben: Ein Verfahren n-ter Ordnung begeht einen lokalen Fehler der Größenordnung t^{n+1}. Beispiel: Bei einem Verfahren erster Ordnung reduziert

sich der lokale Fehler bei Halbierung der Zeitschrittweite t um den Faktor $2^2 = 4$.

- **Die Stabilität** eines Verfahrens beschreibt seine Robustheit gegenüber Störungen in den Eingabedaten. Formal betrachtet man den Wert $e_S = ||f(x^*) - f^*(x^*)||$. Bei einem stabilen numerischen Verfahren lässt sich für eine gegebene Differenzialgleichung ein endlicher Grenzwert definieren, unterhalb dessen der Fehler zwischen numerischer und exakter Lösung liegen wird.

Im Bereich Dynamiksimulation für interaktive VR-Anwendungen wird den unterschiedlichen numerischen Integrationsverfahren relativ wenig Aufmerksamkeit geschenkt. Dies hat drei wesentliche Gründe:

1. Die hohe Genauigkeit der Integration spielt in diesem Kontext oft nur eine untergeordnete Rolle. Viele andere Faktoren, z.B. die Fehler bei der Bestimmung der Zwangskräfte, schränken die Genauigkeit der Simulation in viel größerem Maße ein.

2. Ein gutes Laufzeitverhalten ist für interaktive Realzeit-Anwendungen ein essenzielles Kriterium. Während ein einfaches, explizites Euler-Verfahren erster Ordnung durch Bestimmung einer einzelnen algebraischen Anweisung auszuführen ist, wird für Verfahren mit höherer Genauigkeit ein Vielfaches an Operationen notwendig. Ist für einen Integrationsschritt gar die Ausführung der in der Regel extrem rechenaufwendigen inversen Dynamik zur Bestimmung wirkender Zwangskräfte zu unterschiedlichen Zeitpunkten und für unterschiedliche Ausgangskonfigurationen notwendig, wird ein entsprechendes numerisches Verfahren für interaktive, echtzeitfähige Anwendungen unpraktikabel.

3. Die Verfahren zur Bestimmung der inversen Dynamik sind häufig eng verknüpft mit der Animation der Bewegungsgleichungen, so dass deren strikte Separation und dadurch die Anwendung unterschiedlicher numerischer Integrationsverfahren gar nicht möglich sind.

Alle heute üblichen Verfahren, die zur Bestimmung der inversen Dynamik den Lagrange-Multiplikatoren-Ansatz nutzen, verwenden - prinzipbedingt - ausnahmslos teilweise („semi-“) [126, 6, 113, 114, 115, 127, 128, 131] oder implizite [14, 7, 35, 34] Euler-Integrationsschemata. Zur Herleitung des *impliziten Euler-Integrationsschemas*

beginnt man zunächst mit einer Rückwärts-Euler-Diskretisierung der Bewegungsgleichungen:

$$\begin{aligned} \frac{\vec{v}(t+h) - \vec{v}(t)}{h} &= \mathbf{M}^{-1}(\vec{x}(t+h)) \cdot \vec{f}(x(t+h), v(t+h), t+h) \\ \frac{\vec{x}(t+h) - \vec{x}(t)}{h} &= \vec{v}(t+h) \end{aligned} \tag{2.54}$$

Umgestellt nach den Geschwindigkeiten $\vec{v}(t+h)$ und den Positionen und Orientierungen $\vec{x}(t+h)$ im nächsten Zeitschritt erhält man entsprechende Animationsvorschriften:

$$\begin{aligned} \vec{v}(t+h) &= \vec{v}(t) + h\mathbf{M}^{-1}(\vec{x}(t+h)) \cdot \vec{f}(x(t+h), v(t+h), t+h) \\ \vec{x}(t+h) &= \vec{x}(t) + h\vec{v}(t+h) \end{aligned} \tag{2.55}$$

In diesen Vorschriften sind die Größen der rechten Seiten leider nur *implizit* gegeben. Die inversen Trägheitstensoren formuliert im Weltkoordinatensystem (enthalten in $\mathbf{M}^{-1}$) wie auch alle auftretenden Kräfte, externe wie Zwangskräfte (enthalten in $\vec{f}(x(t+h), v(t+h), t+h)$) sind abhängig von den *neuen* Positionen und Orientierungen und den *neuen* Geschwindigkeiten, die gerade erst bestimmt werden sollen. Würden die rechten Seiten stattdessen zum Zeitpunkt t ausgewertet, handelte es sich um ein *explizites* Integrationsschema.

Lösen lässt sich das Problem der nur implizit gegebenen Größen durch eine Fixpunktiteration: Die rechte Seite der ersten Gleichung in 2.55 wird zunächst zum Zeitpunkt t ausgewertet, Ergebnis ist der erste Schätzwert für $\vec{v}_1(t+h)$. Mit diesem lässt sich ein erster Schätzwert $\vec{x}_1(t+h)$ bestimmen. Mit diesen Schätzwerten kann die rechte Seite der ersten Gleichung erneut ausgewertet werden, um zu neuen Schätzwerten $\vec{v}_2(t+h)$ und $\vec{x}_2(t+h)$ zu gelangen. Die Iteration läuft so lange, bis sie den *Fixpunkt* ausreichend genau erreicht hat, das heißt die Differenz der aufeinanderfolgenden Schätzwerte eine gegebene Fehlerschranke unterschreitet.

Die implizite Euler-Integration hat die Eigenschaft, dass sie *unbedingt stark stabil* ist. Übertragen auf den Kontext von Bewegungsgleichungen bedeutet dies, das System verhält sich immer dissipativ, unabhängig davon wie groß die Zeitschrittweite h gewählt wird. Dies kommt dem Wunsch nach großen Schrittweiten in der realzeitfähigen Simulation entgegen.

Im Kontext von Mehrkörperdynamiksimulation kann die mehrfache Bestimmung der wirkenden Kräfte und Zwangskräfte (rechte Seite der ersten Zeile in Gleichung 2.55)

z.B. bei Einsatz einer Lagrange-Multiplikatoren-Methode allerdings sehr aufwändig und damit unpraktikabel sein. Der Mehraufwand hierfür kann sich lohnen, wenn sogenannte steife Differenzialgleichungen zu behandeln sind. Das ist z.B. der Fall, wenn Federn mit großen Federkonstanten simuliert werden. Die wirkenden Kräfte ändern sich dann sehr stark mit nur kleinen Veränderungen im Positionsvektor $\vec{x}$. Anitscu und Potra [7] stellen eine entsprechende Formulierung mit Lagrange-Multiplikatoren und impliziter Integration vor, Cline [35] beschreibt ihre Implementierung.

Um die Schwierigkeiten bei impliziten Verfahren zu umgehen, wird häufig das semi-implizite oder symplektische Euler-Verfahren angewandt. Dazu wird nur in der ersten Gleichung von 2.54 eine Vorwärts-Euler-Diskretisierung eingesetzt. Das bedeutet, dass die rechte Seite nur der ersten Zeile in Gleichung 2.55 zum Zeitpunkt t anstelle $t + h$ ausgewertet wird. Ein iteratives Lösungsverfahren ist dann nicht notwendig.

Auch beim Einsatz anderer Verfahren zur Bestimmung der wirkenden Zwangskräfte oder -impulse im Kontext von VR-Anwendungen werden in der Literatur selten aufwändigere Integrationsschemata verwendet, die meisten Realisierungen nutzen das Euler-Schema. Schmitt und Bender [22, 21, 17, 18] verwenden in ihrem impulsbasierten Verfahren einen Runge-Kutta-Integrator vierter Ordnung. Schmitt [119, 120] entwickelt darüber hinaus impulsbasierte Verfahren noch höherer Ordnungen, kommt jedoch zu dem Schluss, dass der erhaltene Nutzen den notwendigen Aufwand nicht rechtfertigt.

2.7. Verfahren zur Fehlerkorrektur

Bei den Verfahren zur Bestimmung der inversen Dynamik, die nur Zwangsbedingungen auf der Ebene von Geschwindigkeiten oder gar Beschleunigungen berücksichtigen können, werden positions- und orientierungsbezogene Zwangsbedingungen nur indirekt über ihre Ableitungen berücksichtigt. Durch die unvermeidliche Ungenauigkeit aufgrund der zeitlichen Diskretisierung treten kleine Fehler in den positions- und orientierungsbezogenen Zwangsbedingungen auf, die mithilfe eines zusätzlichen Stabilisierungsschritt behandelt werden müssen.

2.7.1. Stabilisierung von Zwangsbedingungen

Das einfachste und bekannteste Verfahren ist die Stabilisierung von Zwangsbedingungen („Constraint-Stabilisierung“) nach Baumgarte [16], eingesetzt u.a. in der Open Dy-

namics Engine [121]. Die Idee ist einfach: Ein Anteil des entstandenen Fehlers wird als Pseudo-Soll-Geschwindigkeit in der geschwindigkeitsbezogenen Zwangsbedingung berücksichtigt. Abbildung 2.16 verdeutlicht das Verfahren. Bei einer Kontaktzwangsbedingung stellt eine positive Durchdringungstiefe e den Fehler in der ursprünglichen, positionsbezogenen Zwangsbedingung dar. Die Zwangsbedingung für den nächsten Zeitschritt wird entsprechend angepasst, so dass für die relative Geschwindigkeit im Kontaktpunkt entlang der Kontaktnormalen zwischen den Körpern eine echt positive (> 0) anstelle einer verschwindenden Geschwindigkeit ($= 0$) erzwungen wird, die beide Körper auseinander bewegt. Formal bedeutet dies: Ist der Fehler im Kontaktpunkt Null, so wäre die Zwangsbedingung zum Kontaktpunkt:

$$C(\vec{v}_1, \vec{v}_2) = \vec{n} \cdot (\vec{v}_1 - \vec{v}_2) \geq 0 \tag{2.56}$$

Bei einer bestehenden Durchdringung der Tiefe e erzwingt die folgende modifizierte Zwangsbedingung eine relative Kontaktnormalengeschwindigkeit, die den bestehenden Fehler in 10 Zeitschritten der Länge h vollständig korrigieren würde:

$$C(\vec{v}_1, \vec{v}_2) = \vec{n} \cdot (\vec{v}_1 - \vec{v}_2) \geq \frac{e}{10h} \tag{2.57}$$

Dieses Verfahren ist für jede Art von positions- oder orientierungsbezogener Zwangsbedingung in einem geschwindigkeits- oder beschleunigungsbasierten Modell anwendbar. Da es praktisch keinerlei Mehraufwand bedeutet, besitzt es ein äußerst günstiges Laufzeitverhalten.

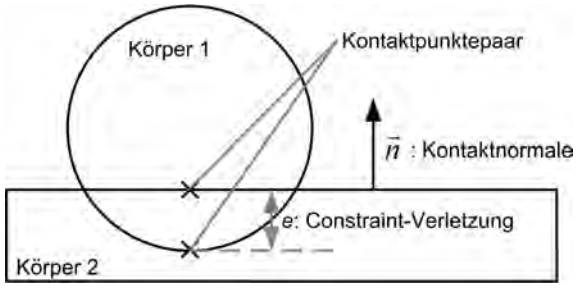


Abbildung 2.16.: Constraint-Stabilisierung nach Baumgarte [16]: Ein Anteil des entstandenen Fehlers wird als Pseudo-Soll-Geschwindigkeit auf die Zwangsbedingung aufgeschlagen.

Seine Effektivität ist jedoch eingeschränkt: Auch durch die Stabilisierung werden die ursprünglichen, positions- und orientierungsbezogenen Zwangsbedingungen niemals

direkt korrigiert, es wird immer nur eine korrigierende Tendenz erzeugt. Da der Korrekturparameter (im Beispiel $1/10$) typischerweise global definiert wird, kann er niemals ideal für jede mögliche Konfiguration einer Szene sein. Typische Problemmodelle sind lange Ketten oder hohe Stapel. In solchen Modellen werden an verschiedenen Kontakt- oder Gelenkpunkten immer unterschiedlich große Fehler bestehen bleiben. Ein zu kleiner Korrekturparameter hinterlässt große Fehler, ein zu großer Korrekturparameter führt zu Überschwingen und damit z.B. zum „Abprallen“ in einem eigentlich unflexiblen Kontakt. Dennoch: In der Praxis, in der solche Problemmodelle eine untergeordnete Rolle spielen, ist dieses Verfahren äußerst tauglich und wird häufig eingesetzt.

2.7.2. Fehlerkorrektur durch Projektion

Die Motivation für das Projektionsverfahren liegt in der Schwäche des Stabilisierungsverfahrens begründet: Während bei der Stabilisierung immer nur eine „verbessernde Tendenz“ erzeugt werden kann, sollen Projektionsverfahren alle bestehenden Verletzungen von Zwangsbedingungen in einem Schritt vollständig beheben. Dazu werden Positionen und Orientierungen des Mehrkörpersystems in einen gültigen Unterraum projiziert.

Cline und Pai [34] beschreiben einen besonders praktischen Ansatz unter dem Begriff „*Post-Stabilization*“. Ausgehend von einer Lagrange-Multiplikatoren Formulierung wie in Gleichung 2.37, entwickeln sie einen Nachverarbeitungsschritt, der alle Fehler im System unter Wiederverwendung der bestehenden numerischen Infrastrukturen beseitigen kann. Nach einem vollständigen Zeitschritt, also nach inverser Dynamik und Update von Geschwindigkeiten sowie Positionen, stellen sie das LCP mit Jacobi- und aktualisierter Massenmatrix erneut auf. Einziger Unterschied: Die rechten Seiten der geschwindigkeitsbezogenen Zwangsbedingungen werden mit genau solchen Soll-Relativ-Geschwindigkeiten „befüllt“, welche die bestehenden positions- und orientierungsbezogenen Fehler in einem imaginären Zeitschritt der Länge h vollständig korrigieren würden. Die Lösung des Systems liefert dann einen Vektor mit kartesischen Geschwindigkeiten. Ein Integrationsschritt mit diesen Geschwindigkeiten über die Zeitlänge h hinweg versetzt das Mehrkörpersystem in einen gültigen Zustand.

Erleben [43, 44] verbindet die Ideen der Projektion und der Stoßfortpflanzung miteinander. Grundlegender Gedanke hinter unterschiedlichen Verfahren der Stoßfortpflanzung ist es, sich das Wissen über dominante Körper in einer Mehrkörpersimulation zu Nutze zu machen und somit die Auflösung der Zwangsbedingungen entsprechend zu

vereinfachen. Beispiel: Die Bestimmung der Kontaktkräfte im Modell eines Kistenstapels, der auf massivem Untergrund steht, lässt sich enorm vereinfachen, wenn man annimmt, dass alle Kisten zusammen keinen Einfluss auf die Bewegung des Untergrundes haben werden, auf dem sie stehen. Physikalisch ist das zwar nicht exakt - die Rückwirkungen auf die Flugbahn des Planeten Erde bei Aufschichtung eines Kistenstapels können für die meisten Anwendungen aber vernachlässigt werden. Der Untergrund übt also einen beliebig großen Stoß auf die unterste Kiste aus, um ihren Fall zu stoppen. Die Kisten auf der ersten Ebene „propagieren“ diesen beliebig großen Stoß weiter in die zweite Ebene - eine Kiste der zweiten Ebene des Stapels ist damit nicht in der Lage, eine Kiste der ersten Ebene nach unten zu bewegen usw..

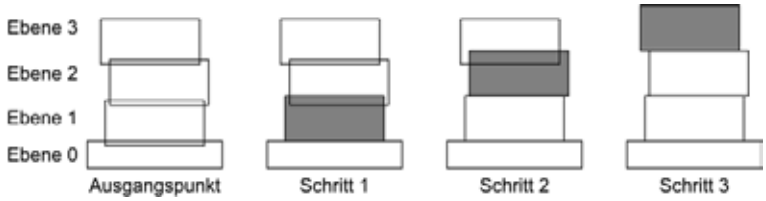


Abbildung 2.17.: Fehlerkorrektur durch Projektion mit Stoßfortpflanzung („*Velocity based shock propagation*“ [44]): Die Positionsfehler in einem Stapel werden ebenenweise von unten nach oben korrigiert. Die Position der jeweils hervorgehobenen Box wurde zuletzt korrigiert.

Nimmt man diese Voraussetzungen als gegeben an, kann man das Verfahren der Projektion ebenenweise „von unten nach oben“ entlang eines Stapels ausführen. Zunächst wird ein LCP für die Kisten der untersten Ebene und den Untergrund formuliert. Die Lösung dieses Systems liefert die notwendigen Verschiebungen der untersten Ebene, so dass keine Durchdringungen mehr mit dem Untergrund auftreten. Nachdem diese Durchdringungen behoben wurden, wird die erste Ebene als Untergrund angenommen und das Verfahren mit der zweiten Ebene fortgesetzt. Auf diesem Wege kann man die Projektion durch Lösen vieler kleiner anstelle eines großen LCP ausführen. Abbildung 2.17 illustriert das Vorgehen. Erleben [43] erreicht mit diesem Verfahren stabile Simulationen sehr großer Stapel trotz eines vergleichsweise simplen LCP-Lösers.

Kapitel 3.

Inverse Dynamik in Mehrkörpersystemen mit der Lagrange-Multiplikatoren-Methode

Die im Rahmen dieser Arbeit entwickelte Dynamik-Simulationskomponente unterstützt zur Lösung des Problems der inversen Dynamik schwerpunktmäßig eine Variante des „kleinen“ Lagrange-Multiplikatoren-Ansatzes. Das Verfahren hat sich in einer Vielzahl praktischer Anwendungen als robust, flexibel und ausreichend präzise erwiesen. Da diese Komponente zentral und die komplexeste des realisierten Mehrkörperdynamiksimulationssystems ist, wird das Verfahren hier ausführlich beschrieben und erläutert.

Das erste Unterkapitel beschreibt zunächst, wie Bewegungsgleichungen von n Starrkörpern zusammen mit den zwischen ihnen geltenden linearen geschwindigkeitsbezogenen Zwangsbedingungen in einem einheitlichen mathematischen System formuliert werden können. Die Unterkapitel 3.2, 3.3 und 3.4 beschreiben detailliert, wie oben genannte Zwangsbedingungen für eine Vielzahl praxisrelevanter, mechanischer Zusammenhänge formuliert werden. Zuletzt folgen in Kapitel 3.5 die detaillierte Beschreibung und Analyse zweier unterschiedlicher numerischer Verfahren zur Lösung des formulierten mathematischen Problems der Mehrkörperdynamik.

3.1. Einheitliche Formulierung von Bewegungsgleichungen und Zwangsbedingungen

Soweit nicht explizit anders definiert sind Ortsvektoren in diesem Kapitel immer in Weltkoordinaten angegeben. Gleiches gilt für die Geschwindigkeitsvektoren: Dabei beschreibt $\vec{v}_i$ die Schwerpunktsgeschwindigkeit des Körpers i in Weltkoordinaten, $\vec{\omega}_i$ seine Winkelgeschwindigkeit in Weltkoordinaten. Jeder Körper besitzt ein eigenes Körperkoordinatensystem. In dieser Arbeit fällt dessen Ursprung immer mit dem Körperschwerpunkt zusammen. Sei $\mathbf{T}_{K_i}^W$ die homogene Transformation (siehe z.B. [23]) des Körperkoordinatensystems von Körper i in Weltkoordinaten, so liegt sein Schwerpunkt in Weltkoordinaten bei $\mathbf{T}_{K_i}^W \cdot \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T$. Die homogene Transformation eines Vektors $\vec{r} = \begin{pmatrix} r_1 & r_2 & r_3 \end{pmatrix} \in \mathbb{R}^3$ wird - obwohl formal wegen der fehlenden Homogenisierung unkorrekt - vereinfacht in der Form $\mathbf{T}_{K_i}^W \vec{r}$ dargestellt. Als Ergebnis wird ein Vektor im $\mathbb{R}^3$ angenommen.

3.1.1. Die Bewegungsgleichungen nach Newton und Euler

Der Impulssatz nach Newton besagt: Die zeitliche Änderung des Impulses $\dot{\vec{p}}$ eines Körpers ist gleich der Summe aller angreifenden Kräfte $\vec{f}$.

$$\dot{\vec{p}} = \vec{f} \tag{3.1}$$

Der Impuls eines Körpers ist definiert als das Produkt aus seiner Masse m und seiner Geschwindigkeit $\vec{v}$, $\vec{p} = m\vec{v}$. Unter Annahme einer konstanten Masse gilt $\dot{\vec{p}} = m\dot{\vec{v}}$ und daher:

$$m\dot{\vec{v}} = \vec{f} \tag{3.2}$$

Mit $\mathbf{M}_m = \begin{pmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{pmatrix}$ kann dies in eine Matrix-Vektor-Form gebracht werden:

$$\begin{aligned} \mathbf{M}_m\dot{\vec{v}} &= \vec{f} \\ \dot{\vec{v}} &= \mathbf{M}_m^{-1} \cdot \vec{f} \end{aligned} \tag{3.3}$$

Der Eulersche Drall- oder Momentsatz stellt analog zum Impulssatz eine Relation zwischen der Änderung des Dralls oder Drehimpulses und dem auf den Körper einwirkenden Drehmoment her. Der Drehimpuls $\vec{L}_S$ in Weltkoordinaten bezogen auf den Schwerpunkt S eines Körpers ist definiert als Produkt aus Trägheitstensor $\mathbf{\Theta_S}$, bezogen auf den Schwerpunkt und seiner Winkelgeschwindigkeit $\vec{\omega}$.

$$\vec{L}_S = \mathbf{\Theta}_S \cdot \vec{\omega} \tag{3.4}$$

Den Trägheitstensor $\mathbf{\Theta}_S(t)$ im Weltkoordinatensystem erhält man durch Transformation des Trägheitstensors im körperfesten Systems ${}_K\mathbf{\Theta}_S$ entsprechend Gleichung A.6 (siehe Anhang A.1).

Der Drallsatz in Bezug auf den Schwerpunkt des Starrkörpers besagt: Die Summe der angreifenden Drehmomente $\vec{\tau}$ ist gleich der Änderung des Drehimpulses $\dot{\vec{L}}_S$:

$$\begin{aligned} \vec{\tau} &= \frac{\mathrm{d}}{\mathrm{d}t}\left(\vec{L}_S\right) \\ &= \mathbf{\Theta}_S\dot{\vec{\omega}} + \vec{\omega} \times \vec{L}_S \\ &= \mathbf{\Theta}_S\dot{\vec{\omega}} + \vec{\omega} \times (\mathbf{\Theta}_S\vec{\omega}) \end{aligned} \tag{3.5}$$

Der Zusammenhang $\frac{\mathrm{d}}{\mathrm{d}t}\left(\vec{L}_S\right) = \mathbf{\Theta}_S\dot{\vec{\omega}} + \vec{\omega} \times \vec{L}_S$ ergibt sich aus dem allgemeinen Zusammenhang zwischen der zeitlichen Ableitung eines Vektors im bewegten System (hier: $\mathbf{\Theta}_S\dot{\vec{\omega}}$) und seiner zeitlichen Ableitung im Weltkoordinatensystem. Für eine detailliertere Herleitung dieses Zusammenhangs siehe z.B. [56].

Umgestellt nach der Winkelbeschleunigung $\dot{\vec{\omega}}$ ergibt sich:

$$\dot{\vec{\omega}} = \mathbf{\Theta}_S^{-1} \cdot (\tau - \vec{\omega} \times (\mathbf{\Theta}_S\vec{\omega})) \tag{3.6}$$

Anmerkung: An Gleichung 3.6 wird deutlich, dass die Winkelbeschleunigung ungleich Null sein kann, selbst wenn die Summe der angreifenden Drehmomente Null ist. Dies ist der Fall, wenn der Starrkörper um eine Achse rotiert, die keine seiner Symmetrieachsen ist (vgl. Anhang A.1).

Auf dem Weg zu einer einheitlichen Formulierung der dynamischen Zusammenhänge im Mehrkörpersystem ist die Zusammenfassung der Bewegungsgleichungen 3.3 und

3.6 der erste Schritt:

$$\begin{pmatrix} \dot{\vec{v}}_i \\ \dot{\vec{\omega}}_i \end{pmatrix} = \begin{pmatrix} \mathbf{M}_{m,i}^{-1} \vec{f}_i \\ \mathbf{\Theta}_i^{-1} \left(\vec{\tau}_i - (\vec{\omega}_i \times (\mathbf{\Theta}_i \vec{\omega}_i)) \right) \end{pmatrix} \tag{3.7}$$

Als nächstes werden die Matrizen $\mathbf{M}_{m,i}^{-1}$ und inverser Trägheitstensor $\mathbf{\Theta}_i^{-1}$ zu einer $6x6$-Matrix zusammengefasst:

$$\begin{pmatrix} \dot{\vec{v}}_i \\ \dot{\vec{\omega}}_i \end{pmatrix} = \begin{pmatrix} \mathbf{M}_{m,i}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Theta}_i^{-1} \end{pmatrix} \begin{pmatrix} \vec{f}_i \\ \vec{\tau}_i - (\vec{\omega}_i \times (\mathbf{\Theta}_i \vec{\omega}_i)) \end{pmatrix} \tag{3.8}$$

Zuletzt werden die Bewegungsgleichungen aller n Körper des Mehrkörpersystems zusammen in ein Matrix-Vektor-System gefasst:

$$\underbrace{\begin{pmatrix} \dot{\vec{v}}_1 \\ \dot{\vec{\omega}}_1 \\ \vdots \\ \dot{\vec{v}}_n \\ \dot{\vec{\omega}}_n \end{pmatrix}}_{\dot{\vec{v}}} = \underbrace{\begin{pmatrix} \mathbf{M}_{m,1}^{-1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Theta}_1^{-1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{M}_{m,n}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{\Theta}_n^{-1} \end{pmatrix}}_{\mathbf{M}^{-1}} \underbrace{\begin{pmatrix} \vec{f}_1 \\ \vec{\tau}_1 - (\vec{\omega}_1 \times (\mathbf{\Theta}_1 \vec{\omega}_1)) \\ \vdots \\ \vec{f}_n \\ \vec{\tau}_n - (\vec{\omega}_n \times (\mathbf{\Theta}_n \vec{\omega}_n)) \end{pmatrix}}_{\vec{f}} \tag{3.9}$$

Anmerkung: Die rotatorischen Anteile des Vektors $\vec{f}$ enthalten nicht nur explizit externe oder eingeprägte Momente wie z.B. ein Motormoment, sondern zusätzlich den sogenannten „gyroskopischen Term“ des Drallsatzes $(\vec{\omega}_n \times (\mathbf{\Theta}_n \vec{\omega}_n))$. Die einheitliche Matrix-Vektor-Formulierung aller Bewegungsgleichungen erlaubt es, Gleichung 3.9 im Folgenden in der kompakten Form

$$\dot{\vec{v}} = \mathbf{M}^{-1} \cdot \vec{f} \tag{3.10}$$

darzustellen.

Sei $\vec{x}$ der Vektor der Positionen und Orientierungen des Mehrkörpersystems, dann gilt: $\dot{\vec{x}} = \vec{v}$. Damit beschreibt Gleichung 3.10 ein Differenzialgleichungssystem der zweiten Ordnung. Auf dem Weg von der analytischen Formulierung hin zur zeitdiskreten Simulation wird daher die zweifache numerische Integration der Bewegungsgleichungen notwendig. Um die Problematik ungenauer oder sogar instabiler Integrationsverfahren zu vermeiden, wird hier eine *impulsbasierte* Herangehensweise gewählt (vgl. auch Ka-

pitel 2.5.5 ab Seite 55): Anstatt über einen Zeitschritt hinweg variable, zeitkontinuierliche Kräfte und Momente auf die Körper des Systems aufzubringen, werden die resultierenden zeitdiskreten Impulse aufgebracht. Der Impuls entspricht dabei dem Integral einer Kraft oder eines Drehmomentes über einen diskreten Zeitschritt hinweg.

Um eine impulsbasierte Formulierung zu erreichen, wird das Differenzial $\dot{\vec{v}}$ links durch einen Differenzenquotienten ersetzt und so umgeformt, dass eine Vorschrift zur Bestimmung der Geschwindigkeiten im nächsten Zeitschritt $t + h$ entsteht.

$$\begin{aligned} \frac{\vec{v}(t+h) - \vec{v}(t)}{h} &= \mathbf{M}^{-1}(t) \cdot \vec{f}(t) \\ \vec{v}(t+h) &= \vec{v}(t) + \mathbf{M}^{-1}(t) \cdot \vec{f}(t) \cdot h \end{aligned} \tag{3.11}$$

Diese Vorschrift entspricht dem expliziten Euler-Schema für die numerische Integration der Geschwindigkeiten, weil der Ausdruck auf der rechten Seite zum Zeitpunkt t ausgewertet wird (vgl. Kapitel 2.6).

3.1.2. Integration bilateraler geschwindigkeitsbezogener Zwangsbedingungen

Wie geschwindigkeitsbezogene Zwangsbedingungen für konkrete Anwendungsprobleme formuliert werden, behandeln die nachfolgenden Unterkapitel ausführlich. Im Allgemeinen lassen sich bilaterale geschwindigkeitsbezogene Zwangsbedingungen in folgender Form darstellen (vgl. auch Kapitel 2.5.1):

$$\mathbf{J}_{e,i}\vec{v} = b_{e,i} \tag{3.12}$$

Hierin ist $\mathbf{J}_{e,i}$ die Jacobimatrix einer einzelnen Zwangsbedingung.

Mehrere solcher Zwangsbedingungen lassen sich in einer Matrix-Vektor-Form formulieren, indem die Koeffizienten aus den m Gleichungen $\mathbf{J}_{e,i}\vec{v} = b_{e,i}, i \in \{1..m\}$ zeilenweise zur Jacobimatrix der Zwangsbedingungen zusammengefasst werden:

$$\begin{aligned} \begin{pmatrix} \mathbf{J}_{e,1} \\ \vdots \\ \mathbf{J}_{e,m} \end{pmatrix} \vec{v} &= \vec{b_e} \\ \mathbf{J}_e \cdot \vec{v} &= \vec{b_e} \end{aligned} \tag{3.13}$$

Um nun die Formulierung der Bewegungsgleichungen (Gleichung 3.11) und der Zwangsbedingungen (Gleichung 3.13) zusammenfassen zu können, sind zwei Schritte notwendig:

1. Die Zwangskräfte und -momente sind Teil der Summe aller Kräfte und Momente in Impuls- und Drallsatz. Die bereits diskretisierten Bewegungsgleichungen (Gleichung 3.11) müssen daher um ihren Einfluss in Form der aus ihnen resultierenden Zwangsimpulse $h\vec{f}_Z$ ergänzt werden. Entsprechend dem d'Alembert-Prinzip (vgl. Kapitel 2.5.1) gilt sie:

   $$h\vec{f}_Z = \mathbf{J}^T\vec{\lambda} \cdot h \tag{3.14}$$

   Zusammenführen von 3.11 und 3.14 führt auf:

   $$\vec{v}(t+h) = \vec{v}(t) + \mathbf{M}^{-1}(t) \cdot \left(\vec{f}(t) \cdot h + \cdot\mathbf{J}_e^T\vec{\lambda}_e \cdot h\right) \tag{3.15}$$

2. Damit die geschwindigkeitsbezogenen Zwangsbedingungen für die neuen Geschwindigkeiten $\vec{v}(t+h)$ erfüllt sind, werden die Zwangsbedingungen (Gleichung 3.13) auf die Vorschrift für die Geschwindigkeiten im nächsten Zeitschritt (Gleichung 3.15) angewandt:

   $$\begin{aligned} \mathbf{J}_e \cdot \vec{v}(t+h) &= \vec{b}_e \\ \Rightarrow \mathbf{J}_e \cdot \left(\vec{v}(t) + \mathbf{M}^{-1}(t) \cdot \vec{f}(t) \cdot h + \mathbf{M}^{-1}(t) \cdot \mathbf{J}_e^T \cdot \vec{\lambda}_e \cdot h\right) &= \vec{b}_e \\ \Rightarrow \mathbf{J}_e\mathbf{M}^{-1}(t)\mathbf{J}_e^T\vec{\lambda}_e \cdot h + \mathbf{J}\left(\vec{v}(t) + \mathbf{M}^{-1}(t) \cdot \vec{f}(t) \cdot h\right) &= \vec{b}_e \end{aligned} \tag{3.16}$$

Bei Gleichung 3.16 handelt es sich um ein lineares Gleichungssystem (LGS) ohne Nebenbedingungen, das nach den Unbekannten $\vec{\lambda}_e h$ gelöst wird. Hieran wird deutlich, dass mit $\vec{\lambda}_e h$ Beträge von Zwangs*impulsen* und nicht von Zwangs*kräften* bestimmt werden, die zur Einhaltung der Zwangsbedingungen in $t+h$ über den Zeitschritt von t nach $t+h$ hinweg aufgebracht werden müssen. Damit sind alle Teile der rechten Seite in Gleichung 3.15 bekannt und die Geschwindigkeiten für den nächsten Zeitschritt können bestimmt werden.

3.1.3. Integration unilateraler Zwangsbedingungen - reibungsfreie Kontakte

Ein Kontaktpunkt zwischen Starrkörpern impliziert zwei physikalische Effekte: Die Kontaktnormalenkraft steht normal auf der Kontaktfläche und verhindert die Durchdringung der Körper. Im Gegensatz dazu wirken Reibungskräfte entlang der Kontaktfläche und dämpfen die relative Geschwindigkeit der Körper entlang der Kontaktfläche. Im Folgenden werden zunächst die Kontaktnormalenzwangsbedingungen behandelt, im nächsten Unterkapitel die Reibungszwangsbedingungen.

Anders als die bisher behandelten bilateralen Zwangsbedingungen, die z.B. aus klassischen Gelenken resultieren, sind Kontaktnormalenzwangsbedingungen *unilateral*. Während in einem Gelenk gefordert wird, dass die relative Geschwindigkeit *gleich* Null ist, wird im Kontaktpunkt gefordert, dass sie *größer oder gleich* Null ist: Driften die Körper in einem Kontaktpunkt bereits auseinander, ist die Bedingung des Kontaktes erfüllt. Für genau diesen Fall ist in Kontakten außerdem eine zweite Bedingung notwendig: Der Impuls entlang der Kontaktnormalen muss, wie die relative Geschwindigkeit, *größer oder gleich* Null sein, wenn man annimmt, dass sie derart orientiert ist, dass ein negativer Betrag das Zusammenhalten der beiden Körper durch den Kontakt bedeuten würde. Ein Kontakt darf also *drücken* aber nicht *ziehen*. Außerdem sind diese beiden Bedingungen zueinander komplementär, was sich in folgender Fallunterscheidung äußert:

1. Die beiden Körper driften auseinander: In diesem Fall ist die relative Geschwindigkeit entlang der Kontaktnormalen größer Null ($v_{\mathrm{rel}} > 0$), der Normalenimpuls jedoch gleich Null ($\lambda_n h = 0$).

2. Der Kontakt verhindert die Durchdringung der beiden Körper: In diesem Fall ist die relative Geschwindigkeit gleich Null ($v_{\mathrm{rel}} = 0$), der Normalenimpuls entsprechend größer als Null ($\lambda_n h > 0$).

Diese beiden Fälle lassen sich mathematisch fassen, indem man sowohl für die relative Geschwindigkeit in Normalenrichtung v_{rel} als auch für den Betrag des Kontaktnormalenimpulses $\lambda_n h$ fordert, dass sie größer oder gleich Null sind und dass zusätzlich ihr Produkt gleich Null ist:

$$v_{\mathrm{rel}} \geq 0,\ \lambda \geq 0, v_{\mathrm{rel}} \cdot \lambda = 0 \tag{3.17}$$

Fasst man nun die Kontaktnormalenzwangsbedingungen zusammen in der Jacobimatrix der Kontaktnormalenzwangsbedingungen $\mathbf{J}_n$, so erhält man:

$$\mathbf{J}_n \cdot \vec{v} - \vec{b}_n \geq \vec{0} \tag{3.18}$$

Hinweis: In einer reinen Nicht-Durchdringungsbedingung gilt für die rechte Seite $b = 0$. Für andere Anwendungsfälle von unilateralen Zwangsbedingungen oder durch die Fehlerkorrektur (vgl. Kapitel 3.1.5) kann b jedoch auch Werte ungleich Null annehmen.

Die Integration dieser Zwangsbedingungen mit den Bewegungsgleichungen geschieht in Anlehnung an Gleichung 3.15 durch Anwendung auf die Vorschrift für die Geschwindigkeiten im nächsten Zeitschritt. Hier müssen zusätzlich die Komplementaritätsbedingungen berücksichtigt werden:

$$\begin{aligned} &\mathbf{J}_n\mathbf{M}^{-1} \cdot \mathbf{J}_n^T \cdot h\vec{\lambda}_n + \mathbf{J}_n \left(\vec{v}(t) + \mathbf{M}^{-1} \cdot h \cdot \vec{f}\right) - \vec{b}_n = \vec{a}_n \\ &\qquad \text{mit den komplementären Bedingungen} \\ &\qquad\qquad \vec{a}_n \geq \vec{0},\ \vec{\lambda}_n \geq \vec{0},\ \vec{a}_n^T\vec{\lambda}_n = 0 \end{aligned} \tag{3.19}$$

Anmerkung: Die Komplementarität der Vektoren $\vec{\lambda}_n$ und $\vec{a}_n$ gilt elementweise. Die Anwendung des Skalarprodukts in $\vec{a}_n^T\vec{\lambda}_n = 0$ ist nur deshalb gültig, weil für jedes einzelne Element die Nichtnegativitätsbedingung gilt. Durch die komplementären Nebenbedingungen erhält man in Gleichung 3.19 ein Lineares Komplementaritätsproblem (engl.: *Linear Complementary Problem, LCP*) [38, 39].

3.1.4. Integration unilateraler Zwangsbedingungen - Berücksichtigung Coulombscher Kontaktreibung

In Kapitel 2.5.5 ab Seite 57 wurde bereits die in der Literatur vielfach behandelte Variante zur Integration von Coulombscher Kontaktreibung sowohl im „großen“ wie auch im „kleinen“ Lagrange-Multiplikatoren-Ansatz beschrieben. Hierin werden zunächst Zwangsbedingungen formuliert, die in ihrer Summe die tangentiale Relativgeschwindigkeit in einem Kontakt beschränken. Zusätzliche „Hilfszwangsbedingungen“ sorgen dafür, dass die dafür aufgebrachten Reibungsimpulse das Coulombsche Gesetz erfüllen.

Wichtige Eigenschaft dieses Ansatzes ist es, dass er in einem linearen Komplemen-

taritätsproblem entsprechend seiner klassischen Definition [38] mündet. Die Formulierung besitzt außerdem Vorteil, dass das komplexe Problem der Kontaktreibung durch eine bekannte mathematische Formulierung approximiert werden kann.

Nachteilig wirkt sich aus, dass die Systemmatrix mit einer steigenden Anzahl an Kontakten sehr schnell sehr groß wird: Bei einer Approximation mit vier Basisvektoren (Approximation des Reibungskonus durch eine Pyramide mit quadratischer Grundfläche) erfordert jeder Kontakt insgesamt bereits 6 Zwangsbedingungen (1x Kontaktnormalen-, 4x Reibungs- und 1x Hilfszwangsbedingung), bei 6 Basisvektoren 8 Zwangsbedingungen usw.. Größere Systeme können auch mit ausgeklügelten Algorithmen nicht mehr echtzeitfähig gelöst werden. Aus diesem Grund und weil sich hieraus eine äußerst flexible Formulierung für ein breites Anwendungsspektrum ergibt, wird in dieser Arbeit eine alternative Formulierung gewählt. Sie führt zu einer deutlich kleineren Systemmatrix und damit zu günstigerem Laufzeitverhalten. Erkauft wird dies durch einen größeren Fehler in der Approximation des Coulombschen Modells. Da im Rahmen der geplanten Anwendungen des Systems die besonders exakte Simulation von Coulombscher Reibung in Starrkörperkontakten jedoch nur eine untergeordnete Rolle spielt, soll diese Tatsache vernachlässigt werden. Für realitätsnahe Kontaktsituationen, z.B. in der Anwendung „Virtuelles Testbed für Laufroboter“ aus Kapitel 6.4, sollen ohnehin externe Kontaktmodelle in die Mehrkörpersimulation integriert werden, da das Modell des Starrkörperkontakts nur für wenige Anwendungen von Kontaktsimulation eine geeignete Approximation der Realität darstellt. Für solche Simulationen, in denen die Kontaktsimulation eher Beiwerk der eigentlichen Anwendung ist, reicht die hier eingesetzte Approximation des Coulombschen Modells völlig aus.

Anstatt stets paarweise unilaterale Zwangsbedingungen in $2n$ jeweils entgegengesetzten Richtungen zu formulieren, werden bilaterale Zwangsbedingungen in n Richtungen formuliert. Abbildung 3.1 verdeutlicht die Idee. Entsprechend müssen die Lagrange-Multiplikatoren anders eingeschränkt werden als in der klassischen Komplementaritätsformulierung:

- Anstelle der Nichtnegativitätsbedingung der Lagrange-Multiplikatoren, die die Größe des Reibungsimpulses angeben, tritt eine Einschränkung durch einen oberen und einen unteren Grenzwert $\lambda_{\mathrm{high},i}, \lambda_{\mathrm{low},i}$, die beide unabhängig voneinander gewählt werden können.

- Anstelle der Elemente des Vektors $\vec{a}$ mit den für sie geltenden Nichtnegativitätsbedingungen und der Komplementarität von $\vec{a}$ und $\vec{\lambda}$ tritt die Schlupfvariable $\vec{w}$ mit

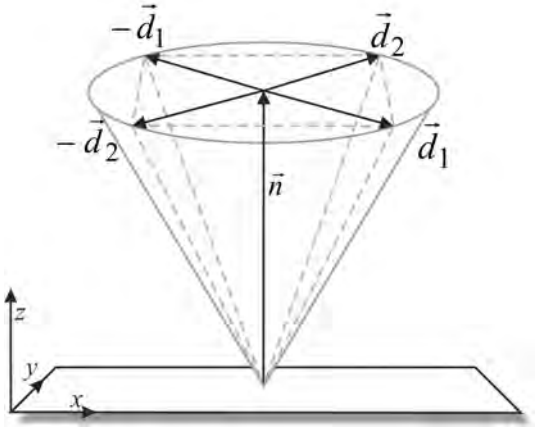


Abbildung 3.1.: Der diskretisierte Reibungskonus zur Formulierung linearer Reibungszwangsbedingungen in der bilateralen Ausprägung

einem Gültigkeitsbereich in Abhängigkeit vom Lagrange-Multiplikator.

Weiterhin wird auf die Formulierung der Hilfs-Zwangsbedingungen aus Gleichung 2.48, Seite 63 zur expliziten Beschränkung der maximalen Reibungskräfte gänzlich verzichtet. Die Einhaltung dieser Bedingungen muss stattdessen während der numerischen Lösung des Problems sichergestellt werden. Details hierzu werden in Kapitel 3.5 präsentiert.

Führt man die Jacobimatrizen der Kontaktnormalen- und der Kontaktreibungszwangsbedingungen zeilenweise zusammen, wendet sie nach bekanntem Muster auf die Geschwindigkeiten für den nächsten Zeitschritt an und berücksichtigt vorgenannte Bedingungen, erhält man die folgende Formulierung:

$$\begin{pmatrix} \mathbf{J_n}\mathbf{M}^{-1}\mathbf{J_n}^T & \mathbf{J_n}\mathbf{M}^{-1}\mathbf{J_f}^T \\ \mathbf{J_f}\mathbf{M}^{-1}\mathbf{J_n}^T & \mathbf{J_f}\mathbf{M}^{-1}\mathbf{J_f}^T \end{pmatrix} \begin{pmatrix} h\vec{\lambda}_n \\ h\vec{\lambda}_f \end{pmatrix} + \begin{pmatrix} \mathbf{J_n} \\ \mathbf{J_f} \end{pmatrix} \left[\vec{v} + \mathbf{M}^{-1}\vec{f}h \right] = \begin{pmatrix} \vec{b_n} \\ \vec{b_f} \end{pmatrix} + \begin{pmatrix} \vec{w_n} \\ \vec{w_f} \end{pmatrix} \tag{3.20}$$

Gesucht werden Tupel (λ_i, w_i), für die gilt:

$$(\lambda_i, w_i) : \begin{cases} \lambda_i = \lambda_{\text{low},i}, & \rightarrow w_i > 0 \\ \lambda_i = \lambda_{\text{high},i}, & \rightarrow w_i < 0 \\ \lambda_{\text{low},i} < \lambda_i < \lambda_{\text{high},i}, & \rightarrow w_i = 0 \end{cases}$$

Der offensichtliche Vorteil gegenüber der klassischen Variante zur Berücksichtigung

von Kontaktreibungskräften ist die deutlich kleinere Systemmatrix $\mathbf{A}$. Bei einer Approximation des Reibungskonus durch eine Pyramide mit quadratischer Grundfläche werden in der klassischen Variante insgesamt sechs Zwangsbedingungen formuliert, in dieser Formulierung sind es derer nur drei (1x Kontaktnormale, 2x Reibung). Doch ergeben sich auch Nachteile:

- Diese Formulierung ist kein LCP im Sinne seiner klassischen Definition [38] mehr. Bestehende Theorie und Lösungsverfahren hierzu können daher nicht oder zumindest nicht ohne Anpassungen angewandt werden.

- Die Grenzwerte für die Lagrange-Multiplikatoren der Reibungszwangsbedingungen sind abhängig von den Lagrange-Multiplikatoren der Kontaktnormalenzwangsbedingungen und können daher nicht im Vorhinein festgelegt werden. Dieser Zusammenhang wird hier nicht berücksichtigt und muss entsprechend durch einen Lösungsalgorithmus hergestellt werden.

Neben einem günstigeren Laufzeitverhalten bedingt durch die kleinere Systemmatrix hat diese Formulierung noch einen erheblichen Vorteil: Die flexible Wahl der Grenzwerte der Lagrange-Multiplikatoren ($\lambda_{\mathrm{low},i}, \lambda_{\mathrm{high},i}$) erlaubt die Formulierung von Zwangsbedingungen für eine Vielzahl praktischer Anwendungen. So lassen sich z.B. Motoren mit begrenztem Drehmoment in Form geschwindigkeitsbezogener Zwangsbedingungen modellieren. Im Gegensatz zur Modellierung eines Motors durch Aufbringen eines externen Drehmoments (als Teil der Vektoren $\vec{f}$ bzw. $\vec{\tau}$ in den Bewegungsgleichungen 3.9), erlaubt dieses Verfahren erheblich größere Zeitschrittweiten und eignet sich daher besonders für realzeitorientierte, interaktive Anwendungen.

Die gleichzeitige Berücksichtigung von uni- und bilateralen Zwangsbedingungen ist ebenfalls möglich. Tatsächlich entscheidet lediglich die Wahl der Grenzwerte der Lagrange-Multiplikatoren darüber, ob eine Zwangsbedingung uni- oder bilateral ist. Daher wird bei der mathematischen Formulierung auch nicht länger zwischen uni- und bilateralen Zwangsbedingungen unterschieden, wie es in den klassischen Varianten üblich ist. Sie werden jetzt in der Jacobimatrix der Nicht-Reibungszwangsbedingungen $\mathbf{J}_{nf}$ zusammengefasst. Lediglich die Reibungszwangsbedingungen $\mathbf{J}_f$ müssen explizit deklariert sein, da die Grenzwerte ihrer Lagrange-Multiplikatoren von den Lagrange-Multiplikatoren der zugehörigen Kontaktnormalenzwangsbedingungen abhängig sind.

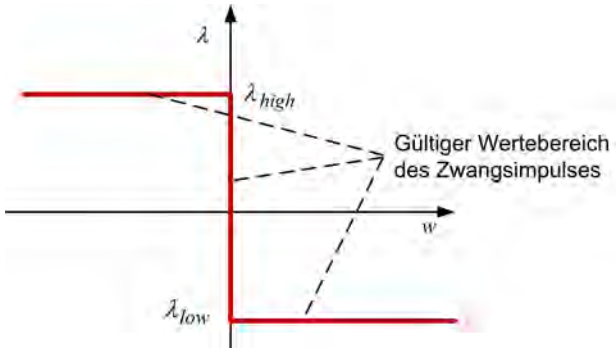


Abbildung 3.2.: Darstellung des Zusammenhangs von Schlupfvariable w und gültigem Wertebereich des zugehörigen Zwangsimpulses λ einer Zwangsbedingung.

Die resultierende vereinfachte Formulierung hat die Form:

$$\begin{pmatrix} \mathbf{J}_{nf}\mathbf{M}^{-1}\mathbf{J}_{nf}^T & \mathbf{J}_{nf}\mathbf{M}^{-1}\mathbf{J}_f^T \\ \mathbf{J}_f\mathbf{M}^{-1}\mathbf{J}_{nf}^T & \mathbf{J}_f\mathbf{M}^{-1}\mathbf{J}_f^T \end{pmatrix} \begin{pmatrix} h\vec{\lambda}_{nf} \\ h\vec{\lambda}_f \end{pmatrix} + \begin{pmatrix} \mathbf{J}_{nf} \\ \mathbf{J}_f \end{pmatrix} \left[\vec{v} + \mathbf{M}^{-1}\vec{f}h\right] = \begin{pmatrix} \vec{b}_{nf} \\ \vec{b}_f \end{pmatrix} + \begin{pmatrix} \vec{w}_{nf} \\ \vec{w}_f \end{pmatrix} \tag{3.21}$$

Gesucht werden Tupel (λ_i, w_i), für die gilt:

$$(\lambda_i, w_i) : \begin{cases} \lambda_i = \lambda_{\text{low},i}, & \leftrightarrow w_i > 0 \\ \lambda_i = \lambda_{\text{high},i}, & \leftrightarrow w_i < 0 \\ \lambda_{\text{low},i} < \lambda_i < \lambda_{\text{high},i}, & \leftrightarrow w_i = 0 \end{cases}$$

Die Werte der Schlupfvariablen w_i repräsentieren die Abweichungen der relativen Geschwindigkeit im Sinn einer Zwangsbedingung vom entsprechenden Sollwert. Hat der Lagrange-Multiplikator einer Zwangsbedingung einen seiner Grenzwerte erreicht, darf die Abweichung vom Sollwert größer oder kleiner Null werden. Dieser Zusammenhang ist analog zur Komplementarität in der klassischen Variante.

Am Beispiel Kontaktnormale heißt das: Der Sollwert der Relativgeschwindigkeit entlang der Kontaktnormalen ist gleich Null. Der untere Grenzwert λ_{low} einer Kontaktnormalenzwangsbedingung ist ebenfalls gleich Null, der obere unendlich. Ein Kontakt darf beliebig stark „drücken“, jedoch nicht „ziehen“. Driften zwei Körper im Kontakt i auseinander, d.h. $w_i > 0$, nimmt der Kontaktnormalenimpuls den Wert Null an: Der Kontakt hat keinen Einfluss. Verhindert der Kontakt die Durchdringung, bewegen sich die Körper im Kontaktpunkt nicht aufeinander zu ($w_i = 0$) und der Kontaktnormalenimpuls wird echt

größer als Null ($\lambda_i > 0$).

Am Beispiel einer Motorzwangsbedingung bedeutet das: Ein zu modellierender Motor habe ein maximales Drehmoment von τ_m und ein minimales Drehmoment von 0. Der obere Grenzwert des Lagrange-Multiplikators ist entsprechend: $\lambda_{\text{high}} = h \cdot \tau_m$, der untere entsprechend $\lambda_{\text{low}} = h \cdot 0 = 0$. Die Grenzwerte entsprechen dem maximalen bzw. minimalen Drehimpuls, den der Motor über einen Zeitschritt hinweg aufbringen kann unter der Annahme, dass er sein maximales und minimales Drehmoment über einen ganzen Zeitschritt hinweg aufrechterhalten kann. Eine geschwindigkeitsbezogene Zwangsbedingung fordert für die relative Winkelgeschwindigkeit zwischen den über den Motor verbundenen Körpern einen Wert $b_{i,\text{Soll}}$ von z.B. 3 [rad/s]. Führt die gegenwärtige Konfiguration des Mehrkörpersystems dazu, dass die geforderte Sollgeschwindigkeit aufgrund eines zu großen Lastmoments nicht erreicht werden kann und somit unterschritten wird ($\Rightarrow w_i < 0$), so erbringt der Motor im selben Zeitschritt den maximal möglichen Drehimpuls, d.h. er versucht, die Relativgeschwindigkeit weiter zu beschleunigen, um die Zwangsbedingung zu erfüllen. Abbildung 3.2 verdeutlicht den Zusammenhang grafisch.

Wird die Sollgeschwindigkeit hingegen sogar überschritten ($\Rightarrow w_i > 0$), z.B. bei einer Bergabfahrt eines PKW, wird der Motor im selben Zeitschritt den minimalen Drehimpuls, also $\lambda_{\text{low}} = 0$ aufbringen. Der Motor im Beispiel kann damit nicht verzögernd, sondern ausschließlich beschleunigend wirken.

Beide Beispiele geben bereits einen guten Eindruck davon, wie vielseitig praxisrelevante Zwangsbedingungen in dieser Variante formuliert werden können. Die Kapitel 3.2, 3.3 und 3.4 beschreiben detailliert, wie geschwindigkeitsbezogene Zwangsbedingungen für eine Vielzahl praxisrelevanter Anwendungen für die vereinfachte Formulierung aus Gleichung 3.21 aufzustellen sind.

3.1.5. Stabilisierung von Zwangsbedingungen

Zwangsbedingungen, die durch klassische Gelenke aber auch durch Kontakte impliziert werden, sind naturgemäß positions- bzw. orientierungsbezogen. Das bedeutet, sie stellen Bedingungen für die Relationen zwischen den Positionen und Orientierungen der beteiligten Körper auf. Die impulsbasierte Formulierung mit Lagrange-Multiplikatoren kann jedoch nur geschwindigkeitsbezogene Zwangsbedingungen berücksichtigen. Zwar ist es möglich, für jede prinzipiell positionsbezogene Zwangsbedingung eine entsprechende geschwindigkeitsbezogene zu finden, die das gleiche Ver-

halten hervorruft. Treten jedoch kleine Fehler bei der Einhaltung geschwindigkeitsbezogener Zwangsbedingungen auf, wächst mit fortschreitender Zeit der Fehler in der entsprechenden positionsbezogenen Zwangsbedingung unbegrenzt an. Am Beispiel eines Gelenks erkennt man, dass die verbundenen Körper mit der Zeit immer weiter auseinanderdriften, obwohl die geschwindigkeitsbezogene Zwangsbedingung in jedem Schritt nur minimal verletzt wird.

Um dies auszugleichen, ist eine Fehlerkorrektur notwendig. Ein besonders einfaches und gleichzeitig effizientes Verfahren ist die Stabilisierung von Zwangsbedingungen oder *Constraintstabilisierung* nach Baumgarte [16]. Die Idee: Die relative Sollgeschwindigkeit der Zwangsbedingung (ihre rechte Seite) wird um einen additiven Wert ergänzt, der den bestehenden positionsbezogenen Fehler über die nächsten Simulationsschritte hinweg ausgleicht. Ihr Betrag wird proportional zur Größe des Fehlers angesetzt. Als Proportionalitätskonstante k_{er} wird typischerweise ein Wert zwischen 0,1 und 1,0 gewählt. Ein Wert von 1,0 bedeutet, die korrigierende Relativgeschwindigkeit ist genau so groß, dass der bestehende Fehler in einem einzigen Zeitschritt vollständig ausgeglichen wird. Sei also e der positionsbezogene Fehler einer Zwangsbedingung und h die Zeitschrittweite, dann hat der Fehlerkorrekturterm b_{error}, der die Dimension einer Relativgeschwindigkeit hat, die Form:

$$b_{\mathrm{error}} = k_{\mathrm{er}} \cdot \frac{e}{h} \tag{3.22}$$

Die Werte für die Stabilisierung werden einfach auf die Sollwerte $\vec{b}$ der geschwindigkeitsbezogenen Zwangsbedingung aufaddiert:

$$\mathbf{J}\vec{v} = \vec{b} + \vec{b}_{\mathrm{error}} \tag{3.23}$$

Die Wahl eines geeigneten Wertes für k_{er} wird gemeinhin als besondere Schwäche dieses Verfahrens angeführt. Richtig ist: Ein zu großer Wert führt zu Überschwingen, der Fehler läuft nicht gegen Null, sondern pendelt zwischen Werten größer und kleiner als Null. Gelenke scheinen hin und her zu federn. Bei einem zu kleinen Wert hingegen wird der Fehler nur sehr langsam und in zu geringem Maße korrigiert, Gelenke sehen „wabbelig“ aus. Die Erfahrung mit vielen unterschiedlichen Anwendungen hat jedoch gezeigt, dass nur zwei Richtgrößen für fast alle kombinierten Anwendungen mit Kontakten und Gelenken sehr gute Ergebnisse liefern: $k_{\mathrm{er}} = 0,15$ für Kontaktnormalenzwangsbedingungen und $k_{\mathrm{er}} = 0,5$ für alle anderen.

3.2. Geschwindigkeitsbezogene Zwangsbedingungen für klassische Gelenke

Bei der Formulierung von Zwangsbedingungen gibt es große Überschneidungen, z.B. zwischen den unterschiedlichen Gelenktypen. Daher werden zunächst einige gemeinsame Notationen für die nachfolgenden Abschnitte festgelegt, mit deren Hilfe die einzelnen Zwangsbedingungen kompakt und übersichtlich formuliert werden können.

3.2.1. Bestimmung von Position und Orientierung von Gelenken zur Simulationslaufzeit

Bei den klassischen Gelenktypen ist es häufig erforderlich, dem Gelenk eine eigene Position und Orientierung zuzuordnen. Erst hierdurch wird es möglich, die relativen Bewegungen der verbundenen Körper im Sinne der Ausrichtung des Gelenks einzuschränken. Man spricht vom Gelenkkoordinatensystem. Initial, also zu Beginn der Simulation, ist es durch die Modellierung bestimmt, es ergibt sich aus den Achsen des Gelenks. Sei dieses initiale Koordinatensystem gegeben durch die homogene Transformation $\mathbf{T}_g^W \in \mathbb{R}^{4\times4}$ in Weltkoordinaten. Um die Lage der Gelenkachsen zu einem beliebigen Zeitpunkt der Simulation bestimmen zu können, werden zunächst die Lagen des initialen Gelenkkoordinatensystems in den initialen Körperkoordinatensystemen $\mathbf{T}_{K1}^W, \mathbf{T}_{K2}^W$ bestimmt:

$$\begin{aligned}\mathbf{T}_g^{K1} &= {\mathbf{T}_{K1}^W}^{-1} \cdot \mathbf{T}_g^W \\ \mathbf{T}_g^{K2} &= {\mathbf{T}_{K2}^W}^{-1} \cdot \mathbf{T}_g^W\end{aligned} \tag{3.24}$$

Die Transformationen $\mathbf{T}_g^{K1}$ und $\mathbf{T}_g^{K2}$ sind konstant während der gesamten Simulationslaufzeit. Seien $\mathbf{T}_{K1}^W(t), \mathbf{T}_{K2}^W(t) \in \mathbb{R}^{4\times4}$ die Koordinatensysteme der verbundenen Körper in Form homogener Transformationen im Verlauf der Simulation, dann ergeben sich die zwei Lagen des Gelenks in Weltkoordinaten zur Laufzeit aus:

$$\begin{aligned}\mathbf{T}_g^{W,K1}(t) &= \mathbf{T}_{K1}^W(t) \cdot \mathbf{T}_g^{K1} \\ \mathbf{T}_g^{W,K2}(t) &= \mathbf{T}_{K2}^W(t) \cdot \mathbf{T}_g^{K2}\end{aligned} \tag{3.25}$$

Die beiden Positionen des Gelenks, d.h. die Ursprünge ihrer Koordinatensysteme,

entsprechen den Ankerpunkten des Gelenks. Hier greifen die Zwangskräfte bzw. -impulse des Gelenks an den verbundenen Körpern an. Die beiden Rotationen des Gelenks, mathematisch gesehen also die oberen linken 3×3-Matrizen von $\mathbf{T}_g^{WK1/2}(t)$: $\mathbf{R}_g^{W,K1/2}(t)$, dienen der Bestimmung der Orientierung und damit der Gelenkachsen zur Laufzeit der Simulation. Wie die jeweiligen Achsen aus diesen beiden Rotationen bestimmt werden, hängt vom einzelnen Gelenktyp ab.

3.2.2. Einheitliche Schreibweise für Zwangsbedingungen

Geschwindigkeitsbezogene Zwangsbedingungen sollen in der Form $\mathbf{J} \cdot \vec{v} = \vec{b}, \mathbf{J} \in \mathbb{R}^{m \times 6n}, \vec{v} \in \mathbb{R}^{6n}, \vec{b} \in \mathbb{R}^m$ zusammengefasst werden. Bestehen zwischen zwei Körpern i, j keine Zwangsbedingungen, so bestehen zwischen ihnen sechs relative Freiheitsgrade. Eine starre (engl.: *rigid*) Verbindung zwischen zwei Körpern bedeutet, dass alle sechs relativen Freiheitsgrade, drei translatorische und drei rotatorische, entfernt werden. Auf der Ebene der Geschwindigkeiten müssen entsprechend lineare und rotatorische Geschwindigkeiten beider Körper identisch bzw. ihre Differenz gleich Null sein:

$$\begin{aligned} \vec{v}_i - \vec{v}_j &= \vec{0} \\ \vec{\omega}_i - \vec{\omega}_j &= \vec{0} \end{aligned} \tag{3.26}$$

Nehmen wir an, das Mehrkörpersystem bestehe nur aus den beiden Körpern i und j, so würden diese Zwangsbedingungen wie folgt in der geforderten Matrix-Vektor-Form formuliert:

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}}_{\mathbf{J}} \cdot \underbrace{\begin{pmatrix} \vec{v}_i \\ \vec{\omega}_i \\ \vec{v}_j \\ \vec{\omega}_j \end{pmatrix}}_{\vec{v}} = \vec{b}, \quad \vec{b} = \vec{0} \tag{3.27}$$

Zur besseren Übersicht werden die Einträge der Jacobimatrix im Folgenden wie folgt auf sogenannte *Teil-Jacobimatrizen* aufgeteilt:

$$\mathbf{J} = \begin{pmatrix} \mathbf{J}_{\mathrm{rigid,tra},i} & \mathbf{J}_{\mathrm{rigid,tra},j} \\ \mathbf{J}_{\mathrm{rigid,rot},i} & \mathbf{J}_{\mathrm{rigid,rot},j} \end{pmatrix} \tag{3.28}$$

Die Teil-Jacobimatrizen $\mathbf{J}_{\mathrm{rigid},tra}$ (translatorisch) und $\mathbf{J}_{\mathrm{rigid},rot}$ (rotatorisch) besitzen je sechs Spalten und drei Zeilen, entsprechend der Anzahl der translatorischen und rotatorischen Freiheitsgrade, die beschränkt werden sollen. Für andere Gelenkverbindungen, die nicht alle sechs Freiheitsgrade beschränken, haben die Teil-Jacobimatrizen entsprechend auch weniger als drei Zeilen.

Die Einteilung in rotatorische und translatorische Zwangsbedingung bezieht sich immer auf das Gelenkkoordinatensystem. Da dieses i.A. nicht mit den Schwerpunkten der Körper zusammenfällt, betreffen die meisten translatorischen Zwangsbedingungen auch die rotatorischen Geschwindigkeiten der betroffenen Körper. Eine Ausnahme stellt außerdem die Schraubverbindung aus Abschnitt 3.4.1 dar, da sie nur einen „kombinierten“ Freiheitsgrad zwischen den Körpern übrig lässt.

Im Beispiel der starren Verbindung besitzen alle vier Teilmatrizen je drei Zeilen. Die gesamte Jacobimatrix wird analog aus solchen Teilmatrizen zusammengesetzt. Bestehe das Mehrkörpersystem aus den drei Körpern i, j und k und seien je i und j sowie i und k starr miteinander verbunden, so hätte die Jacobimatrix entsprechend die Form:

$$\mathbf{J} = \begin{pmatrix} \mathbf{J}_{\mathrm{rigid,tra},i} & \mathbf{J}_{\mathrm{rigid,tra},j} & \mathbf{0} \\ \mathbf{J}_{\mathrm{rigid,rot},i} & \mathbf{J}_{\mathrm{rigid,rot},j} & \mathbf{0} \\ \mathbf{J}_{\mathrm{rigid,tra},i} & \mathbf{0} & \mathbf{J}_{\mathrm{rigid,tra},k} \\ \mathbf{J}_{\mathrm{rigid,rot},i} & \mathbf{0} & \mathbf{J}_{\mathrm{rigid,rot},k} \end{pmatrix} \tag{3.29}$$

Aufbauend auf dieser Art der Beschreibung werden im nächsten Kapitel die Teil-Jacobimatrizen für alle praxisrelevanten Gelenktypen formuliert. Zur vollständigen Formulierung einer Zwangsbedingung im Sinne von Gleichung 3.21 gehören außerdem ein Sollwert für die rechte Seite (b) sowie Grenzwerte für die Lagrange-Multiplikatoren ($\lambda_{\mathrm{low}}, \lambda_{\mathrm{high}}$). In der Mehrheit der Fälle dürfen Zwangsimpulse „unendlich stark“ sein, d.h. für ihre Grenzwerte gilt: $\lambda_{\mathrm{low}} = -\infty, \lambda_{\mathrm{high}} = \infty$. In diesem Fall werden sie nicht explizit angegeben. Für die eingeschränkten Freiheitsgrade an einem Gelenk gilt gerade, dass keine relative Geschwindigkeit auftritt. Soweit nicht anders angegeben, gilt daher für die rechte Seite einer Zwangsbedingung: $b = 0$.

3.2.3. Das Kugelgelenk

Den Anfang macht das Kugelgelenk, denn es stellt die Basis für die Realisierung weiterer Gelenke dar. Abbildung 3.3 zeigt ein 3D-Modell eines Kugelgelenks sowie einen 2D-Schnitt durch das Modell. Das Kugelgelenk (engl.: *Ball-In-Socket*) beschränkt drei der drei möglichen translatorischen Freiheitsgrade der Relativbewegung zweier Körper. Alle rotatorischen Freiheitsgrade bleiben unbeschränkt, entsprechend werden keine rotatorischen Teil-Jacobimatrizen benötigt.

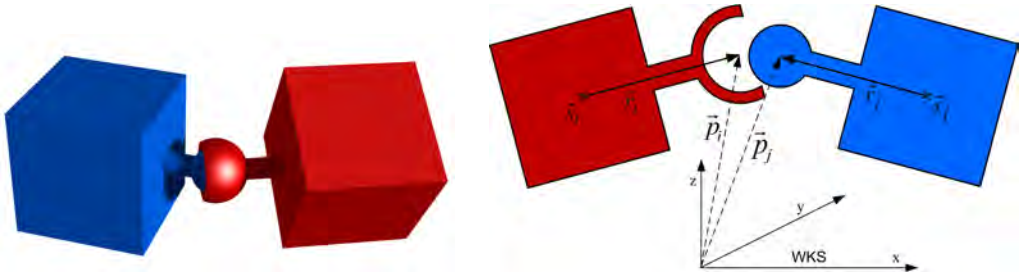


Abbildung 3.3.: Links: 3D-Modell eines Kugelgelenks. Rechts: 2D-Abbildung des Kugelgelenks.

Geht man von einem gültigen Ausgangszustand aus, wird das Kugelgelenk auf der Ebene von Geschwindigkeiten dadurch realisiert, dass die linearen Geschwindigkeiten der Ankerpunkte $\dot{\vec{p}}_i$ und $\dot{\vec{p}}_j$ gleich sind. Abbildung 3.3 verdeutlicht die Zusammenhänge: Seien $\vec{v}_i, \vec{\omega}_i, \vec{v}_j, \vec{\omega}_j$ die Schwerpunkts- und die Winkelgeschwindigkeiten in Weltkoordinaten der Körper i und j und $\vec{r}_i, \vec{r}_j$ die Vektoren von den Schwerpunkten der Körper zum jeweiligen Ankerpunkt, dann lässt sich dies formulieren als:

$$\begin{aligned} \dot{\vec{p}}_i - \dot{\vec{p}}_j &= \vec{b}, \vec{b} = \vec{0} \\ \Rightarrow \vec{v}_i + \vec{\omega}_i \times \vec{r}_i - \vec{v}_j - \vec{\omega}_j \times \vec{r}_j &= \vec{b} \\ \Rightarrow \vec{v}_i - \vec{r}_i \times \vec{\omega}_i - \vec{v}_j + \vec{r}_j \times \vec{\omega}_j &= \vec{b} \end{aligned} \tag{3.30}$$

Auf dem Weg zur einheitlichen Matrix-Vektor-Formulierung in der geforderten Form $\mathbf{J}_{\text{kugel}} \cdot \vec{v} = \vec{b}$ werden noch die Einheitsmatrix $\mathbf{E}$ und anstelle der Kreuzprodukte gleichwertige Kreuzproduktmatrizen (siehe Anhang A.3) eingefügt:

$$\Rightarrow \mathbf{E} \cdot \vec{v}_i - \mathbf{r}_i^\star \cdot \vec{\omega}_i - \mathbf{E} \cdot \vec{v}_j + \mathbf{r}_j^\star \cdot \vec{\omega}_j = \vec{b} \tag{3.31}$$

Die Teil-Jacobimatrizen erhalten dann die Form:

$$\begin{aligned}\mathbf{J}_{\text{kugel,tra},i} &= \begin{pmatrix} 1 & 0 & 0 & 0 & r_{i,3} & -r_{i,2} \\ 0 & 1 & 0 & -r_{i,3} & 0 & r_{i,1} \\ 0 & 0 & 1 & r_{i,2} & -r_{i,1} & 0 \end{pmatrix}, \\ \mathbf{J}_{\text{kugel,tra},j} &= \begin{pmatrix} -1 & 0 & 0 & 0 & -r_{j,3} & r_{j,2} \\ 0 & -1 & 0 & r_{j,3} & 0 & -r_{j,1} \\ 0 & 0 & -1 & -r_{j,2} & r_{j,1} & 0 \end{pmatrix}\end{aligned} \tag{3.32}$$

Stabilisierung

Die Ankerpunkte des Gelenks entsprechen den Ursprüngen der beiden Gelenkkoordinatensysteme $\mathbf{T}_g^{W,K_i}(t)$ und $\mathbf{T}_g^{W,K_j}(t)$. Liegt kein Fehler vor, sind beide Positionen identisch. Ist ein Fehler vorhanden, so entspricht er in der positionsbezogenen Zwangsbedingung der Abweichung zwischen beiden Ankerpunktpositionen :

$$\vec{e} = \vec{p_i} - \vec{p_j} = \mathbf{T}_g^{W,K_i}(t) \cdot \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T - \mathbf{T}_g^{W,K_j}(t) \cdot \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T \tag{3.33}$$

Entsprechend der Idee der Stabilisierung wird die Zwangsbedingung derart angepasst, dass sie eine Relativgeschwindigkeit proportional zum entstandenen Fehler erzwingt, die diesen wieder korrigiert. Die Richtung der Korrekturgeschwindigkeit entspricht der Richtung des Fehlers, ihr Betrag ist proportional dazu:

$$\begin{aligned}\vec{b}_{\text{error,kugel}} &= k_{\text{er}} \cdot \frac{1}{h} \cdot \vec{e} \\ &= k_{\text{er}} \cdot \frac{1}{h} \cdot (\vec{p_i} - \vec{p_j})\end{aligned} \tag{3.34}$$

3.2.4. Das Drehgelenk

Abbildung 3.4 zeigt ein 3D-Modell eines Drehgelenks. Das Drehgelenk entfernt fünf der sechs relativen Freiheitsgrade zwischen zwei Körpern: Wie beim Kugelgelenk werden alle drei translatorischen Freiheitsgrade entfernt, zusätzlich noch zwei der drei rotatorischen. Die Analogie zum Kugelgelenk führt dazu, dass die rotatorischen Teil-Jacobimatrizen mit denen des Kugelgelenks übereinstimmen:

$$\mathbf{J}_{\text{dreh,tra}} = \mathbf{J}_{\text{kugel,tra}} \tag{3.35}$$

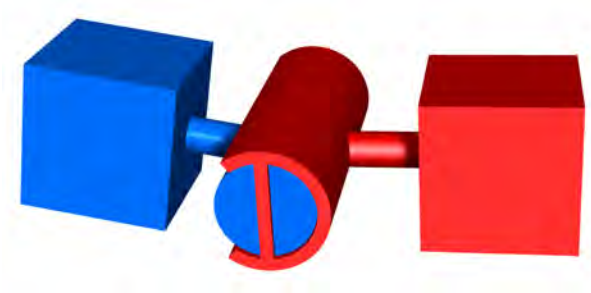


Abbildung 3.4.: 3D-Modell eines rotatorischen oder Dreh-Gelenks

Zusätzlich müssen zwei der drei rotatorischen Freiheitsgrade beschränkt werden. Sei $\mathbf{R}_g^W(t)$ die Rotation des Gelenks zum Zeitpunkt t. Sei weiterhin o.B.d.A. die z-Achse des Gelenkkoordinatensystems seine Drehachse, d.h. alle relativen Bewegungen im Gelenk entlang seiner x- und y-Achse sollen unterbunden werden. Dann müssen zunächst die Koordinatenachsen des Gelenks in Weltkoordinaten $\vec{x},\ \vec{y}$ bestimmt werden:

$$\vec{x} = \mathbf{R}_g^W(t) \cdot \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}^T, \quad \vec{y} = \mathbf{R}_g^W(t) \cdot \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}^T \tag{3.36}$$

Die Drehgeschwindigkeit ist in jedem Punkt eines Starrkörpers gleich. Der Anteil der Winkelgeschwindigkeit $\vec{\omega}_i$ von Körper i in Richtung einer gegebenen Drehachse $\vec{x}, |\vec{x}| = 1$ ergibt sich daher direkt aus der Projektion des Drehgeschwindigkeitsvektors auf den Achsenvektor: $\vec{x} \cdot \vec{\omega}_i$. Die Formulierung der Zwangsbedingung für die rotatorische Geschwindigkeit ist daher recht einfach:

$$\begin{aligned} \vec{x} \cdot \vec{\omega_i} - \vec{x} \cdot \vec{\omega_j} &= 0 \\ \vec{y} \cdot \vec{\omega_i} - \vec{y} \cdot \vec{\omega_j} &= 0 \end{aligned} \tag{3.37}$$

Die rotatorischen Teil-Jacobimatrizen haben folglich die Form:

$$\begin{aligned} \mathbf{J}_{\text{dreh,rot},i} &= \begin{pmatrix} 0 & 0 & 0 & x_1 & x_2 & x_3 \\ 0 & 0 & 0 & y_1 & y_2 & y_3 \end{pmatrix} \\ \mathbf{J}_{\text{dreh,rot},j} &= \begin{pmatrix} 0 & 0 & 0 & -x_1 & -x_2 & -x_3 \\ 0 & 0 & 0 & -y_1 & -y_2 & -y_3 \end{pmatrix} \end{aligned} \tag{3.38}$$

Stabilisierung

Der Stabilisierungsterm setzt sich - analog zur Jacobimatrix - aus den drei Komponenten für die translatorischen Zwangsbedingungen und zwei weiteren für die rotatorischen Zwangsbedingungen zusammen. Die Bestimmung der Fehlerterme für die translatorischen Zwangsbedingungen verläuft wie beim Kugelgelenk.

$$\vec{b}_{\text{error,dreh,trans}} = \vec{b}_{\text{error,kugel}} \tag{3.39}$$

Zur Korrektur des rotatorischen Fehlers kann man die folgende Überlegung anstellen: Sei $\vec{z}_i$ die Drehachse des Gelenks von Körper i aus: $\vec{z}_i = \mathbf{R}_g^{W,K_i} \cdot \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^T$. Sei weiterhin $\vec{z}_j$ analog dazu die Drehachse des Gelenks von Körper j aus. Sind beide Achsen identisch, liegt kein Fehler vor. Angenommen, es liegt eine Verdrehung beider Achsen um den Winkel ϵ vor. Zur Korrektur müssen die Achsen um ϵ Radiant entlang der Achse $\vec{z}_i \times \vec{z}_j$ verdreht werden. Die Richtung einer relativen Korrekturgeschwindigkeit ω_{kor} entspricht dem Vektor $\frac{\vec{z}_i \times \vec{z}_j}{|\vec{z}_i \times \vec{z}_j|}$, ihr Betrag muss proportional zum Fehler ϵ sein. Die Korrektur der Abweichung zwischen den beiden z-Achsen wird durch Verdrehungen entlang der x- und y-Achsen des Gelenks erreicht, die jeweiligen Anteile der Korrekturgeschwindigkeit erhält man durch Projektion auf diese Achsen. Demnach erhält der Korrekturterm die Form:

$$b_{\text{error,dreh,rot}} = \frac{1}{h} \cdot k_{\text{er}} \cdot \epsilon \begin{pmatrix} \vec{x} \cdot \vec{\omega}_{\text{kor}} \\ \vec{y} \cdot \vec{\omega}_{\text{kor}} \end{pmatrix} = \frac{1}{h} \cdot k_{\text{er}} \cdot \frac{\epsilon}{|\vec{z}_i \times \vec{z}_j|} \begin{pmatrix} \vec{x} \cdot (\vec{z}_i \times \vec{z}_j) \\ \vec{y} \cdot (\vec{z}_i \times \vec{z}_j) \end{pmatrix} \tag{3.40}$$

Die Vektoren $\vec{z}_i$ und $\vec{z}_j$ sind Einheitsvektoren, daher gilt:

$$|\vec{z}_i \times \vec{z}_j| = \sin(\epsilon) \tag{3.41}$$

Für kleine ϵ kann außerdem $\sin(\epsilon) \approx \epsilon$ angenommen werden. Somit ergibt sich der rotatorische Stabilisierungsterm des Drehgelenks zu:

$$\vec{b}_{\text{error,dreh,rot}} = \frac{1}{h} k_{\text{er}} \begin{pmatrix} \vec{x} \cdot (\vec{z}_i \times \vec{z}_j) \\ \vec{y} \cdot (\vec{z}_i \times \vec{z}_j) \end{pmatrix} \tag{3.42}$$

3.2.5. Das Kardan- oder Kreuzgelenk

Abbildung 3.5 zeigt ein Modell eines Kardangelenks (engl.: *Universal Joint*). Das Kardangelenk kann eine Drehachse mit Kraftübertragung um einen Winkel knicken. Es besteht im Prinzip aus zwei fest verbundenen orthogonalen Drehachsen, von denen eine Achse konstante Orientierung zu Körper i, die andere konstante Orientierung zu Körper j besitzt. Die relative Winkelgeschwindigkeit zwischen den Körpern wird entlang einer dritten Achse eingeschränkt, die senkrecht auf den beiden Drehachsen steht. Entsprechend seiner Mechanik kann das Kardangelenk durch zwei Drehgelenke und einen dritten „Hilfskörper“ aufgebaut werden. Sind die internen Vorgänge im Kardan von untergeordneter Bedeutung, ist eine Realisierung durch zwei Drehgelenke jedoch ineffizient: Das Kardangelenk als Ganzes entfernt nur vier Freiheitsgrade zwischen den verbundenen (in der Abbildung 3.5 blau dargestellten) Körpern und impliziert entsprechend vier Zeilen in der Jacobimatrix. Ein Drehgelenk hingegen entfernt fünf Freiheitsgrade, zwei entsprechend zehn. Durch ein explizites Kardangelenk kann die Anzahl der zu berücksichtigenden Zwangsbedingungen also klein gehalten werden.

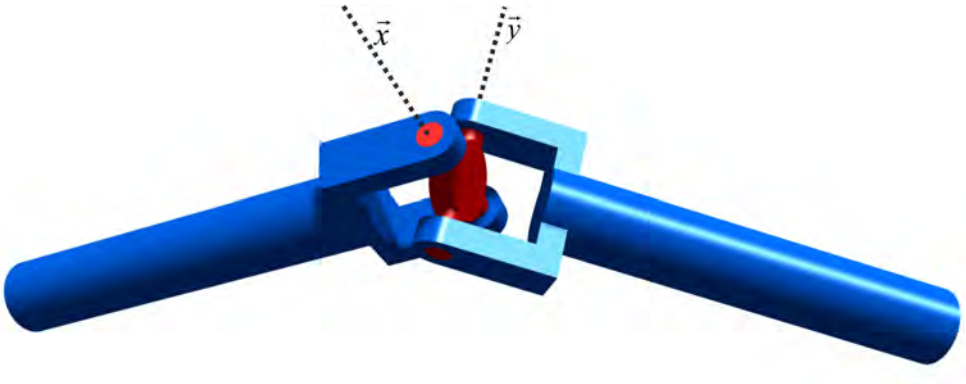


Abbildung 3.5.: 3D-Modell eines Kardan- oder Kreuzgelenks

Für die translatorischen Zwangsbedingungen gilt wiederum, dass sie denen des Kugelgelenks entsprechen. Für die translatorische Teil-Jacobimatrix gilt daher:

$$\mathbf{J}_{\text{kardan,tra}} = \mathbf{J}_{\text{kugel,tra}} \tag{3.43}$$

Zusätzlich muss die relative Drehbewegung um die dritte, „fixe“ Achse eingeschränkt werden. Die Bestimmung der fixen Achse des Kardans birgt eine Besonderheit: Sie ist in keinem der beiden Gelenkkoordinatensysteme konstant, sondern variiert in beiden zur

Simulationslaufzeit. Mit den oben eingeführten Rotationen der Gelenkkoordinatensysteme werden zunächst die beiden Drehachsen des Kardan $\vec{x}, \vec{y}$ bestimmt. Dabei wird eine Achse ($\vec{x}$) entsprechend ihrer relativen Lage zu Körper i, die andere ($\vec{y}$) entsprechend ihrer relativen Lage zu Körper j bestimmt:

$$\begin{aligned} \vec{x} &= \mathbf{R}_g^{W,Ki}(t) \cdot \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}^T \\ \vec{y} &= \mathbf{R}_g^{W,Kj}(t) \cdot \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}^T \end{aligned} \tag{3.44}$$

Die dritte und fixe Achse des Gelenks $\vec{z}_{\text{fix}}$ ergibt sich aus dem Kreuzprodukt von $\vec{x}$ und $\vec{y}$:

$$\vec{z}_{\text{fix}} = \vec{x} \times \vec{y} \tag{3.45}$$

Die relative Winkelgeschwindigkeit entlang der fixen Achse ergibt sich aus der Projektion der relativen Winkelgeschwindigkeiten der Körper i, j auf diese Achse. Die rotatorische Zwangsbedingung lautet also:

$$\begin{aligned} & \vec{z}_{\text{fix}} \cdot (\vec{\omega}_i - \vec{\omega}_j) = 0 \\ \Leftrightarrow \quad & \vec{z}_{\text{fix}} \cdot \vec{\omega}_i - \vec{z}_{\text{fix}} \cdot \vec{\omega}_j = 0 \end{aligned} \tag{3.46}$$

Die Teil-Jacobimatrizen des rotatorischen Teils des Kardan haben folglich die Form:

$$\begin{aligned} \mathbf{J}_{\text{kardan,rot},i} &= \begin{pmatrix} 0 & 0 & 0 & z_{\text{fix}_1} & z_{\text{fix}_2} & z_{\text{fix}_3} \end{pmatrix} \\ \mathbf{J}_{\text{kardan,rot},j} &= \begin{pmatrix} 0 & 0 & 0 & -z_{\text{fix}_1} & -z_{\text{fix}_2} & -z_{\text{fix}_3} \end{pmatrix} \end{aligned} \tag{3.47}$$

Stabilisierung

Der Term zur Stabilisierung der translatorischen Zwangsbedingungen ist der gleiche wie beim Dreh- und Kugelgelenk:

$$\vec{b}_{\text{error,kardan,tra}} = \vec{b}_{\text{error,kugel,tra}} \tag{3.48}$$

Der Fehler im Sinne der rotatorischen Zwangsbedingung lässt sich am Skalarprodukt der beiden Drehachsen ablesen. Liegt kein Fehler vor, muss das Skalarprodukt, also die Projektion der Achse $\vec{y}$ auf die Achse $\vec{x}$, gleich Null sein. Liegt ein rotatorischer Fehler

um ϵ Radiant vor, äußert sich das in diesem Skalarprodukt:

$$\sin(\epsilon) = \vec{x} \cdot \vec{y} \tag{3.49}$$

Ein rotatorischer Fehler um ϵ Radiant wird also durch eine relative Geschwindigkeit um die fixe Achse $\vec{z}_{\text{fix}}$ mit dem Betrag $\arcsin(\vec{x} \cdot \vec{y})$ in einem Zeitschritt korrigiert. Für den rotatorischen Stabilisierungsterm gilt damit:

$$b_{\text{error,kardan,rot}} = \frac{1}{h} \cdot k_{\text{er}} \arcsin(\vec{x} \cdot \vec{y}) \tag{3.50}$$

Für einen kleinen anzunehmenden Fehlerbetrag darf wiederum $\arcsin(x) \approx x$ angenommen werden:

$$b_{\text{error,kardan,rot}} = \frac{1}{h} \cdot k_{\text{er}} (\vec{x} \cdot \vec{y}) \tag{3.51}$$

3.2.6. Das Lineargelenk

Das Lineargelenk (vgl. Abbildung 3.6) entfernt alle rotatorischen und zwei translatorische Freiheitsgrade zwischen zwei Körpern. Alle Achsen des Gelenks in Form der Basisvektoren seines Koordinatensystems $\mathbf{T}^{W}_{\text{g,linear}}(t)$ lassen sich wie beim Drehgelenk direkt aus seiner relativen Transformation zu einem der verbundenen Körper ableiten. Sei o.B.d.A die z-Achse des Gelenks seine bewegliche Achse.

Da die Winkelgeschwindigkeit an jedem Punkt eines Starrkörpers gleich ist, ist die Formulierung der rotatorischen Zwangsbedingungen sehr einfach: Die relative Winkelgeschwindigkeit der Körper soll entlang jeder der drei Basisvektoren des Gelenks gleich Null sein.

$$\begin{aligned} \vec{x} \cdot \vec{\omega}_i - \vec{x} \cdot \vec{\omega}_j &= 0 \\ \vec{y} \cdot \vec{\omega}_i - \vec{y} \cdot \vec{\omega}_j &= 0 \\ \vec{z} \cdot \vec{\omega}_i - \vec{z} \cdot \vec{\omega}_j &= 0 \end{aligned} \tag{3.52}$$

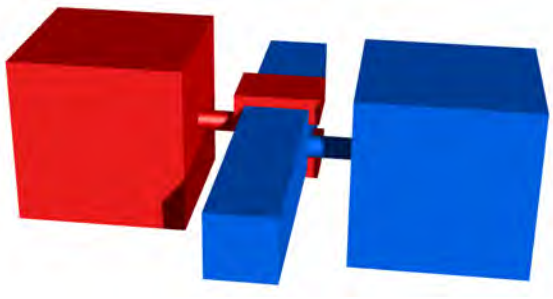


Abbildung 3.6.: 3D-Modell eines Lineargelenks

Hieraus lassen sich direkt die rotatorischen Teil-Jacobimatrizen ablesen:

$$\begin{aligned}\mathbf{J}_{\text{linear,rot},i} &= \begin{pmatrix} 0 & 0 & 0 & x_1 & x_2 & x_3 \\ 0 & 0 & 0 & y_1 & y_2 & y_3 \\ 0 & 0 & 0 & z_1 & z_2 & z_3 \end{pmatrix} \\ \mathbf{J}_{\text{linear,rot},j} &= \begin{pmatrix} 0 & 0 & 0 & -x_1 & -x_2 & -x_3 \\ 0 & 0 & 0 & -y_1 & -y_2 & -y_3 \\ 0 & 0 & 0 & -z_1 & -z_2 & -z_3 \end{pmatrix}\end{aligned} \tag{3.53}$$

Anmerkung: Wenn das Gelenk, wie oben beschrieben, drei rotatorische Freiheitsgrade einschränkt, können statt der Basisvektoren des Gelenks auch die Basisvektoren des Weltkoordinatensystems verwendet werden. In obiger Notation wären die Einträge x_1, y_2 und z_3 gleich Eins, alle anderen Einträge gleich Null. Nutzt man jedoch die Basisvektoren des Gelenkkoordinatensystems, kann man durch Aussparung z.B. der dritten Zeile in $\mathbf{J}_{\text{linear,rot}}$ zusätzlich einen rotatorischen Freiheitsgrad freigeben. Verglichen mit Abbildung 3.6 entspricht dies einem kreisrunden anstatt des quadratischen Querschnitts der Gelenkschiene. Ein solches Gelenk bildet am besten einen Hydraulikzylinder mit kreisrundem Querschnitt ab.

Die relative translatorische Geschwindigkeit der Ankerpunkte muss in zwei der drei Basisrichtungen des Gelenks beschränkt werden. Seien $\vec{x}, \vec{y}$ die zwei orthogonalen Richtungen des Gelenks, in denen keine relative translatorische Bewegung möglich ist. Seien wiederum $\vec{r}_i, \vec{r}_j$ die Vektoren vom jeweiligen Schwerpunkt von Körper i bzw. j zum Ankerpunkt. Dann ergibt sich die relative, translatorische Geschwindigkeit in der

eingeschränkten Richtung $\vec{x}$ wie folgt:

$$\begin{aligned} & \vec{x} \cdot (\vec{v}_j + \vec{\omega}_j \times \vec{r}_j - \vec{v}_i - \vec{\omega}_i \times \vec{r}_i) = 0 \\ \Rightarrow\ & \vec{x} \cdot \vec{v}_j - \vec{x} \cdot \vec{v}_i + \vec{x} \cdot \vec{\omega}_j \times \vec{r}_j - \vec{x} \cdot \vec{\omega}_i \times \vec{r}_i = 0 \\ \Rightarrow\ & \vec{x} \cdot \vec{v}_j - \vec{x} \cdot \vec{v}_i - \vec{x} \times \vec{r}_j \cdot \vec{\omega}_j + \vec{x} \times \vec{r}_i \cdot \vec{\omega}_i = 0 \end{aligned} \tag{3.54}$$

Für die zweite Richtung verläuft die Herleitung analog. Die Einträge in den Teil-Jacobimatrizen sind demnach:

$$\begin{aligned} \mathbf{J}_{\text{linear,tra},i} &= \begin{pmatrix} -x_1 & -x_2 & -x_3 & (\vec{x} \times \vec{r}_i)_1 & (\vec{x} \times \vec{r}_i)_2 & (\vec{x} \times \vec{r}_i)_3 \\ -y_1 & -y_2 & -y_3 & (\vec{y} \times \vec{r}_i)_1 & (\vec{y} \times \vec{r}_i)_2 & (\vec{y} \times \vec{r}_i)_3 \end{pmatrix} \\ \mathbf{J}_{\text{linear,tra},j} &= \begin{pmatrix} x_1 & x_2 & x_3 & -(\vec{x} \times \vec{r}_j)_1 & -(\vec{x} \times \vec{r}_j)_2 & -(\vec{x} \times \vec{r}_j)_3 \\ y_1 & y_2 & y_3 & -(\vec{y} \times \vec{r}_j)_1 & -(\vec{y} \times \vec{r}_j)_2 & -(\vec{y} \times \vec{r}_j)_3 \end{pmatrix} \end{aligned} \tag{3.55}$$

Stabilisierung

Der translatorische Fehler entspricht, wie schon beim Kugelgelenk, der Abweichung zwischen den Ankerpunkten des Gelenks, wenn man ihn einmal von Körper i und einmal von Körper j aus bestimmt.

$$\vec{e} = \mathbf{T}_g^{W,K_i} \cdot \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T - \mathbf{T}_g^{W,K_j} \cdot \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T \tag{3.56}$$

Da das Lineargelenk die translatorische Geschwindigkeit jedoch in nur zwei Richtungen einschränkt, die i.A. nicht zwei Basisvektoren des Weltkoordinatensystem entsprechen, muss dieser Fehler $\vec{e}$ noch in das Koordinatensystem des Gelenks transformiert werden. Dies geschieht am einfachsten durch die Projektion des Fehlers auf die Basisachsen des Gelenkkoordinatensystems $\vec{x}, \vec{y}$:

$$\vec{b}_{\text{error,linear,tra}} = \frac{1}{h} \cdot k_{\text{er}} \cdot \begin{pmatrix} \vec{x} \cdot \vec{e} \\ \vec{y} \cdot \vec{e} \end{pmatrix} \tag{3.57}$$

Ein rotatorischer Fehler äußert sich in einer Abweichung zwischen den Orientierungen des Gelenks, wenn sie einmal aus seiner relativen Lage zu Körper i und einmal aus seiner relativen Lage zu Körper j bestimmt wird. Dies entspricht einer Differenz zwischen $\mathbf{R}_{\text{g}}^{W,K_i}$ und $\mathbf{R}_{\text{g}}^{W,K_j}$. Seien diese Rotationen repräsentiert durch die Rotationsquaternione (siehe Anhang A.5) $\vec{q}_i$ und $\vec{q}_j$, dann entspricht das „Fehlerquaternion“ der

Rotation von $\vec{q}_j$ „aus Sicht von“ $\vec{q}_i$: $\vec{q}_{\text{error}} = \vec{q}_i^{-1} \cdot \vec{q}_j$. Das Rotationsquaternion setzt sich wie folgt zusammen:

$$\vec{q}_{\text{error}} = \left(\cos\left(\frac{\epsilon}{2}\right), \vec{v}\sin\left(\frac{\epsilon}{2}\right)\right) \tag{3.58}$$

Der Vektor $\vec{v}$ ist der normierte Drehvektor, entlang dessen ein rotatorischer Fehler vorliegt, ϵ der Betrag der Verdrehung. Eine korrigierende relative Geschwindigkeit muss genau entlang dieses Vektors verlaufen. Da wegen des anzunehmenden kleinen ϵ wiederum näherungsweise $\sin(\epsilon) \approx \epsilon$ angenommen werden darf, ergibt sich der rotatorische Fehlerterm zu:

$$\vec{b}_{\text{error,linear,rot}} = \frac{1}{h} \cdot k_{\text{er}} \cdot 2 \cdot \vec{v} \tag{3.59}$$

3.2.7. Gelenkanschläge oder -limits

Reale Gelenke werden häufig durch Maximalausschläge begrenzt. Besitze ein Drehgelenk zum Beispiel die maximalen Auslenkungen von $-\pi \ldots + \pi$. Ist der Maximalausschlag erreicht und verlangt das Mehrkörpersystem eine noch größere Auslenkung, muss eine zusätzliche Zwangsbedingung dafür sorgen, dass der entsprechende Freiheitsgrad beschränkt wird, jedoch nur in derjenigen Richtung, in der der Maximalausschlag überschritten würde. Diese Eigenschaft charakterisiert solche Zwangsbedingungen als *unilateral* - sie wirken nur in *eine* Richtung.

Alle oben beschriebenen Gelenktypen lassen sich um Limits ergänzen. Da die Realisierung bei den unterschiedlichen Typen sehr ähnlich ist, wird im Folgenden beispielhaft die Limitierung des Drehgelenks beschrieben. Die Limitierung anderer Gelenktypen unterscheidet sich lediglich geringfügig in der Form der Teil-Jacobimatrizen: Je nach Gelenktyp müssen andere Freiheitsgrade eingeschränkt werden.

Limitierung eines Drehgelenks

Das Drehgelenk aus Abschnitt 3.2.4 besitze einen Minimalausschlag von $-\pi$. Ist dieser Minimalausschlag erreicht, muss eine rotatorische Zwangsbedingung eingeführt werden, die eine weitere Auslenkung in negativer Richtung des Drehsinns um die Drehachse verhindert. Dies wird erreicht, indem für die entsprechende relative Winkelgeschwindigkeit ein Wert größer oder gleich Null gefordert wird. So kann sich das Gelenk

vom Grenzwert weg in Richtung des gültigen Bereichs, jedoch nicht weiter über den Grenzwert hinaus bewegen:

$$\begin{aligned} \vec{z} \cdot (\vec{\omega}_j - \vec{\omega}_i) &= b, b \geq 0 \\ \vec{z} \cdot \vec{\omega}_j - \vec{z} \cdot \vec{\omega}_i &= b, b \geq 0 \end{aligned} \tag{3.60}$$

Um die Form aus 3.21 zu erfüllen, wird in obiger Gleichung die Schlupfvariable w eingeführt. Sie stellt den gültigen Bereich der relativen Geschwindigkeit zur Erfüllung der Zwangsbedingung dar. Mit ihrer Hilfe kann die Größergleich-Bedingung in eine Gleichung mit Nebenbedingung umformuliert werden:

$$\vec{z} \cdot \vec{\omega}_j - \vec{z} \cdot \vec{\omega}_i = b + w, \;\; b = 0, \;\; 0 \leq w \leq \infty \tag{3.61}$$

Aus dieser Form ergeben sich die Einträge für die Teil-Jacobimatrizen:

$$\begin{aligned} \mathbf{J}_{\text{dreh,limit},i} &= \begin{pmatrix} 0 & 0 & 0 & -z_1 & -z_2 & -z_3 \end{pmatrix} \\ \mathbf{J}_{\text{dreh,limit},j} &= \begin{pmatrix} 0 & 0 & 0 & z_1 & z_2 & z_3 \end{pmatrix} \end{aligned} \tag{3.62}$$

Auch der aus dieser Zwangsbedingung resultierende Zwangsimpuls muss begrenzt werden: Er darf beliebig groß werden, um eine weitere Drehung über den Grenzwert hinaus zu verhindern, jedoch nicht kleiner als Null, da er dann auch eine Gelenkbewegung weg vom Grenzwert in Richtung des gültigen Bereiches verhindern oder verzögern würde:

$$0 \leq \lambda_{\text{dreh,limit}} \leq \infty \tag{3.63}$$

Die beiden Größen relative Geschwindigkeit, repräsentiert durch die Schlupfvariable w und aufgebrachter Zwangsimpuls λ_{limit} sind komplementär zueinander: Bleibt das Gelenk im Anschlag, ist die relative Geschwindigkeit gleich Null ($w = 0$) und der Zwangsimpuls größer als Null ($\lambda_{\text{limit}} > 0$). Bewegt es sich vom Anschlagspunkt weg, so ist die relative Geschwindigkeit größer als Null ($w > 0$), die Zwangsbedingung übt jedoch keinen Impuls mehr aus ($\lambda_{\text{limit}} = 0$). Damit erfüllt diese Zwangsbedingung die gleiche Form wie die Kontaktzwangsbedingungen in der vereinfachten geschwindigkeitsbasierten Formulierung in Gleichung 3.21.

Stabilisierung

Auch Limit-Zwangsbedingungen müssen stabilisiert werden. Zwar verhindert die Zwangsbedingung eine relative Geschwindigkeit, die zu einer größeren Verletzung des Maximalausschlags führt. Eine einmal entstandene Überschreitung des Grenzwerts, hervorgerufen z.B. durch einen großen Zeitschritt, wird jedoch nicht korrigiert. Genau wie zur Stabilisierung der Gelenkbedingungen wird hierzu eine korrigierende Relativgeschwindigkeit in Richtung des gültigen Auslenkungsbereichs proportional zur Größe des Fehlers erzwungen. Sei die maximale Auslenkung des Gelenks $\alpha_{\max}$ und die gegenwärtige Auslenkung $\alpha > \alpha_{\max}$, dann ergibt sich der Stabilisierungsterm zu:

$$b_{\text{error,limit,rot}} = \frac{1}{h} \cdot k_{\text{er}} \cdot (\alpha_{\max} - \alpha) \tag{3.64}$$

3.2.8. Realisierung von Motoren durch Zwangsbedingungen

Genau wie Limits können auch Motoren in Gelenken durch zusätzliche Zwangsbedingungen realisiert werden. Wiederum gilt hier, dass die Motorisierung der unterschiedlichen Gelenktypen sehr ähnlich ist. Die Unterschiede liegen lediglich in der Form der Teil-Jacobimatrizen. An dieser Stelle wird exemplarisch ein Linearmotor in einem Lineargelenk realisiert. Natürlich können Motoren auch realisiert werden, indem externe Kräfte oder Momente in das System eingebracht werden. Hierbei entsteht jedoch das Problem eines relativ großen Integrationsfehlers, da Kraft- und Momentwerte dann zweifach explizit integriert werden. Die Kraft- und Momentwerte, die sich aus der Realisierung durch Zwangsbedingungen ergeben, gehen in ein semi-implizites Integrationsschema ein. Sie werden darüber hinaus derart bestimmt, dass vorgegebene Sollwerte für die resultierenden Geschwindigkeiten nicht überschritten werden. Hierdurch ergeben sich ein deutlich besseres Stabilitätsverhalten und die Möglichkeit, größere Zeitschrittweiten und damit ein günstigeres Laufzeitverhalten zu erreichen.

Realisierung eines Motors durch Zwangsbedingungen in einem Lineargelenk

In Kapitel 3.2.6 wird beschrieben, wie zur Realisierung eines Lineargelenks bereits fünf von sechs relativen Freiheitsgraden zwischen zwei Körpern durch das Lineargelenk entfernt werden. Ein Motor muss entsprechend den sechsten und letzten verbliebenen, translatorischen Freiheitsgrad in Richtung der Gelenkachse entfernen. Aus Kapitel 3.2.6

lassen sich somit direkt die Einträge der Teil-Jacobimatrizen ablesen:

$$\begin{aligned}\mathbf{J}_{\text{linearmotor,trans},i} &= \begin{pmatrix} -z_1 & -z_2 & -z_3 & (\vec{z}\times\vec{r}_i)_1 & (\vec{z}\times\vec{r}_i)_2 & (\vec{z}\times\vec{r}_i)_3 \end{pmatrix} \\ \mathbf{J}_{\text{linearmotor,trans},j} &= \begin{pmatrix} z_1 & z_2 & z_3 & -(\vec{z}\times\vec{r}_j)_1 & -(\vec{z}\times\vec{r}_j)_2 & -(\vec{z}\times\vec{r}_j)_3 \end{pmatrix}\end{aligned} \tag{3.65}$$

Nun soll die relative Geschwindigkeit jedoch nicht gleich Null, sondern gleich einer Motor-Sollgeschwindigkeit b_{motor} sein. Entsprechend wird die rechte Seite in dieser Zwangsbedingung nicht gleich Null, sondern gleich der Sollgeschwindigkeit b_{motor} gesetzt.

$$\begin{aligned}\mathbf{J}_{\text{linearmotor,trans}} \cdot \vec{v} &= b + w, \quad b = b_{\text{motor}} \\ w &: \text{Abweichung zwischen relativer Ist- und Sollgeschwindigkeit}\end{aligned} \tag{3.66}$$

Motoren können uni- oder bilateral formuliert werden. Bei einer bilateralen Formulierung kann der Motor gleichermaßen beschleunigen und bremsen, d.h. wird die Sollgeschwindigkeit gegenwärtig unterschritten, so wirkt der Motor beschleunigend, wird sie überschritten, so wirkt er bremsend. Die maximale Kraft des Linearmotors sei $f_{\max}$. Dann gilt für die Grenzwerte der Lagrange-Multiplikatoren:

$$\lambda_{\text{motor,high}} = hf_{\max},\ \lambda_{\text{motor,low}} = -hf_{\max} \tag{3.67}$$

Motoren z.B. im Fahrzeugantrieb verhalten sich zu einem Zeitpunkt typischerweise unilateral, das heißt sie bringen in einer Wirkrichtung ein deutlich größeres Moment bzw. eine deutlich größere Kraft auf als in der anderen. Am Beispiel Auto bedeutet das: Fährt ein Auto z.B. durch das Gefälle der Fahrbahn bereits schneller, als es die Solldrehzahl des Motors hervorrufen würde, wirkt der Motor nicht oder nur in geringem Ausmaß (z.B. 1 Prozent von $f_{\max}$) bremsend. Dieses Verhalten lässt sich durch geeignete Wahl der Grenzwerte der zugehörigen Lagrange-Multiplikatoren bei der Formulierung der Zwangsbedingungen nachempfinden. Im Beispiel gilt für sie:

$$\begin{aligned}\lambda_{\text{motor,high}} = hf_{\max},\ \lambda_{\text{motor,low}} &= -0,01 \cdot hf_{\max} \\ 0 < w &< \infty\end{aligned} \tag{3.68}$$

Im Fall der unilateralen Formulierung gilt für w und den zugehörigen Lagrange-Multiplikator λ_{motor} die gleiche Komplementarität, wie schon für Limitbedingungen: Wird die Sollgeschwindigkeit unterschritten ($w < 0$), bringt der Motor im selben Zeitschritt

den maximal möglichen Impuls auf, um die Sollgeschwindigkeit zu erreichen, ohne dass dieser jedoch seinen oberen Grenzwert überschreitet: $\lambda_{\text{motor}} = \lambda_{\text{high}}$. Wird die Sollgeschwindigkeit überschritten ($w > 0$), bringt der Motor den minimal möglichen Impuls auf, der jedoch seinen unteren Grenzwert nicht unterschreitet: $\lambda_{\text{motor}} = \lambda_{\text{motor,low}}$. Nur wenn die Sollgeschwindigkeit genau erreicht wird, liegt der aufgebrachte Impuls zwischen den Grenzwerten des Motors $\lambda_{\text{motor,low}} \leq \lambda_{\text{motor}} \leq \lambda_{\text{motor,high}}$. Zur Verdeutlichung dieses Zusammenhangs siehe auch Abbildung 3.2 auf Seite 86.

Positionssteller Die Realisierung von Motoren mit Zwangsbedingungen im vorgestellten Framework führt zu geschwindigkeitsgestellten Motoren. In vielen Anwendungen werden jedoch Positionssteller gefordert, die direkt einen bestimmten Winkel oder eine Auslenkung anfahren können, so z.B. bei der Anbindung bestimmter Robotersteuerungen, die Trajektorien in Form von Winkelwerten ausgeben oder zur Realisierung einer Fahrzeuglenkung. Mit den hier vorgestellten, geschwindigkeitsgestellten Motoren lassen sich durch Einsatz z.B. eines PID-Reglers problemlos solche Positionssteller erzeugen. Im Rahmen der Beschreibung der Realisierung einer Vorderradaufhängung an einem PKW-Modell (vgl. Kapitel 3.4.4) wird ein solches Vorgehen im Detail beschrieben.

3.3. Geschwindigkeitsbezogene Zwangsbedingungen für Kontakte und Kontaktreibung

Zwangsbedingungen für Kontakte wurden während der Entwicklung der geschwindigkeitsbasierten Formulierung für die inverse Dynamik bereits implizit verwendet. An dieser Stelle folgt nun die explizite Beschreibung solcher Zwangsbedingungen. Kontaktnormalenbedingungen verhindern die Durchdringung von in Kontakt stehenden Körpern, Kontaktreibungsbedingungen erzwingen Coulombsche Reibungskräfte.

3.3.1. Formulierung von Kontaktnormalenzwangsbedingungen

Abbildung 3.7 zeigt eine 2D-Darstellung eines Kontaktes zwischen den Körpern i und j. In der Abbildung sind $\vec{s}_i$ und $\vec{s}_j$ die jeweiligen Schwerpunktspositionen, $\vec{r}_i$ und $\vec{r}_j$ die Vektoren vom jeweiligen Schwerpunkt zum Kontaktpunkt. $\vec{n}$ ist die Kontaktnormale. Alle Vektoren sind in Weltkoordinaten gegeben.

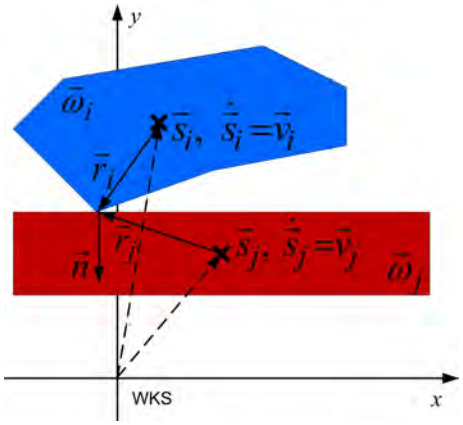


Abbildung 3.7.: 2D-Darstellung eines Kontaktpunktes

Die Kontaktnormalenzwangsbedingung soll erreichen, dass keine Durchdringung der beteiligten Körper in Richtung der Kontaktnormalen stattfindet. Auf der Ebene von Geschwindigkeiten heißt das: Ausgehend von einer gültigen Position, in der keine Durchdringung auftritt, muss die relative Geschwindigkeit der in Kontakt stehenden Punkte der beiden Körper (ihrer Ankerpunkte) entlang der Kontaktnormalen größer oder gleich Null sein:

$$\begin{aligned} \vec{n} \cdot (\vec{v}_j + \vec{\omega}_j \times \vec{r}_j - \vec{v}_i - \vec{\omega}_i \times \vec{r}_i) &= b \\ \vec{n} \cdot \vec{v}_j - \vec{n} \cdot (\vec{r}_j \times \vec{\omega})_j - \vec{n} \cdot \vec{v}_i + \vec{n} \cdot (\vec{r}_i \times \vec{\omega})_i &= b \\ \vec{n} \cdot \vec{v}_j - (\vec{n} \times \vec{r}_j) \cdot \vec{\omega}_j - \vec{n} \cdot \vec{v}_i + (\vec{n} \times \vec{r}_i) \cdot \vec{\omega}_i &= b \\ \text{mit } b = 0 & \end{aligned} \tag{3.69}$$

Hieraus ergeben sich die Einträge der Teil-Jacobimatrizen wie folgt:

$$\begin{aligned} \mathbf{J}_i &= \begin{pmatrix} -n_1 & -n_2 & -n_3 & (\vec{n} \times \vec{r}_i)_1 & (\vec{n} \times \vec{r}_i)_2 & (\vec{n} \times \vec{r}_i)_3 \end{pmatrix} \\ \mathbf{J}_j &= \begin{pmatrix} n_1 & n_2 & n_3 & -(\vec{n} \times \vec{r}_j)_1 & -(\vec{n} \times \vec{r}_j)_2 & -(\vec{n} \times \vec{r}_j)_3 \end{pmatrix} \end{aligned} \tag{3.70}$$

Ein Kontakt darf die beteiligten Körper auseinanderdrücken, darf aber nicht ziehend wirken. Entsprechend gilt für die Grenzwerte der Lagrange-Multiplikatoren:

$$\lambda_{\text{low}} = 0, \quad \lambda_{\text{high}} = \infty \tag{3.71}$$

Stabilisierung

Wie oben beschrieben geht die geschwindigkeitsbezogene Kontaktnormalenzwangsbedingung von einem gültigen Ausgangszustand aus, d.h. von der Annahme, dass keine Durchdringung vorliegt. Ist jedoch bereits eine Durchdringung der beiden Körper der Tiefe e vorhanden, so soll die Zwangsbedingung eine relative Geschwindigkeit erzwingen, die größer oder gleich einer Korrekturgeschwindigkeit ist. Diese Korrekturgeschwindigkeit wird wiederum einfach proportional zum Fehler (der Durchdringungstiefe) angesetzt:

$$b_{\text{error,contact}} = \frac{1}{h} \cdot k_{\text{er}} \cdot e \tag{3.72}$$

3.3.2. Formulierung von Kontaktreibungszwangsbedingungen

Zur Modellierung von Kontaktreibung muss die Geschwindigkeit der in Kontakt stehenden Körper entlang der Basisvektoren des diskretisierten Reibungskonus $\vec{d_n}$ (vgl. Abbildung 3.1 auf Seite 84) eingeschränkt werden. Seien $\vec{v}_i, \vec{v}_j$ die linearen Geschwindigkeiten der Schwerpunkte der in Kontakt stehenden Körper, $\vec{\omega}_i, \vec{\omega}_j$ analog dazu ihre Winkelgeschwindigkeiten. Seien außerdem $\vec{r}_i, \vec{r}_j$ die Vektoren vom jeweiligen Schwerpunkt zum gemeinsamen Kontaktpunkt im Weltkoordinatensystem, dann lautet die Zwangsbedingung zur Einschränkung der relativen Geschwindigkeit v_{rel} entlang einer diskreten Richtung $\vec{d_n}$:

$$v_{\text{rel}} = \vec{d_n} \cdot (\vec{v}_i + \vec{\omega}_i \times \vec{r}_i) - \vec{d_n} \cdot (\vec{v}_j + \vec{\omega}_j \times \vec{r}_j) = b + w,\ b = 0, w\text{: Schlupf} \tag{3.73}$$

Tritt Rutschen auf, nimmt die Schlupfvariable w Werte ungleich Null an.

Um diese Zwangsbedingung in die Matrix-Vektor-Form $\mathbf{J}\vec{v} = \vec{b} + \vec{w}$ überführen zu können, sind noch einige Umformungen notwendig:

$$\begin{aligned}
v_{\text{rel}} &= \vec{d_n} \cdot (\vec{v}_i + \vec{\omega}_i \times \vec{r}_i) - \vec{d_n} \cdot (\vec{v}_j + \vec{\omega}_j \times \vec{r}_j) && = b + w,\ b = 0 \\
&= \vec{d_n} \cdot \vec{v}_i + \vec{d_n} (\vec{\omega}_i \times \vec{r}_i) - \vec{d_n} \cdot \vec{v}_j - \vec{d_n} (\vec{\omega}_j \times \vec{r}_j) && = b + w,\ b = 0 \\
&= \vec{d_n} \cdot \vec{v}_i - \vec{d_n} (\vec{r}_i \times \vec{\omega}_i) - \vec{d_n} \cdot \vec{v}_j + \vec{d_n} (\vec{r}_j \times \vec{\omega}_j) && = b + w,\ b = 0 \\
&= \vec{d_n} \cdot \vec{v}_i - \left(\vec{d_n} \times \vec{r}_i\right) \cdot \vec{\omega}_i - \vec{d_n} \cdot \vec{v}_j + \left(\vec{d_n} \times \vec{r}_j\right) \cdot \vec{\omega}_j && = b + w,\ b = 0
\end{aligned} \tag{3.74}$$

Nimmt man eine Approximation mit zwei Basisvektoren $\vec{d}_1, \vec{d}_2$ an, ergeben sich hieraus

die Einträge der Teil-Jacobimatrizen zu:

$$\begin{aligned}
\mathbf{J}_{\text{contact,friction},i} &= \begin{pmatrix} d_{1_1} & d_{1_2} & d_{1_3} & -\left(\vec{d_1} \times \vec{r_i}\right)_1 & -\left(\vec{d_1} \times \vec{r_i}\right)_2 & -\left(\vec{d_1} \times \vec{r_i}\right)_3 \\ d_{2_1} & d_{2_2} & d_{2_3} & -\left(\vec{d_2} \times \vec{r_i}\right)_1 & -\left(\vec{d_2} \times \vec{r_i}\right)_2 & -\left(\vec{d_2} \times \vec{r_i}\right)_3 \end{pmatrix} \\
\mathbf{J}_{\text{contact,friction},j} &= \begin{pmatrix} -d_{1_1} & -d_{1_2} & -d_{1_3} & \left(\vec{d_1} \times \vec{r_j}\right)_1 & \left(\vec{d_1} \times \vec{r_j}\right)_2 & \left(\vec{d_1} \times \vec{r_j}\right)_3 \\ -d_{2_1} & -d_{2_2} & -d_{2_3} & \left(\vec{d_2} \times \vec{r_j}\right)_1 & \left(\vec{d_2} \times \vec{r_j}\right)_2 & \left(\vec{d_2} \times \vec{r_j}\right)_3 \end{pmatrix}
\end{aligned} \tag{3.75}$$

Die Grenzwerte der Lagrange-Multiplikatoren für die Reibungszwangsbedingungen können nicht im Vorhinein bestimmt werden, da ihre Beträge von den Beträgen der Lagrange-Multiplikatoren abhängen, die die zugehörigen Kontaktnormalenimpulse repräsentieren. Daher wird während der Formulierung der Reibungszwangsbedingungen an der Stelle der Grenzwerte lediglich ein Verweis auf die zu einer Reibungszwangsbedingung gehörende Kontaktnormalenzwangsbedingung hinterlegt. Mithilfe dieses Verweises kann ein Lösungsalgorithmus dann auf die betreffenden Multiplikatoren zugreifen und das proportionale Verhältnis zwischen Normalen- und Reibungsimpuls herstellen. Details solcher Lösungsalgorithmen behandelt Kapitel 3.5.

3.4. Geschwindigkeitsbezogene Zwangsbedingungen für erweiterte mechanische Zusammenhänge

Um die Flexibilität des hier realisierten Verfahrens zu belegen, werden in diesem Kapitel Zwangsbedingungen für weitere mechanische Zusammenhänge formuliert, die über die klassischen Gelenktypen hinausgehen. Hierzu zählen eine Schraubverbindung, Differenzialbedingungen, das Feder-Dämpfer-System sowie eine PKW-Vorderradaufhängung durch ein einziges, integriertes Gelenk.

3.4.1. Das Schraubengelenk

Abbildung 3.8 zeigt das Modell einer Schraubverbindung. Sie ist ein schönes Beispiel einer erweiterten Gelenkverbindung, die sich sehr gut mit geschwindigkeitsbezogenen Zwangsbedingungen realisieren lässt. Eine Schraubverbindung lässt eine lineare und eine rotatorische relative Bewegung zu, besitzt aber dennoch nur einen Freiheitsgrad: Tritt eine rotatorische Bewegung auf, muss auch eine translatorische stattfinden und

umgekehrt, so dass sich mit der Schraubverbindung ein Drehmoment in eine Kraft übersetzen lässt und umgekehrt.

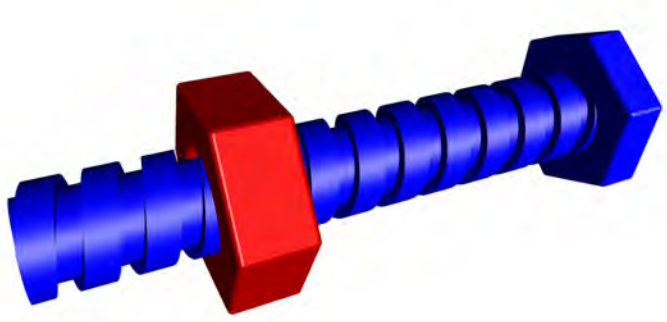


Abbildung 3.8.: 3D-Modell einer Schraubverbindung

Das Koordinatensystem des Gelenktyps sei so ausgerichtet, dass seine z-Achse diejenige ist, um die bzw. entlang derer die rotatorische und translatorische Bewegung stattfinden. Zunächst muss die relative translatorische Geschwindigkeit entlang der x- und der y-Achse des Gelenkkoordinatensystems der Schraubverbindung eingeschränkt werden. Dies geschieht wie beim Lineargelenk:

$$\begin{aligned}
\mathbf{J}_{\text{schraube,tra},i} &= \begin{pmatrix} -x_1 & -x_2 & -x_3 & (\vec{x}\times\vec{r}_i)_1 & (\vec{x}\times\vec{r}_i)_2 & (\vec{x}\times\vec{r}_i)_3 \\ -y_1 & -y_2 & -y_3 & (\vec{y}\times\vec{r}_i)_1 & (\vec{y}\times\vec{r}_i)_2 & (\vec{y}\times\vec{r}_i)_3 \end{pmatrix} \\
\mathbf{J}_{\text{schraube,tra},j} &= \begin{pmatrix} x_1 & x_2 & x_3 & -(\vec{x}\times\vec{r}_j)_1 & -(\vec{x}\times\vec{r}_j)_2 & -(\vec{x}\times\vec{r}_j)_3 \\ y_1 & y_2 & y_3 & -(\vec{y}\times\vec{r}_j)_1 & -(\vec{y}\times\vec{r}_j)_2 & -(\vec{y}\times\vec{r}_j)_3 \end{pmatrix}
\end{aligned} \tag{3.76}$$

Analog dazu wird die relative rotatorische Geschwindigkeit um dieselben Achsen entsprechend dem Drehgelenk eingeschränkt:

$$\begin{aligned}
\mathbf{J}_{\text{schraube,rot},i} &= \begin{pmatrix} 0 & 0 & 0 & x_1 & x_2 & x_3 \\ 0 & 0 & 0 & y_1 & y_2 & y_3 \end{pmatrix} \\
\mathbf{J}_{\text{schraube,rot},j} &= \begin{pmatrix} 0 & 0 & 0 & -x_1 & -x_2 & -x_3 \\ 0 & 0 & 0 & -y_1 & -y_2 & -y_3 \end{pmatrix}
\end{aligned} \tag{3.77}$$

Bis hierher sind vier von sechs Freiheitsgraden eingeschränkt, d.h. das Gelenk würde noch zwei Freiheitsgrade besitzen. Mit der fünften Zwangsbedingung müssen die relativen Bewegungen entlang der bzw. um die z-Achse ($\vec{z}$) des Gelenkkoordinatensystems gekoppelt werden. Die Schraubverbindung besitze eine konstante *Steigung* von z.B.

1,0 cm (0,01 m) pro Umdrehung. Eine Umdrehung muss also eine Verschiebung um 1 cm hervorrufen bzw. umgekehrt eine Verschiebung um 1 m muss 100 ganze Umdrehungen zur Folge haben. Auf der Ebene der Geschwindigkeiten entspricht dies einem direkten linearen Zusammenhang zwischen relativer Winkel- und relativer translatorischer Geschwindigkeit. Bei einer relativen translatorischen Geschwindigkeit von 1 m/s muss eine Winkelgeschwindigkeit von 100 Umdrehungen/s, also $100 \cdot 2\pi \frac{\mathrm{rad}}{s}$ auftreten. Mit der Schraubenkonstante $k_s = 200\pi\mathrm{rad}$ lässt sich dies formulieren als:

$$\begin{aligned} k_s \cdot \vec{z} \cdot (\vec{v}_j - \vec{v}_i) &= \vec{z} \cdot (\vec{\omega}_j - \vec{\omega}_i) \\ k_s \cdot \vec{z} \cdot \vec{v}_j - k_s \cdot \vec{z} \cdot \vec{v}_i - \vec{z} \cdot \vec{\omega}_j + \vec{z} \cdot \vec{\omega}_i &= 0 \end{aligned} \tag{3.78}$$

Hieraus lassen sich die Einträge der fünften, noch fehlenden Zeilen (***kom**biniert*) der Teil-Jacobimatrizen ablesen:

$$\begin{aligned} \mathbf{J}_{\mathrm{schraube,kom},i} &= \begin{pmatrix} -k_s z_1 & -k_s z_2 & -k_s z_3 & z_1 & z_2 & z_3 \end{pmatrix} \\ \mathbf{J}_{\mathrm{schraube,kom},j} &= \begin{pmatrix} k_s z_1 & k_s z_2 & k_s z_3 & -z_1 & -z_2 & -z_3 \end{pmatrix} \end{aligned} \tag{3.79}$$

Stabilisierung

Genau wie die Einträge der Teil-Jacobimatrizen lassen sich auch die zugehörigen Stabilisierungsterme der ersten vier Zwangsbedingungen vom Dreh- bzw. Lineargelenk ableiten:

$$\begin{aligned} \vec{b}_{\mathrm{schraube,tra}} &= \vec{b}_{\mathrm{dreh,tra}} \\ \vec{b}_{\mathrm{schraube,rot}} &= \vec{b}_{\mathrm{linear,rot}} \end{aligned} \tag{3.80}$$

Ein Fehler in der letzten, kombinierten Zwangsbedingung äußert sich in einer Abweichung zwischen Verdrehung und linearer Auslenkung. Die lineare Auslenkung e_l des Gelenks lässt sich direkt aus den Positionen der verbundenen Körper bestimmen. Hieraus ergibt sich über die Schraubensteigung k_s ein Sollwert für die rotatorische Auslenkung e_r:

$$e_r = k_s \cdot e_l \tag{3.81}$$

Die tatsächliche rotatorische Auslenkung des Gelenks θ lässt sich zu einem Zeitpunkt anhand des Zustands der beiden Körper nur im Bereich $-2\pi \ldots 0$ bzw. $0 \ldots 2\pi$

bestimmen. Um die Abweichung in der rotatorischen Auslenkung ermitteln zu können, müssen daher die bereits vollständig zurückgelegten n Umdrehungen vom Sollwert abgezogen werden. Diesen Wert erhält man durch stetige Beobachtung und Integration der Geschwindigkeiten am Schraubengelenk. Damit lautet der Stabilisierungsterm für die kombinierte Zwangsbedingung:

$$b_{\text{schraube,kom}} = \frac{1}{h} \cdot k_{\text{er}} \left(k_s e_l - n2\pi - \theta \right) \tag{3.82}$$

3.4.2. Differenzialbedingungen und Differenzialgetriebe

Alle bisher beschriebenen mechanischen Zusammenhänge beziehen sich auf *zwei* in Relation stehende Körper. Das entwickelte Framework unterstützt jedoch auch die Realisierung von dynamischen Zusammenhängen zwischen mehr als zwei Körpern. Abbildung 3.9 zeigt das Modell eines freilaufenden Differenzialgetriebes. Die drei relevanten Körper sind in blau, rot und gelb dargestellt. Zusätzlich zu den zwei Drehgelenken zwischen den farbigen Körpern ist eine weitere Zwangsbedingung notwendig, um die Funktion der grauen Zahnräder im System nachzubilden. Sei $\vec{d}_r$ der Vektor, der die Drehachse des roten Zahnrades beschreibt und $\vec{d}_g$ analog dazu der Vektor, der die Drehachse des gelben Zahnrades beschreibt. Seien weiterhin die Winkelgeschwindigkeiten des blauen, roten und gelben Körpers gegeben durch die Vektoren $\vec{\omega}_b, \vec{\omega}_r, \vec{\omega}_g$. Hält man das große, graue und halbtransparente Zahnrad fest, sorgt das Differenzial dafür, dass sich rotes und gelbes Zahnrad in entgegengesetzter Richtung mit gleicher Geschwindigkeit drehen werden. Formal bedeutet das:

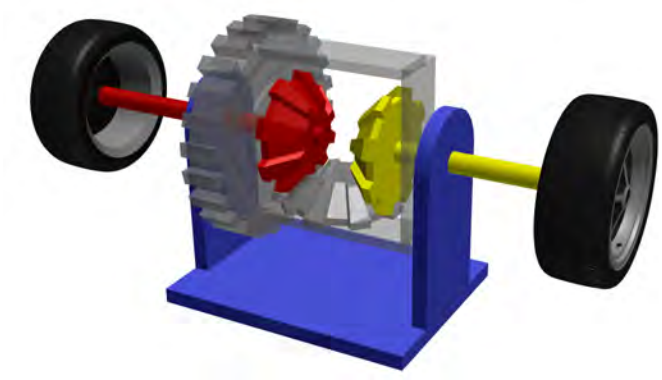


Abbildung 3.9.: Differenzialgetriebe zwischen drei Körpern

$$\vec{d_r} \cdot (\vec{\omega}_b - \vec{\omega}_r) - \vec{d_g} \cdot (\vec{\omega}_b - \vec{\omega}_g) = 0 \tag{3.83}$$

Beim Einsatz eines solchen Getriebes als Querdifferenzial beim Radantrieb im Auto sind das rote und das gelbe Zahnrad je mit linkem und rechtem Rad verbunden, das große graue Zahnrad hingegen mit dem Motor, es wird angetrieben. Nimmt man b_m als die Winkelgeschwindigkeit des großen, grauen Zahnrades an, ergibt sich der formale Zusammenhang:

$$\vec{d_r} \cdot (\vec{\omega}_b - \vec{\omega}_r) - \vec{d_g} \cdot (\vec{\omega}_b - \vec{\omega}_g) = 2b_m \tag{3.84}$$

Dieser kann weiter verallgemeinert werden, indem ein Skalierungsfaktor c_d zwischen den beiden Relativgeschwindigkeiten eingeführt wird. Bei obigem Differenzialgetriebe wurde dieser implizit als -1 angenommen:

$$\begin{aligned} \vec{d_r} \cdot (\vec{\omega}_b - \vec{\omega}_r) + c_d \cdot \vec{d_g} \cdot (\vec{\omega}_b - \vec{\omega}_g) &= 2b_m \\ \vec{d_r} \cdot \vec{\omega}_b - \vec{d_r} \cdot \vec{\omega}_r - c_d \vec{d_g} \cdot \vec{\omega}_b + c_d \vec{d_g} \cdot \vec{\omega}_g &= 2b_m \\ \left(\vec{d_r} - c_d \vec{d_g}\right) \cdot \vec{\omega}_b - \vec{d_r} \cdot \vec{\omega}_r + c_d \vec{d_g} \cdot \vec{\omega}_g &= 2b_m \end{aligned} \tag{3.85}$$

Mit einer solchen Bedingung lassen sich Gelenke direkt miteinander verbinden, z.B. um beliebige Getriebeübersetzungen zu simulieren. Abbildung 3.10 zeigt drei weitere Anwendungen: Bei dem Greifer links müssen sich beide Zangen immer mit gleich großer, jedoch entgegengesetzter Geschwindigkeit gegenüber der Basis bewegen. Bei dem Teleskopgelenk des Krans in der Mitte muss sich das dritte Glied zum zweiten verhalten wie das zweite zum ersten. Die Feuerwehrleiter des Einsatzfahrzeugs verhält sich ebenfalls wie ein Teleskopgelenk mit fünf Gliedern. Es besteht ein Antrieb zwischen zweitem und erstem Glied. Differenziale sorgen dafür, dass sich das dritte zum zweiten verhält wie das zweite zum ersten, das vierte zum dritten wie das dritte zum zweiten usw..

Differenzialbedingungen können sich gleichermaßen auf rotatorische wie lineare Geschwindigkeiten beziehen, auch gemischte Konfigurationen sind problemlos möglich. Anstatt drei können auch vier Körper in Relation gesetzt werden, falls es keinen gemeinsamen Körper gibt. Exemplarisch für verschiedene Differenzialbedingungen lauten die Einträge in den Teil-Jacobimatrizen für das Beispiel des Differenzials zwischen zwei

Abbildung 3.10.: Weitere Anwendungen von Differenzialbedingungen. Links: Ein Greifer mit zwei gleichlaufenden Zangen. Mitte: Das dreigliedrige Teleskopgelenk eines Kranauslegers. Rechts: Die fünfgliedrige Teleskopleiter eines Löschfahrzeugs.

Drehgelenken aus Abbildung 3.9:

$$\begin{aligned}
\mathbf{J}_{\text{diff},r} &= \begin{pmatrix} 0 & 0 & 0 & -d_{r_1} & -d_{r_2} & -d_{r_3} \end{pmatrix} \\
\mathbf{J}_{\text{diff},g} &= \begin{pmatrix} 0 & 0 & 0 & c_d d_{g_1} & c_d d_{g_2} & c_d d_{g_3} \end{pmatrix} \\
\mathbf{J}_{\text{diff},b} &= \begin{pmatrix} 0 & 0 & 0 & (d_{r_1} - c_d d_{g_1}) & (d_{r_2} - c_d d_{g_2}) & (d_{r_3} - c_d d_{g_3}) \end{pmatrix}
\end{aligned} \tag{3.86}$$

Stabilisierung

Je nach Anwendung sind Differenzialbedingung von ihrer Natur her „eher geschwindigkeitsbezogen“ und damit ist keine Stabilisierung notwendig. Betrachtet man zum Beispiel das Differenzialgetriebe beim Radantrieb, spielt ein kleiner Fehler in der Winkelauslenkung der beiden Räder keine große Rolle. Bei den Beispielen Greifer und Teleskopkran jedoch müssen Abweichungen der Winkelstellungen bzw. der jeweiligen linearen Auslenkungen korrigiert werden.

Seien a_1 und a_2 die Auslenkungen am ersten und zweiten Gelenk. Mit der Übersetzungskonstante c_d der Differenzialbedingung ergibt sich der Fehler der Differenzialbedingung e_{diff} dann zu:

$$e_{\text{diff}} = a_1 - c_d \cdot a_2 \tag{3.87}$$

Der Fehlerkorrekturterm stellt sich nach bekanntem Muster dar:

$$\begin{aligned} b_{\text{error,diff}} &= \frac{1}{h} \cdot k_{\text{er}} \cdot e_{\text{diff}} \\ &= \frac{1}{h} \cdot k_{\text{er}} \cdot (a_1 - c_d \cdot a_2) \end{aligned} \tag{3.88}$$

3.4.3. Robuste Realisierung von Feder-Dämpfer-Systemen

Das Modell des Feder-Dämpfer-Systems ist ein grundlegendes Konzept der Mechanik und findet in vielen Bereichen der Dynamiksimulation Anwendung. Offensichtlich sind hier solche Bereiche, in denen explizit reale Feder-Dämpfer-Systeme nachgebildet werden müssen, z.B. zur Simulation einer Radaufhängung bei einer PKW-Simulation. Feder-Dämpfer-Systeme können aber auch implizit auftreten: In nicht starren Kontaktsituationen, z.B. um verformbares Material am Kontaktpunkt nachzubilden oder in nicht ganz starren Verbindungen zwischen Starrkörpern, um Verformungen von Bauteilen durch Belastung nachbilden zu können. Deshalb ist es wichtig, dieses Modell in einem Verfahren für die Mehrkörperdynamiksimulation robust und effizient realisieren zu können.

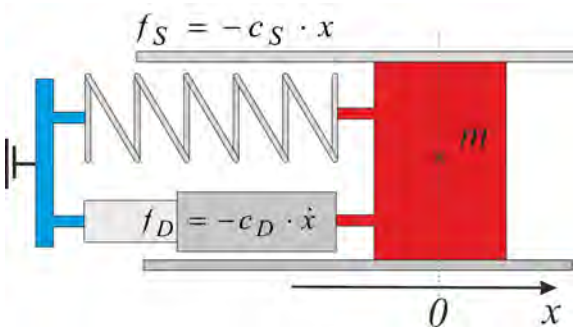


Abbildung 3.11.: 1D-Modell eines parallelen Feder-Dämpfer-Systems

Abbildung 3.11 zeigt das klassische Modell eines parallelen Feder-Dämpfer-Systems. Die Federkraft f_S wirkt proportional zur Auslenkung x und der Federkonstante c_s:

$$f_S = -c_s \cdot x \tag{3.89}$$

Die Dämpfungskraft f_D wirkt proportional zur relativen Geschwindigkeit $v = \dot{x}$ und der

Dämpfungskonstante c_d:

$$f_D = -c_d \cdot v \tag{3.90}$$

Für die Beschleunigung der Masse m gilt demnach die Differenzialgleichung:

$$m \cdot \ddot{x} = -c_s \cdot x - c_d \cdot v \tag{3.91}$$

Der einfachste Weg, beide Kräfte in das einheitliche System aus Gleichung 3.21 einzubinden, verliefe über den Vektor der wirkenden Kräfte $\vec{f}$, der auch die externen Kräfte $\vec{f}_{\text{ext}}$ enthält (vgl. Gleichung 3.9, Seite 78 bzw. Gleichung 3.21, Seite 86). Auslenkung x und Geschwindigkeit v werden zum Zeitpunkt t ausgewertet und die resultierenden Kräfte zur expliziten Integration herangezogen:

$$m \cdot \frac{v(t+h) - v(t)}{h} = -c_s \cdot x(t) - c_d \cdot v(t) \tag{3.92}$$

Dies entspricht dem expliziten Newton-Integrationsschema ersten Grades. Für große Zeitschrittweiten, wie sie in interaktiven Simulationen notwendig sind sowie für große Federkonstanten zur Simulation steifer Federn, ist dieser Ansatz jedoch aufgrund des instabilen Verhaltens ungeeignet.

Um das symplektische oder semi-implizite Integrationsschema anzuwenden, muss die rechte Seite der Gleichung zum Zeitpunkt $t + h$ ausgewertet werden. Mit $x(t+h) \approx x(t) + hv(t+h)$ ergibt sich:

$$\begin{aligned} m \cdot \frac{v(t+h) - v(t)}{h} &= -c_s \cdot x(t+h) - c_d \cdot v(t+h) \\ m \cdot \frac{v(t+h) - v(t)}{h} &= -c_s \cdot (x(t) + v(t+h) \cdot h) - c_d \cdot v(t+h) \end{aligned} \tag{3.93}$$

Umgestellt in eine Vorschrift für die Geschwindigkeit im nächsten Zeitschritt ergibt sich:

$$v(t+h) = -\frac{hc_s x(t) - mv(t)}{h^2 c_s + m + c_d h} \tag{3.94}$$

Um nun eine Analogie zu beliebigen geschwindigkeitsbezogenen Zwangsbedingungen herzustellen, setzen wir folgendes Gedankenspiel an: Anstelle den Massepunkt explizit an Feder und Dämpfer aufzuhängen, soll eine geschwindigkeitsbezogene Zwangs-

bedingung im System dafür sorgen, dass der Massepunkt bei $x = 0$ verbleibt. Die zugehörige geschwindigkeitsbezogene Zwangsbedingung im Sinne von Gleichung 3.16, Seite 80 hat die Form:

$$v(t+h) = 0 \tag{3.95}$$

Sei e der Fehler der positionsbezogenen Zwangsbedingung. Dann hat der Fehlerterm b_{error} für die Stabilisierung (vgl. Kapitel 3.1.5, Seite 87) die Form:

$$b_{\text{error}} = \frac{k_{\text{er}}}{h} e = \frac{k_{\text{er}}}{h}(x_{\text{soll}} - x_{\text{ist}}) = \frac{k_{\text{er}}}{h}(0 - x(t)) = -\frac{k_{\text{er}}}{h} x(t) \tag{3.96}$$

Aus der Open Dynamics Engine [121] stammt die Idee, die Gleichung der Zwangsbedingungen um einen additiven Term zu erweitern, der proportional zur Größe der resultierenden Zwangskraft λ ist. Die Idee hierbei ist es, dem Lösungsalgorithmus bei der Bewertung der Erfüllung einer Zwangsbedingung einen kleinen Spielraum zu überlassen. Die Proportionalität zum Lagrange-Multiplikator bewirkt: Je größer der aufzubringende Zwangsimpuls ist, desto größer wird auch der Spielraum für den Lösungsalgorithmus.

Die Proportionalitätskonstante wird *Constraint Force Mixing* (CFM) genannt. Fügt man diesen Term der geschwindigkeitsbezogenen Zwangsbedingung hinzu, gelangt man zu:

$$v(t+h) + c_{\text{cfm}} \cdot \lambda = -\frac{k_{\text{er}}}{h} x(t) \tag{3.97}$$

Die diskretisierte Newtonsche Bewegungsgleichung nur unter Einfluss der einzigen Zwangskraft λ lautet:

$$\frac{v(t+h) - v(t)}{h} = \frac{\lambda}{m} \tag{3.98}$$

Kombiniert man diese mit der erweiterten Form der Zwangsbedingung 3.97, eliminiert den Multiplikator λ und stellt derart um, dass wiederum eine Vorschrift für die Geschwindigkeit im nächsten Zeitschritt entsteht, gelangt man zu:

$$v(t+h) = -\frac{k_{\text{er}} x(t) - c_{\text{cfm}} m v(t)}{c_{\text{cfm}} m + h} \tag{3.99}$$

Vergleicht man die Gleichungen 3.94 und 3.99 miteinander, so fallen Ähnlichkeiten

auf. Tatsächlich kann man letztere in die erste überführen, setzt man für die beiden Konstanten c_{cfm} und k_{er} die folgenden Ausdrücke ein:

$$c_{\mathrm{cfm}} = \frac{1}{hc_s + c_d}, \quad k_{\mathrm{er}} = \frac{hc_s}{hc_s + c_d} \tag{3.100}$$

Die geschickte Wahl dieser beiden Parameter ermöglicht demnach die Simulation eines Feder-Dämpfer-Systems mit einem stabilen, semi-impliziten Integrationsschema für das angegebene, eindimensionale Beispiel. Tatsächlich lässt sich dieses Prinzip auf das gesamte Mehrkörpersystem erweitern. Die Formulierung der Zwangsbedingungen hat dann die allgemeine Form:

$$\mathbf{J}\vec{v} + \mathbf{C}_{\mathrm{cfm}} \cdot \vec{\lambda} = \vec{b} + \frac{1}{h}\mathbf{K}_{\mathrm{er}}\vec{e} \tag{3.101}$$

$\mathbf{C}_{\mathrm{cfm}}$ und $\mathbf{K}_{\mathrm{er}}$ sind hierin Diagonalmatrizen, die entlang ihrer Hauptdiagonale je einen Wert c_{cfm} bzw. k_{er} für die betreffende Zwangsbedingung enthalten. Der Vektor $\vec{e}$ enthält die positions- bzw. orientierungsbezogenen Fehlerterme. Zur Erinnerung noch einmal die diskretisierten Bewegungsgleichungen des Mehrkörpersystems:

$$\begin{aligned} \frac{\vec{v}(t+h) - \vec{v}(t)}{h} &= \mathbf{M}^{-1}\left(\mathbf{J}^T\vec{\lambda} - \vec{f}\right) \\ \vec{v}(t+h) &= \vec{v}(t) + \mathbf{M}^{-1}\left(\mathbf{J}^T\vec{\lambda}h - \vec{f}h\right) \end{aligned} \tag{3.102}$$

Wendet man nun 3.101 auf 3.102 an, so gelangt man zu folgendem System:

$$\left(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T + \frac{1}{h}\mathbf{C}_{\mathrm{cfm}}\right) h\vec{\lambda} + \mathbf{J}\left(\vec{v}(t) + \mathbf{M}^{-1}h\vec{f}\right) = \vec{b} + \frac{1}{h}\mathbf{K}_{\mathrm{er}}\vec{e} \tag{3.103}$$

Es ist zu erkennen, dass die CFM-Werte einfach auf die betreffenden Elemente der Hauptdiagonale der Systemmatrix aufaddiert werden, um zusammen mit den Fehlerkorrekturparametern k_{er} in einer beliebigen Zwangsbedingung anstatt einer starren Verbindung das Verhalten eines Feder-Dämpfer-Systems unter Anwendung eines semi-impliziten, numerischen Integrationsschemas erster Ordnung zu erzeugen. Dies lässt sich in rotatorischen wie in translatorischen, in bilateralen und in unilateralen Zwangsbedingungen, also z.B. auch in Kontaktpunkten anwenden. Das nächste Unterkapitel behandelt anschaulich die Realisierung einer Radaufhängung als Anwendungsbeispiel eines Feder-Dämpfer-Systems.

3.4.4. Realisierung einer Radaufhängung durch ein integriertes Gelenk

Die Radaufhängung eines PKW enthält eine Vielzahl über Gelenke interagierender Elemente. Soll eine Vorderradaufhängung detailliert ausmodelliert werden, benötigt man z.B. an einer Doppelquerlenkeraufhängung am Vorderrad mindestens drei Dreh- und zwei Kugelgelenke. Liegt der Fokus der Anwendung jedoch nicht primär auf dem Fahrwerk des Fahrzeugs, sind „Speziallösungen“ sinnvoll, die ein visuell plausibles Verhalten bei deutlich reduziertem Aufwand erlauben.

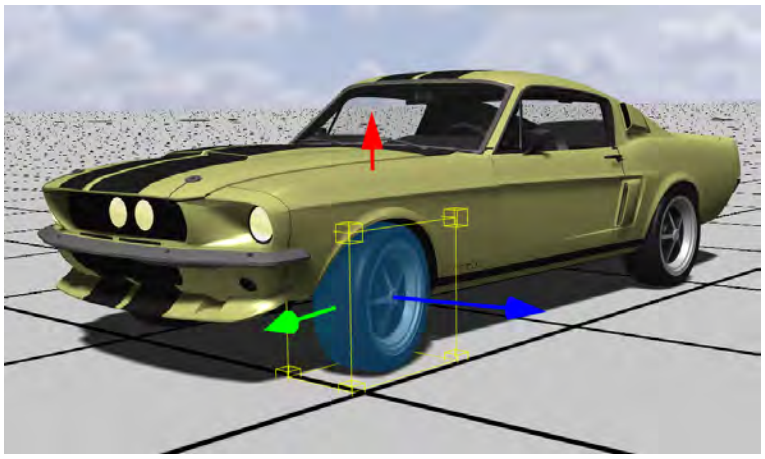


Abbildung 3.12.: Simulation eines PKW-Modells mit laufzeit-optimaler Radaufhängung

Abbildung 3.12 zeigt das Simulationsmodell eines PKW, Vorderrad und dessen Körperkoordinatensystem sind hervorgehoben. Phänomenologisch lassen sich die Beziehungen zwischen Rad und Chassis wie folgt beschreiben:

- Die Lenkachse des Rades entspricht der x-Achse (rot). Die Lenkachse ist fix im Koordinatensystem des Chassis.
- Die Drehachse des Rades entspricht der z-Achse (blau). Die Drehachse ist fix im Koordinatensystem des Rades.
- Federung und Dämpfung verlaufen translatorisch entlang der x-Achse (rot).
- In Richtung der y-Achse (grün) findet weder translatorische noch rotatorische Bewegung statt. Die y-Achse ist sowohl im Koordinatensystem des Rades als auch im Koordinatensystem des Chassis veränderlich. Sie ergibt sich zu jedem Zeitpunkt aus dem Kreuzprodukt von z- und x-Achse.

Das gesamte oben beschriebene Verhalten lässt sich in einem einzigen speziellen Gelenktyp realisieren: Zunächst muss eine Orientierung des Gelenks bestimmt werden. Wie oben erwähnt, lassen sich x- und z-Achse aus ihrer initialen relativen Rotation zu Chassis und Rad bestimmen, die y-Achse ergibt sich als deren Kreuzprodukt ($\vec{y} = \vec{z} \times \vec{x}$). Damit sind alle drei Basisvektoren des Gelenkkoordinatensystems bekannt. Das Vorgehen ist identisch dem beim Kardangelenk. Zwar wird die relative translatorische Bewegung an diesem Gelenk in allen drei Raumrichtungen eingeschränkt, in x-Richtung im Gelenksinn soll aber das Feder-Dämpfer-System realisiert werden. Aus diesem Grund ist es notwendig, die Zwangsbedingungen in den Richtungen der Basisvektoren des Gelenkkoordinatensystems zu formulieren. Wie schon bei den Standardgelenktypen verlaufen die Vektoren $\vec{r}_i, \vec{r}_j$ von den jeweiligen Schwerpunkten von Körper i bzw. j zum Ankerpunkt des Gelenks. Die translatorischen Teil-Jacobimatrizen lauten demnach:

$$\begin{aligned}
\mathbf{J}_{\mathrm{rad,tra},i} &= \begin{pmatrix} -x_1 & -x_2 & -x_3 & (\vec{x}\times\vec{r}_i)_1 & (\vec{x}\times\vec{r}_i)_2 & (\vec{x}\times\vec{r}_i)_3 \\ -y_1 & -y_2 & -y_3 & (\vec{y}\times\vec{r}_i)_1 & (\vec{y}\times\vec{r}_i)_2 & (\vec{y}\times\vec{r}_i)_3 \\ -z_1 & -z_2 & -z_3 & (\vec{z}\times\vec{r}_i)_1 & (\vec{z}\times\vec{r}_i)_2 & (\vec{z}\times\vec{r}_i)_3 \end{pmatrix} \\
\mathbf{J}_{\mathrm{rad,tra},j} &= \begin{pmatrix} x_1 & x_2 & x_3 & -(\vec{x}\times\vec{r}_j)_1 & -(\vec{x}\times\vec{r}_j)_2 & -(\vec{x}\times\vec{r}_j)_3 \\ y_1 & y_2 & y_3 & -(\vec{y}\times\vec{r}_j)_1 & -(\vec{y}\times\vec{r}_j)_2 & -(\vec{y}\times\vec{r}_j)_3 \\ z_1 & z_2 & z_3 & -(\vec{z}\times\vec{r}_j)_1 & -(\vec{z}\times\vec{r}_j)_2 & -(\vec{z}\times\vec{r}_j)_3 \end{pmatrix}
\end{aligned} \tag{3.104}$$

Zur Realisierung des Stoßdämpfers kommt es nun auf die korrekte Anwendung der Konstanten c_{cfm} und k_{er} an. Nur in Richtung der x-Achse des Gelenks soll das Feder-Dämpfer-System wirken, die $\mathbf{C}_{\mathrm{cfm}}$-Matrix zu den oben beschriebenen ersten drei Zwangsbedingungen hat daher die Form:

$$\mathbf{C}_{\mathrm{cfm}} = \begin{pmatrix} c & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad \text{mit} \quad c = \frac{1}{c_s \cdot h + c_d} \tag{3.105}$$

c_s : Federkonstante

c_d : Dämpfungskonstante

Der Fehler in den drei translatorischen Zwangsbedingungen entspricht wie beim Ku-

gelgelenk dem Abstand der beiden Gelenkkoordinatensysteme.

$$\vec{e}_{\mathrm{rad,trans}} = \mathbf{T}_g^{W,K_i} \cdot \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T - \mathbf{T}_g^{W,K_j} \cdot \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T \tag{3.106}$$

Zur Bestimmung des Stabilisierungsterms muss der translatorische Fehler wiederum in das Gelenkkoordinatensystem transformiert werden, weil auch die Zwangsbedingungen im Gelenkkoordinatensystem formuliert worden sind. In folgender Gleichung wird dies durch die Projektion des Fehlers in Weltkoordinaten auf die drei Achsen des Gelenkkoordinatensystems erreicht. Zusätzlich werden unterschiedliche Konstanten k_{er} angesetzt. Für die federnde Achse $k_{\mathrm{er,feder}}$ erfolgt die Wahl entsprechend Feder- und Dämpferkonstante und dem Term 3.100, für die beiden übrigen Achsen wird der Erfahrungswert k_{er} gewählt:

$$\vec{b}_{\mathrm{rad,trans}} = \frac{1}{h} \cdot \begin{pmatrix} k_{\mathrm{er,feder}} \cdot (\vec{x} \cdot \vec{e}_{\mathrm{rad,trans}}) \\ k_{\mathrm{er}} \cdot (\vec{y} \cdot \vec{e}_{\mathrm{rad,trans}}) \\ k_{\mathrm{er}} \cdot (\vec{z} \cdot \vec{e}_{\mathrm{rad,trans}}) \end{pmatrix} \tag{3.107}$$

Im Folgenden wird davon ausgegangen, dass das Vorderrad sowohl gelenkt als auch angetrieben wird. Daher müssen drei rotatorische Freiheitsgrade eingeschränkt werden. Die Teil-Jacobimatrizen sind entsprechend einfach zu formulieren. Als Drehachsen müssen die Basisvektoren des Gelenkkoordinatensystems gewählt werden, da die Unterscheidung nach fixer, angetriebener und gelenkter Achse ebenfalls in diesem Koordinatensystem erfolgt:

$$\begin{aligned} \mathbf{J}_{\mathrm{rad,rot},i} &= \begin{pmatrix} 0 & 0 & 0 & -x_1 & -x_2 & -x_3 \\ 0 & 0 & 0 & -y_1 & -y_2 & -y_3 \\ 0 & 0 & 0 & -z_1 & -z_2 & -z_3 \end{pmatrix} \\ \mathbf{J}_{\mathrm{rad,rot},j} &= \begin{pmatrix} 0 & 0 & 0 & x_1 & x_2 & x_3 \\ 0 & 0 & 0 & y_1 & y_2 & y_3 \\ 0 & 0 & 0 & z_1 & z_2 & z_3 \end{pmatrix} \end{aligned} \tag{3.108}$$

Die rechten Seiten sind für jede der drei Zwangsbedingungen individuell zu bestimmen. Um die y-Achse findet keine Bewegung statt, entsprechend muss hier lediglich der Stabilisierungsterm eingesetzt werden. Ein Fehlermaß für die y-Rotation ergibt sich wie beim Kardan (vgl. Kapitel 3.2.5) aus dem Skalarprodukt von z-Achse in Relation zu Körper j und x-Achse in Relation zu Körper i.

Um die z-Achse soll der Antrieb erfolgen, d.h. hier muss ein Motor realisiert werden, vgl. Kapitel 3.2.8. Die rechte Seite enthält entsprechend den Sollgeschwindigkeitswert $\omega_{\text{antriebsmotorsoll}}$, als Grenzwerte für die Lagrange-Multiplikatoren werden das minimale und maximale Drehmoment $\tau_{\min}, \tau_{\max}$ multipliziert mit der Zeitschrittweite h eingesetzt:

$$h \cdot \tau_{\min} \leq \lambda \leq h \cdot \tau_{\max} \tag{3.109}$$

Um die x-Achse soll das Rad gelenkt werden, d.h. hier muss ein Positionssteller zur Einstellung eines Lenkwinkels realisiert werden. In die rechte Seite ist daher ebenfalls eine Sollgeschwindigkeit einzusetzen. Ihr Wert muss durch einen Regler bestimmt werden, hier soll ein PID-Regler genutzt werden. Seien im Folgenden

α_{soll}	der Sollwinkel für den Lenkausschlag,
$\alpha_{\text{ist}}(t)$	der Istwinkel des Lenkausschlags zum Zeitpunkt t,
p, i, d	die Faktoren für den **P**roportional-, den **I**ntegral- und den **D**ifferenzialanteil des PID-Reglers,
$e(t_0) = \alpha_{\text{soll}} - \alpha_{\text{ist}}(t_0)$	die Regelabweichung zwischen Soll- und Istwert des Lenkausschlags zum Zeitpunkt t_0,
$\frac{\mathrm{d}}{\mathrm{d}t}e(t_0)$	die Ableitung, d.h. die Änderungsrate des Fehlers zum Zeitpunkt $t = t_0$ und
$\int_0^{t_0} e(t)\mathrm{d}t$	das Integral der Fehlerwerte vom Zeitpunkt $t = 0$ bis $t = t_0$,

dann errechnet sich die Sollgeschwindigkeit $\omega_{\text{lenkmotorsoll}}$ und damit die rechte Seite der Zwangsbedingung zu:

$$\omega_{\text{lenkmotorsoll}} = p \cdot e(t_0) + i \cdot \int_0^{t_0} e(t)\mathrm{d}t + d \cdot \frac{\mathrm{d}}{\mathrm{d}t}e(t_0) \tag{3.110}$$

Wie beim Antriebsmotor kann auch das maximale Drehmoment des Lenkmotors durch entsprechende Grenzwerte der Lagrange-Multiplikatoren begrenzt werden.

Für die rechte Seite der drei rotatorischen Zwangsbedingungen ergibt sich damit im

Ganzen:

$$\vec{b}_{\mathrm{rad,rot}} = \begin{pmatrix} p \cdot e(t_0) + i \cdot \int_0^{t_0} e(t)\mathrm{d}t + d \cdot \frac{\mathrm{d}}{\mathrm{d}t} e(t_0) \\ \frac{k_{\mathrm{er}}}{h} \cdot (\vec{z} \cdot \vec{x}) \\ \omega_{\mathrm{antriebsmotorsoll}} \end{pmatrix} \tag{3.111}$$

3.5. Eingesetzte Verfahren zur Lösung des Komplementaritätsproblems

Die einheitliche Formulierung von Bewegungsgleichungen und einem breiten Spektrum von Zwangsbedingungen ist nur der erste Schritt auf dem Weg zur Simulation. Der zweite sind effiziente und robuste Lösungsroutinen für das formulierte mathematische Problem.

Im Rahmen dieser Arbeit wurden vorrangig zwei Lösungsalgorithmen für das Gesamtsystem aus Gleichung 3.21 untersucht: Das projizierte Gauß-Seidel Iterationsverfahren [43] und eine Variante der sogenannten Dantzig Routine [38]. Die folgenden Kapitel beschreiben beide Verfahren im Detail und vergleichen ihre Vor- und Nachteile anhand eines praxisnahen Versuchsaufbaus in Form der Simulation eines repräsentativen Referenzmodells.

Zur Erinnerung noch einmal das gegebene Problem in allgemeiner Form:

$$\mathbf{A}\vec{\lambda} = \vec{b} + \vec{w}, \quad \vec{\lambda}_{\mathrm{low}} \leq \vec{\lambda} \leq \vec{\lambda}_{\mathrm{high}}$$

Gesucht werden Tupel (λ_i, w_i), für die gilt:

$$(\lambda_i, w_i) : \begin{cases} \lambda_i = \lambda_{\mathrm{low},i}, & \rightarrow w_i > 0 \\ \lambda_i = \lambda_{\mathrm{high},i}, & \rightarrow w_i < 0 \\ \lambda_{\mathrm{low},i} < \lambda_i < \lambda_{\mathrm{high},i}, & \rightarrow w_i = 0 \end{cases} \tag{3.112}$$

Mit Bezug auf Gleichung 3.21, Seite 86 gelten diese Entsprechungen:

$$\begin{aligned}
\mathbf{A} &= \begin{pmatrix} \mathbf{J}_{nf}\mathbf{M}^{-1}\mathbf{J}_{nf}^{T} & \mathbf{J}_{nf}\mathbf{M}^{-1}\mathbf{J}_{f}^{T} \\ \mathbf{J}_{f}\mathbf{M}^{-1}\mathbf{J}_{nf}^{T} & \mathbf{J}_{f}\mathbf{M}^{-1}\mathbf{J}_{f}^{T} \end{pmatrix} \\
\vec{b} &= -\begin{pmatrix} \mathbf{J}_{nf} \\ \mathbf{J}_{f} \end{pmatrix} \left[\vec{v} + \mathbf{M}^{-1} h\vec{f} \right] + \begin{pmatrix} \vec{b}_{nf} \\ \vec{b}_{f} \end{pmatrix} \\
\vec{\lambda} &= \begin{pmatrix} h\vec{\lambda}_{nf} \\ h\vec{\lambda}_{f} \end{pmatrix}
\end{aligned} \tag{3.113}$$

3.5.1. Das projizierte Gauß-Seidel Iterationsverfahren

Das Gauß-Seidel (GS) Iterationsverfahren dient ursprünglich der Lösung linearer Gleichungssysteme ohne Nebenbedingungen. Mit leichten Modifikationen kann es jedoch auch zur Lösung des in Gleichung 3.21 formulierten vereinfachten linearen Komplementaritätsproblems genutzt werden. Es ist einfach zu verstehen und zu implementieren und liefert für eine Vielzahl Simulationen ausreichend schnell ausreichend genaue Lösungen. Vor allem Erleben [43, 44] belegt dies mit der Simulationen großer Stapelmodelle, für die er die Gauß-Seidel Routine zusammen mit seiner Implementierung der Stoßfortpflanzung (engl.: *Shock Propagation*) einsetzt.

Zur Herleitung des Verfahrens geht man zunächst von einem linearen Gleichungssystem (LGS) aus:

$$\mathbf{A}\vec{\lambda} = \vec{b} \tag{3.114}$$

Die Matrix $\mathbf{A}$ wird zerlegt in eine rechte obere Dreiecksmatrix $\mathbf{R}$ und eine linke untere Dreiecksmatrix $\mathbf{L}$, die die Elemente der Hauptdiagonalen enthält:

$$\begin{aligned}
(\mathbf{L} + \mathbf{R})\,\vec{\lambda} &= \vec{b} \\
\mathbf{L}\vec{\lambda} &= \vec{b} - \mathbf{R}\vec{\lambda}
\end{aligned} \tag{3.115}$$

Hieraus lässt sich eine Iterationsvorschrift für den Vektor $\vec{\lambda}$ ableiten:

$$\vec{\lambda}^{k+1} = \mathbf{L}^{-1}\left(\vec{b} - \mathbf{R}\vec{\lambda}^{k}\right) \tag{3.116}$$

Unter Ausnutzung der besonderen Dreiecksform der Matrix $\mathbf{L}$ lässt sich hieraus eine

iterative, elementweise Berechnungsvorschrift für die Elemente des Vektors $\vec{\lambda}$ ableiten:

$$\lambda_m^{k+1} = \frac{1}{a_{m,m}} \left(b_m - \sum_{i=0}^{m-1} a_{k,i} \cdot \lambda_i^{k+1} - \sum_{i=m+1}^{n} a_{k,i} \cdot \lambda_i^k \right) \tag{3.117}$$

Algorithmus 1 beschreibt eine direkte Implementierung des Gauß-Seidel Iterationsverfahrens in Pseudocode.

Algorithmus 1: Gauß-Seidel Iterationsverfahren zur Lösung eines linearen Gleichungssystems ohne Nebenbedingungen

```
for index = 0 ... numberIterations -1 do
    for zeile = 0 ... n -1 do
        summe ← 0;
        for spalte = 0 ... (zeile − 1) do
            summe ← summe +A[zeile][spalte] · λ[spalte];
        end
        for spalte = (zeile + 1) ... (n − 1) do
            summe ← summe + A[zeile][spalte] · λ[spalte];
        end
        λ[zeile] ← (b[zeile] − summe)/A[zeile][zeile]
    end
end
```

Verglichen mit dem linearen Gleichungssystem ohne Nebenbedingungen aus Gleichung 3.114 enthält die vereinfachte Komplementaritätsformulierung nur geringfügige Erweiterungen. Die rechte Seite wird additiv um die Schlupfvariablen im Vektor $\vec{w}$ erweitert, im Gegenzug wird der gesuchte Vektor $\vec{\lambda}$ mit oberen und unteren Grenzwerten eingeschränkt. Dabei ist der Wert des Vektors $\vec{w}$ für das Ergebnis nicht von Bedeutung. So kann das Gauß-Seidel Verfahren zur Lösung eines Gleichungssystems ohne Nebenbedingungen durch einen einfachen Projektionsschritt des Lösungsvektors $\vec{\lambda}$ am Ende einer Iteration zur Lösung des Komplementaritätsproblems erweitert werden. Die resultierende Routine in Pseudo-Code ist in Algorithmus 2 dargestellt.

Berücksichtigung der Reibungs-Grenzwerte Wie in Kapitel 3.1.4 beschrieben, verzichtet die hier eingesetzte Formulierung auf zusätzliche Zwangsbedingungen, die die Beträge der Reibungszwangsimpulse in Bezug zu den zugehörigen Kontaktnormalenzwangsimpulsen einschränken. In der vereinfachten Formulierung muss diese Beziehung durch den Lösungsalgorithmus hergestellt werden.

Algorithmus 2: Projiziertes Gauß-Seidel Iterationsverfahren zur Lösung des vereinfachten Komplementaritätsproblems

```
for index = 0 ... numberIterations − 1 do
    for zeile = 0 ... (n − 1) do
        summe ← 0;
        for spalte = 0 ... (zeile − 1) do
            summe ← summe + A[zeile][spalte] · λ[spalte];
        end
        for spalte = (zeile + 1) ... (n − 1) do
            summe ← summe + A[zeile][spalte] · λ[spalte];
        end
        λ[zeile] ← (b[zeile] − summe)/A[zeile][zeile];
    end
    // Der Projektionsschritt ;
    for zeile = 0 ... n − 1 do
        λ[zeile] ← min((max(λ_low[zeile], λ[zeile]), λ_high[zeile]) ;
    end
end
```

In der Gauß-Seidel Routine geschieht dies durch wiederholte Anpassung der Grenzwerte. Immer wenn der Algorithmus eine Zeile bearbeitet, die eine Reibungszwangsbedingung repräsentiert, werden die Grenzwerte vor dem Projektionsschritt aktualisiert. Deshalb wird auch hier angenommen, dass die Reibungszwangsbedingungen immer *hinter* den Kontaktnormalenzwangsbedingungen stehen, ihre Zeilennummern also immer größer als die der entsprechenden Normalenbedingungen sind. Die angepasste Routine ist in Algorithmus 3 dargestellt. Die Methode `getNormalIndex(k)` liefert den Index der Kontaktnormalenbedingung zur Reibungszwangsbedingung k.

3.5.2. Eine effiziente und problemorientierte Implementierung der Dantzig Routine

Die sogenannte Dantzig Routine [38] dient der Lösung klassischer Linearer Komplementaritätsprobleme (LCPs). Baraff [12] stellt eine Implementierung für die konkrete Anwendung in Mehrkörpersystemen vor. Diese Implementierung ist die Grundlage des hier beschriebenen Verfahrens, ein eng verwandtes Verfahren ist auch in der Open Dynamics Engine [121] implementiert.

Seien $1..j$ die Indizes aller Zwangsbedingungen des Systems. Entsprechend repräsentiert

Algorithmus 3: Projiziertes Gauß-Seidel Iterationsverfahren mit Behandlung der Reibungszwangsbedingungen

for index $= 0 \ldots$ numberIterations $- 1$ **do**
  for zeile $= 0 \ldots (n-1)$ **do**
    summe $\leftarrow 0$;
    for spalte $= 0 \ldots ($zeile $- 1)$ **do**
      summesumme $+ \mathbf{A}$[zeile][spalte] $\cdot \lambda$[spalte];
    end
    for spalte $= ($zeile $+ 1) \ldots (n-1)$ **do**
      summe $\leftarrow$ summe $+ \mathbf{A}$[zeile][spalte] $\cdot \lambda$[spalte];
    end
    λ[zeile] $\leftarrow (b$[zeile] $-$ summe$)/\mathbf{A}$[zeile][zeile];
  end
  // Anpassung der Grenzwerte der Reibungsimpulse ;
  if *isFrictionConstraint*(zeile) **then**
    λ_{low}[zeile] $\leftarrow -\mu \cdot \lambda$[`getNormalIndex`(zeile)] ;
    λ_{high}[zeile] $\leftarrow \mu \cdot \lambda$[`getNormalIndex`(zeile)] ;
  end
  // Der Projektionsschritt ;
  for zeile $= 0 \ldots (n-1)$ **do**
    λ[zeile] $\leftarrow \min\left(\left(\max(\lambda_{\text{low}}[\text{zeile}], \lambda[\text{zeile}])\right., \lambda_{\text{high}}[\text{zeile}]\right)$;
  end
end

sentieren die Elemente des Vektors $\vec{\lambda}_{1..j}$ die Zwangsimpulse zu den einzelnen Zwangsbedingungen. Die Elemente des Vektors $\vec{w}_{1..j}$ sind die Schlupfvariablen der einzelnen Zwangsbedingungen und beschreiben damit die Abweichungen zwischen Soll- und Istwert einer Zwangsbedingung.

Da der Algorithmus das Gesamtsystem durch schrittweise Hinzunahme weiterer Zwangsbedingungen löst, werden im weiteren Verlauf Teile des Gesamtsystems folgendermaßen identifiziert: So ist $\mathbf{A}_{1..i,1..j}$ eine Submatrix von $\mathbf{A}$, die nur die Zeilen $1..i$ und die Spalten $1..j$ enthält. Ebenso seien $\vec{\lambda}_{1..i}$ bzw. $\vec{w}_{1..i}$ entsprechende Teil-Vektoren.

Der Algorithmus ist - verglichen mit der Gauß-Seidel Routine aus dem vorherigen Kapitel - relativ komplex und eine laufzeiteffiziente Implementierung bedarf vieler Optimierungen. Es folgt daher zunächst ein Grobüberblick in sechs Schritten, danach werden die wichtigsten Teilaufgaben in Form von „Prozeduren“ im Detail beschrieben. Am Ende erfolgt eine detaillierte Beschreibung der Hauptroutine in Pseudocode-Form.

Der grundlegende Ablauf der Dantzig Routine ist wie folgt:

1. Ausgangspunkt ist eine gültige Lösung eines Teils des Gesamtproblems, d.h. für $\mathbf{A}_{1..j,1..j}$ und $\vec{b}_{1..j}$ ist $< \vec{\lambda}_{1..j}, \vec{w}_{1..j} >$ eine gültige Lösung unter Einhaltung der zugehörigen Komplementaritätsbedingungen. Für $j = 1$ ist diese Lösung trivial und direkt bestimmbar.

   Die bereits im System enthaltenen Zwangsbedingungen $1..j$ sind einer von zwei Mengen zugeordnet: Entweder der zugehörige Lagrange-Multiplikator hat den Wert einer seiner Grenzwerte angenommen und der Fehlerwert w ist ungleich Null. Am Beispiel der klassischen Kontaktbedingung bedeutet das: Der Kontaktimpuls ist Null ($\lambda_i = 0$) und die in Kontakt stehenden Körper bewegen sich auseinander ($w_i > 0$). Solche Zwangsbedingungen werden der Menge B (**B**egrenzt) zugeordnet. Oder aber der zugehörige Lagrange-Multiplikator liegt echt innerhalb seiner Grenzwerte ($\lambda_{\mathrm{low},i} < \lambda_i < \lambda_{\mathrm{high},i}$) und der Fehlerwert ist gleich Null ($w_i = 0$). Diese Zwangsbedingungen werden der Menge N (**N**icht begrenzt) zugeordnet. Um die Darstellung des Algorithmus im Folgenden zu vereinfachen, wird angenommen, die ersten n Bedingungen seien nicht begrenzt, also in N und die nächsten b Bedingungen seien begrenzt, also in B. Formal bedeutet dies:

   $$N = \{1 \ldots (n)\}, B = \{n + 1 \ldots (n + b)\} \tag{3.118}$$

2. Dem Teilproblem wird eine der noch nicht berücksichtigten Zwangsbedingungen

hinzugefügt, indem die Matrix um eine Zeile und eine Spalte ($\mathbf{A}_{1..(j+1),1..(j+1)}$) und die rechte Seite um ein Element ergänzt werden ($\vec{b}_{1..(j+1)}$). Der zugehörige Lagrange-Multiplikator λ_{j+1} erhält initial den Wert Null. Hiermit kann nun der Fehlerwert der neuen Bedingung w_{j+1} berechnet werden. Die Details beschreibt die Prozedur ***computeError***.

3.
   - Falls die neu hinzugekommene Bedingung $j+1$ mit dem Initialwert Null für λ_{j+1} erfüllt ist, fahre fort mit Schritt 2.
   - Sonst: Suche eine Richtung $\Delta\vec{\lambda}$, in die $\vec{\lambda}_{1..j+1}$ „geschoben“ werden kann, so dass der Fehler der neuen Zwangsbedingung w_{j+1} reduziert wird und gleichzeitig alle bisher enthaltenen Zwangsbedingungen $1..j$ Gültigkeit behalten. Die Details beschreibt die Prozedur ***findDirection***.

4. Suche eine maximale Schrittweite s, um die $\vec{\lambda}_{1..j+1}$ in Richtung $\Delta\vec{\lambda}$ verschoben werden kann, so dass für alle Bedingungen $k \in \{1..(j+1)\}$ des Systems gilt:

   a) Keines der Elemente $\lambda_i, i \in N \cup j+1$ unter- oder überschreitet seine Grenzwerte, d.h. es darf nicht kleiner als $\lambda_{\mathrm{low},i}$ oder größer als $\lambda_{\mathrm{high},i}$ werden.

   b) Keines der Elemente $w_i, i \in B \cup j+1$ ändert sein Vorzeichen: Der zugehörige Lagrange-Multiplikator λ_i ist von einem seiner Grenzwerte in Richtung seines gültigen Bereiches geschoben worden.

   Die Bestimmung der maximalen Schrittweite s erfüllt die Prozedur ***findStepSize***. Sie liefert außerdem den Index derjenigen Zwangsbedingung zurück, die die maximale Schrittweite am stärksten einschränkt.

5.
   - Falls die gefundene Schrittweite kleiner Null ist, kann der Algorithmus keine Lösung bestimmen. In diesem Fall wird nach einer kleinen numerischen Modifikation der Systemmatrix ein neuer Lösungsversuch gestartet. Siehe dazu Abschnitt „Terminierung des Algorithmus“.
   - Sonst: Schiebe den Lösungsvektor um die gefundene Schrittweite in Richtung des Vektors $\Delta\vec{\lambda}$: $\vec{\lambda} = \vec{\lambda} + s \cdot \Delta\vec{\lambda}$.

6.
   - Falls die die Schrittweite s einschränkende Bedingung die zuletzt hinzugefügte $j+1$ war, ist eine Lösung für das vergrößerte System gefunden.
     - Enthält das System weitere Zwangsbedingungen, wird mit Schritt 2 fortgefahren.

  - Sind alle Zwangsbedingungen durchlaufen, ist eine Lösung für das Gesamtsystem gefunden und der Algorithmus terminiert.
- Sonst: Wiederhole die Schritte 4 bis 6 so lange, bis die neu hinzugefügte Zwangsbedingung diejenige war, welche die maximale Schrittweite begrenzte.

Prozedur: *computeError* Für den Lagrange-Multiplikator der neuen Bedingung $j+1$ wird zunächst ein Initialwert von Null angenommen ($\lambda_{j+1} = 0$). Damit kann der Fehlerwert der neuen Zwangsbedingung w_{j+1} einfach bestimmt werden: $w_{j+1} = \mathbf{A}_{j+1,1..(j+1)} \cdot \vec{\lambda}_{j+1} - \vec{b}_{j+1}$. Für ihn werden nun vier Fälle unterschieden:

1. $w_{j+1} = 0$ und $\lambda_{\mathrm{low},j+1} \leq 0$ und $\lambda_{\mathrm{high},j+1} \geq 0$:
   In diesem Fall ist die Bedingung unter Einhaltung ihrer Grenzwerte erfüllt. $j+1$ wird der Menge N hinzugefügt.

2. $w_{j+1} > 0$ und $\lambda_{\mathrm{low},j+1} = 0$:
   Es tritt zwar eine Relativgeschwindigkeit ungleich Null in positiver Richtung auf, die Zwangskraft für diese Bedingung darf jedoch nicht kleiner Null werden, um dies zu korrigieren. Der Fall entspricht der klassischen Kontaktnormalenbedingung: Die in Kontakt stehenden Körper driften auseinander, der Kontakt nimmt also keinen Einfluss. Die Bedingung ist erfüllt. Der Index $j+1$ wird der Menge B hinzugefügt.

3. $w_{j+1} < 0$ und $\lambda_{\mathrm{high},j+1} = 0$:
   Wie der vorherige Fall, nur mit je umgedrehten Vorzeichen. Das Verhalten des Algorithmus ist identisch mit dem im vorherigen Fall. Die Bedingung ist erfüllt.

4. $w_{j+1} \neq 0$ und $\lambda_{\mathrm{low},j+1} < 0$ und $\lambda_{\mathrm{high},j+1} > 0$
   Dies ist der interessanteste Fall. Die neue Bedingung $j+1$ ist nicht erfüllt: Es tritt eine Relativgeschwindigkeit auf, der entsprechende Grenzwert für λ_{j+1} ist aber noch nicht erreicht. Nur in diesem Fall erreicht die Routine Schritt 4 in der Gesamtbeschreibung.

Prozedur: *findDirection* Es wird weiterhin angenommen, dass die ersten n Zwangsbedingungen diejenigen sind, die innerhalb ihrer Grenzwerte liegen und die nächsten b diejenigen, deren Lagrange-Multiplikatoren einen ihrer jeweiligen Grenzwerte angenommen haben. Mit diesen Annahmen kann man die Systemmatrix $\mathbf{A}_{1..n+b+1}$ wie folgt

aufteilen:

$$\mathbf{A}_{1..n+b+1,1..n+b+1} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \vec{v}_1 \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} & \vec{v}_2 \\ \vec{v}_1^T & \vec{v}_2^T & a \end{pmatrix} \tag{3.119}$$

Hierin sind $\mathbf{A}_{11} \in \mathbb{R}^{n \times n}$, $\mathbf{A}_{12} \in \mathbb{R}^{n \times b}$ und $\mathbf{A}_{22} \in \mathbb{R}^{b \times b}$.
Der Vektor $\Delta\vec{\lambda}$ muss die folgende Form annehmen:

$$\Delta\vec{\lambda} \begin{pmatrix} \vec{x} \\ \vec{0} \\ e \end{pmatrix} \text{ mit } e = \begin{cases} 1, \text{ falls } w_{b+1} < 0 \\ -1, \text{ falls } w_{b+1} > 0 \end{cases} \tag{3.120}$$

Erläuterung: Die ersten n Lagrange-Multiplikatoren haben ihre Grenzwerte noch nicht erreicht (oder sind unbeschränkt). Sie können also variieren, um die zugehörigen Fehlerwerte w_i auf Null zu halten. Die Lagrange-Multiplikatoren der folgenden b Zwangsbedingungen haben ihre Grenzwerte bereits erreicht und weisen einen Fehler ungleich Null auf ($w_i \neq 0$). Die zugehörigen Lagrange-Multiplikatoren werden daher nicht verändert - Veränderungen im Gesamtsystem würden zunächst zu Veränderungen am Fehlerwert w_i führen. Die Veränderungen in den Fehlerwerten der Zwangsbedingungen Δw ergeben sich durch Anwendung der Systemmatrix auf den Vektor $\Delta\vec{\lambda}$:

$$\Delta\vec{w} = \mathbf{A}\Delta\vec{\lambda} = \mathbf{A} \begin{pmatrix} \vec{x} \\ \vec{0} \\ e \end{pmatrix} = \begin{pmatrix} \mathbf{A}_{11}\vec{x} + \vec{v}_1 \\ \mathbf{A}_{12}^T\vec{x} + \vec{v}_2 \\ \vec{v}_1^T\vec{x} + a \end{pmatrix} \tag{3.121}$$

Da die ersten n Elemente des Vektors $\Delta\vec{w}$ Null sein müssen (die Zwangsimpulse haben ihre Grenzwerte noch nicht erreicht und die w-Werte sind Null und sollen es auch bleiben), kann man dies umformen zu:

$$\mathbf{A}_{11}\vec{x} = -\vec{v}_1 \tag{3.122}$$

Löst man dies nach $\vec{x}$, sind alle Elemente von $\Delta\vec{\lambda}$ bestimmt.

Die Lösung von $\mathbf{A}_{11}\vec{x} = -\vec{v}_1$ ist eine der Schlüsselstellen im Algorithmus, will man eine effiziente Implementierung erreichen. Eine Möglichkeit hierzu ist der Einsatz einer inkrementellen $\mathbf{LDL}^T$-Zerlegung. Eine vollständige Lösung des LGS besitzt eine Laufzeitkomplexität von $O(n^3)$. Tatsächlich unterscheidet sich das LGS bei jedem Aufruf des

Lösers jedoch nur geringfügig von der Form während eines früheren Aufrufs: Ein bestehendes System, für das bereits eine Lösung bestimmt wurde, wird lediglich um eine Zeile und eine Spalte ergänzt. Dies geschieht immer dann, wenn ein Index der Menge N hinzugefügt wird. Diese Tatsache kann man sich zu Nutze machen:

Sei die $\mathbf{LDL}^T$-Zerlegung (vgl. Anhang A.4) der symmetrischen Matrix $\mathbf{A}$ gegeben durch:

$$\mathbf{A} = \mathbf{L} \cdot \mathbf{D} \cdot \mathbf{L}^T \tag{3.123}$$

Seien $\mathbf{a}, \mathbf{b}$ neue Zeilen und Spalten, die dem System hinzugefügt werden sollen, dann hat die neue Zerlegung die Form:

$$\begin{aligned} \begin{pmatrix} \mathbf{A} & a \\ a^T & b \end{pmatrix} &= \begin{pmatrix} \mathbf{L} & \\ \mathbf{l}^T & \mathbf{e} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{D} & \\ & \mathbf{d} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{L}^T & \mathbf{l} \\ & \mathbf{e}^T \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{LDL}^T & \mathbf{LDl} \\ \mathbf{l}^T\mathbf{DL}^T & \mathbf{l}^T\mathbf{Dl} + \mathbf{ede}^T \end{pmatrix} \end{aligned} \tag{3.124}$$

Zur Bestimmung der vollständigen Zerlegung des vergrößerten Systems geht man wie folgt vor:

$$\begin{aligned} \mathbf{LDl} &= \mathbf{a} \\ \mathbf{Dl} &= \mathbf{L}^{-1}\mathbf{a} \\ \mathbf{l} &= \mathbf{D}^{-1} \cdot \mathbf{Dl} = \mathbf{D}^{-1} \cdot \mathbf{L}^{-1}\mathbf{a} \\ &\text{und} \\ \mathbf{l}^T\mathbf{Dl} + \mathbf{ede}^T &= \mathbf{b} \\ \mathbf{ede}^T &= \mathbf{b} - \mathbf{l}^T\mathbf{Dl} \end{aligned} \tag{3.125}$$

Die Hauptdiagonalelemente von $\mathbf{L}$ sind 1, i.A. ist $\mathbf{e}$ also die Einheitsmatrix. Damit sind $\mathbf{l}$ und $\mathbf{d}$ bestimmt und die Zerlegung der erweiterten Systemmatrix ist vollständig.

Zur Komplexität: Sei die Matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, es werden eine Zeile und eine Spalte hinzugefügt, die Elemente a, b sind also Skalare, $e = 1$. Für die Bestimmung der einzelnen Terme sind Berechnungen mit den folgenden Komplexitäten (in Landau-Notation) notwendig:

Term	Komplexität
$\mathbf{Dl}$	$O(\frac{n^2}{2})$
$\mathbf{l}$	$O(n)$
$\mathbf{d}$	$O(n)$

Aus der neuen $\mathbf{LDL}^T$-Zerlegung lässt sich mit noch einmal der Komplexität $O(n^2)$ die neue Lösung bestimmen. Im Ganzen kann also eine Lösung des um eine Zeile und eine Spalte erweiterten Systems mit quadratischer Laufzeitkomplexität bestimmt werden. Verglichen mit einer erneuten Gesamtlösung bietet dieses Vorgehen also ein um eine Potenz besseres Laufzeitverhalten und leistet so einen wichtigen Beitrag, um die Dantzig Routine laufzeiteffizient implementieren zu können. In der unten beschriebenen Routine in Pseudocode wird das Hinzufügen einer Zeile und einer Spalte zum System immer dort durchgeführt, wo ein Index der Menge N hinzugefügt wird.

Prozedur: *findStepSize* Bei der Korrektur des Fehlers in Bedingung $j+1$ durch Schieben des Vektors $\vec{\lambda}$ in Richtung $\Delta\vec{\lambda}$ darf die Schrittweite s nur so groß sein, dass alle im System enthaltenen Bedingungen erfüllt bleiben. Um dies sicherzustellen, läuft der Algorithmus über alle Zwangsbedingungen $k \in \mathbf{N} \cup \mathbf{B} \cup d$ und sucht nach der größtmöglichen Schrittweite s, bevor eine der folgenden Bedingungen verletzt wird.

- Falls λ_k echt innerhalb seiner Grenzwerte liegt, d.h. $\lambda_{\mathrm{low},k} < \lambda_k < \lambda_{\mathrm{high},k}$, überschreitet und unterschreitet auch $\lambda_k + s\Delta\lambda_k$ nicht seine Grenzwerte: $\lambda_{\mathrm{low},k} \leq \lambda_k + s\Delta\lambda_k$ und $\lambda_k + s\Delta\lambda_k \leq \lambda_{\mathrm{high},k}$). Für die Schrittweite gilt daher: $s \leq (\lambda_{\mathrm{high},i} - \lambda_i)/\Delta\lambda_i$ bzw. $s \leq (\lambda_{\mathrm{low},i} - \lambda_i)/\Delta\lambda_i$. Am Beispiel der Kontaktnormalenbedingung: Ein Kontaktnormalenimpuls, der bisher „auseinanderdrückend“ wirkt, darf beliebig weiter ansteigen ($\lambda_{\mathrm{high},k} = \infty$), jedoch nicht unter Null fallen ($\lambda_{\mathrm{low},k} = 0$), weil ein Kontakt nicht „ziehend“ wirken darf.

- Sonst: Der Fehlerterm w_k ändert nicht sein Vorzeichen, d.h. $w_k > 0 \rightarrow w_k + s\Delta w_k \geq 0$ und $w_k < 0 \rightarrow w_k + s\Delta w_k \leq 0$. Für die Schrittweite gilt: $s \leq -w_i/\Delta w_i$. Für das Kontaktbeispiel heißt dies: Driften die Körper im Kontakt auseinander, darf ihre relative Geschwindigkeit beliebig ansteigen, jedoch nicht unter Null fallen, da dies eine Durchdringung zur Folge hätte.

Spezielle Behandlung unbeschränkter Zwangsbedingungen

Für die Grenzwerte der Lagrange-Multiplikatoren unbeschränkter Zwangsbedingungen gilt: $\lambda_{\text{low},i} = -\infty, \lambda_{\text{high},i} = \infty$. Ihre Fehlerterme w_i müssen gleich Null bleiben. Die Gesamtmenge der unbeschränkten Zwangsbedingungen formt daher ein lineares Gleichungssystem, kein LCP. Sie können auf besonderem Wege in der Dantzig Routine berücksichtigt werden.

Die u nicht beschränkten Zwangsbedingungen werden als erste in der Jacobimatrix aufgenommen und werden entsprechend durch die Submatrix $\mathbf{A}_{1..u,1..u}$ und die ersten u Elemente des Vektors $\vec{\lambda}$ repräsentiert. Zu Beginn des Verfahrens wird zunächst durch eine initiale $\mathbf{LDL}^T$-Zerlegung der Teilmatrix $\mathbf{A}_{1..u,1..u}$ eine Lösung für dieses Subsystem bestimmt. Alle Indizes $1..u$ werden zu Beginn in die Menge N aufgenommen und der Algorithmus läuft genauso ab, wie oben beschrieben. Der Algorithmus stellt sicher, dass die Gleichungen der unbeschränkten Zwangsbedingungen während des Hinzufügens von Komplementaritätsbedingungen ihre Gültigkeit behalten.

Berücksichtigung der Reibungs-Grenzwerte

Genau wie beim Gauß-Seidel Iterationsverfahren müssen auch bei Einsatz der Dantzig Routine die Grenzwerte für die Reibungsimpulse in Abhängigkeit von den zugehörigen Kontaktnormalenimpulsen skaliert werden, da die hier eingesetzte Formulierung keine entsprechenden Hilfs-Zwangsbedingungen wie in den klassischen Formulierungen (siehe z.B. Gleichung 2.37, Seite 59) einsetzt.

Das Gauß-Seidel Verfahren läuft in jeder Iteration erneut über alle Zwangsbedingungen, d.h. die Grenzwerte der Reibungskräfte können in jeder Iteration erneut angepasst werden. Die Dantzig Routine hingegen fügt nach und nach alle Zwangsbedingungen dem Gesamtsystem hinzu - sind einmal alle Zwangsbedingungen enthalten, terminiert das Verfahren. Entsprechend können die Reibungsimpuls-Grenzwerte nur einmalig bestimmt werden. Um dennoch eine möglichst gute Annäherung an die exakte Lösung zu erhalten, werden die Reibungszwangsbedingungen als letzte in die Jacobimatrix eingefügt, so dass sie auch erst als letzte von der Dantzig Routine behandelt werden. Bevor der Algorithmus die erste Reibungszwangsbedingung behandelt, werden ihre Grenzwerte also in Abhängigkeit von den zugehörigen Kontaktnormalenimpulsen bestimmt. In der Pseudocode-Formulierung wird dies durch die Prozedur *compileFrictionLimits()* repräsentiert. Diese Prozedur bestimmt schlicht die Maximalwerte der Reibungsimpulse

durch Multiplikation von Kontaktnormalenimpulsen und Reibungskoeffizienten.

Algorithmus 4 beschreibt die gesamte Hauptroutine des Dantzig Verfahrens in Pseudocode.

Terminierung des Algorithmus

Es treten Konfigurationen auf, in denen oben beschriebener Algorithmus keine Lösung findet. Dies äußert sich in einer maximalen Schrittweite $s \leq 0$, die die Prozedur findStepSize() liefert. Eine numerische Modifikation der Systemmatrix hilft, dieses Problem zu umgehen: Wie bei einem linearen Gleichungssystem kann die Verstärkung bzw. Herstellung einer Diagonaldominanz der Systemmatrix $\mathbf{A}$ die Chance steigern, dass der Algorithmus eine Lösung findet. Unter Diagonaldominanz versteht man die Eigenschaft, dass das Hauptdiagonalelement jeder Zeile größer als die Summe aller anderen Elemente derselben Zeile ist. Mit diesem Wissen wird die Routine wie folgt ergänzt: Kann der Algorithmus für ein gegebenes System keine Lösung finden, wird die diagonale Dominanz erhöht, indem ein konstanter Aufschlag auf die Hauptdiagonalelemente aufaddiert wird. Kann der Algorithmus immer noch nicht lösen, wird der konstante Aufschlag verdoppelt und die Lösung erneut versucht. Dies wird solange wiederholt, bis eine Lösung gefunden werden kann oder der Aufschlag so groß ist, dass die Verfälschung der Lösung so groß ist, dass diese nicht mehr sinnvoll wäre. Tritt dieser Fall tatsächlich einmal ein, wird für $\vec{\lambda}$ der Nullvektor eingesetzt und damit ein „ballistischer“ Integrationsschritt ohne Einfluss von Zwangskräften ausgeführt. Mit diesem Verfahren konnte die Simulation in allen untersuchten Anwendungen und allen bis heute erreichten Konfigurationen erfolgreich fortgesetzt werden.

Als Startwert für den additiven Aufschlag für alle realisierten Anwendungen hat sich $1 \cdot 10^{-4}$ als guter Erfahrungswert herauskristallisiert. Wird $1 \cdot 10^{-1}$ überschritten, bricht die Routine ab und fährt mit der „ballistischen Lösung“ $\vec{\lambda} = \vec{0}$ fort.

3.5.3. Vergleich und Bewertung der eingesetzten Lösungsverfahren

Drei wesentliche Merkmale dienen dem Vergleich und der Bewertung der unterschiedlichen vorgestellten Lösungsverfahren. Genauigkeit, Geschwindigkeit und Stabilität. Diese Arbeit fokussiert konkrete, produktive Anwendungsszenarien der Verfahren. Um dieser Anforderung gerecht zu werden, wird auf abstrakte Referenzmodelle, wie hohe Tür-

Algorithmus 4: Hauptroutine des Dantzig Verfahrens in Pseudo-Code

```
B ← {};
N ← {};
u ← numberUnlimitedConstraints() ;
if u ≥ 0 then
    N = {1..u};
    λ⃗_N ← lgsSolve(A_{N,N}, b⃗_N) ;
for i = u + 1 ... n do
    // i ist der Index der neu hinzugefügten Zwangsbedingung ;
    if isFirstFrictionConstraint(i) then
        compileFrictionLimits() ;
    w[i] ← A_{N∪i} · λ⃗_{N∪i} − b⃗_{N∪i} ;
    if w[i] ≤ 0 && λ_high[i] == 0 // w[i] ≥ 0 && λ_low[i] == 0 then
        B ← B ∪ i ;
    else if w[i] == 0 then
        N ← N ∪ i;
    else
        while true do
            Δλ⃗ ← findDirection() ;
            Δw⃗ ← A_{B∪i,B∪i} · Δλ⃗ ;
            {stepSize, c} ← findStepSize() // Zwangsbedingung c begrenzt die Schrittweite.;
            if stepSize ≤ 0 then
                return false // Algorithmus findet keine Lösung!;
            else
                w⃗_B ← w⃗_B + stepSize · Δw⃗_B ;
                λ⃗_N ← λ⃗_N + stepSize · Δλ⃗_N ;
            if i == c then
                if w[i] < 0 then
                    λ[i] ← λ_high[i] ;
                    B ← B ∪ i;
                else if w[i] > 0 then
                    λ[i] ← λ_low[i];
                    B ← B ∪ i;
                else
                    N ← N ∪ i;
                    break // Weiter mit nächster Zwangsbedingung!;
            else
                if j ∈ N then
                    N ← N − c ;
                    B ← B ∪ c ;
                    if λ[c] ≥ 0 && λ_high[c] < ∞ then
                        λ[c] ← λ_high[c];
                    else if λ[c] < 0 && λ_low[c] > −∞ then
                        λ[c] ← λ_low[c];
                else if c ∈ B then
                    N ← N ∪ c ;
                    B ← B − c ;
        end
end
```

me aus gestapelten Boxen (vgl. [43]) oder riesige Mobiles aus dutzenden oder hunderten weitgehend identischer Teilen (vgl. [17]) verzichtet.

Stattdessen kommt die Simulation eines Holzvollernters oder Harvesters als Vergleichsmodell zum Einsatz. Solche Fahrzeuge sind wesentlicher Teil der modernen Forstwirtschaft und ihre Komplexität erfordert den Einsatz von Simulatoren in Ausbildung und Training von Maschinenführern. Da im Rahmen dieser Arbeit u.a. derartige Simulatoren entwickelt wurden und die Simulation eines solchen Systems hohe Anforderungen an das Lösungsverfahren stellt, bietet es sich als Referenzmodell an.

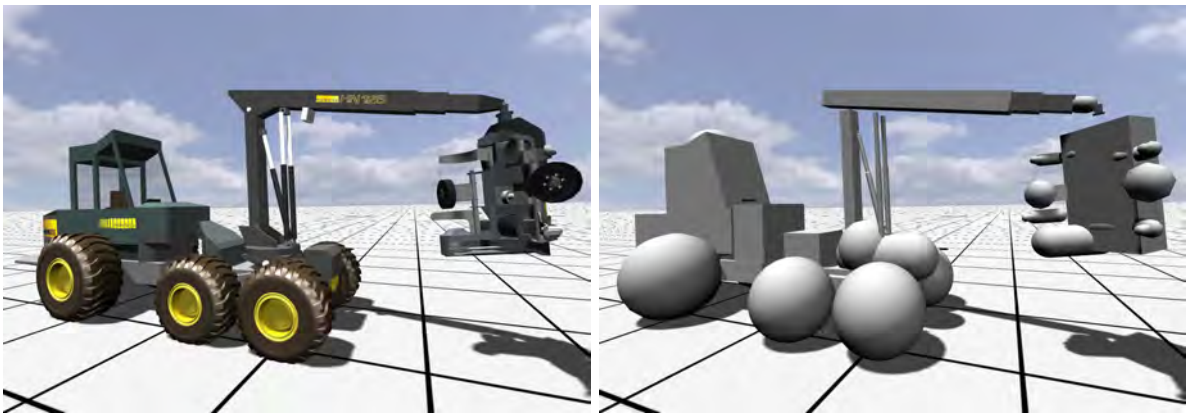


Abbildung 3.13.: Referenzmodell zur Bewertung der Lösungsverfahren. Links: Visualisierungsmodell. Rechts: Physikalisches Ersatzmodell.

Abbildung 3.13 zeigt den Screenshot der Simulation eines Harvesters, links das Visualisierungsmodell, rechts eine Darstellung des vereinfachten physikalischen Modells. Die Vereinfachung besteht in der Approximation der Kollisionskörper durch geometrische Primitive zur Beschleunigung der Kollisionserkennung. Das dargestellte Modell wird vollständig simuliert, einschließlich des Kranauslegers und des Harvesteraggregats enthält es 25 Drehgelenke (6x Räder, 2x Boogiegelenke, 5x Kran, 14x im Aggregat), 4 Lineargelenke (im Teleskopkran und in den Hydraulikzylindern), 10 Differenzialbedingungen (4x im Fahrwerk, 1x im Teleskopkran, 5x im Aggregat) sowie ein Gelenk zur Realisierung der Knicklenkung unter Berücksichtigung einer Twistachse (freie Drehung zwischen Vorder- und Hinterteil des Fahrzeugs um die Längsachse) als einen speziellen Gelenktyp. Außerdem werden alle Motoren in Gelenken durch Zwangsbedingungen realisiert. Das Fahrzeug steht auf dem Boden, daher kommen 6 Kontaktnormalen- und 12 Kontaktreibungszwangsbedingungen hinzu. Das Modell enthält außerdem zwei permanente Zyklen (gebildet durch die Hydraulikzylinder am Kran parallel zu den Kran-

gelenken). Bei Fahrmanövern können weitere Zyklen durch Kollisionen z.B. zwischen Aggregat und Fahrzeug entstehen. Über Räder und Boden werden hier keine Zyklen gebildet, da der Boden als völlig fixer Körper keine Impulse weiterleitet.

Steht der Harvester auf dem Boden, ist zu seiner Simulation die Berücksichtigung von insgesamt 201 Zwangsbedingungen notwendig, die Systemmatrix $\mathbf{A}$ hat entsprechend ein Größe von 201 Zeilen und Spalten.

Das Modell dient der Messung von Geschwindigkeit und Genauigkeit. Die Geschwindigkeitsmessungen beschreiben die durchschnittliche Zeitdauer zur Lösung des Komplementaritätsproblems. Für die Messung der Genauigkeit wurde folgendes Fehlermaß eingeführt: Da es sich um eine geschwindigkeitsbasierte Formulierung handelt, entsteht an allen Zwangsbedingungen ein positions- (bzw. orientierungs-) bezogener Fehler, der durch eine zusätzliche, positionsbezogene Fehlerkorrektur behoben werden muss (vgl. Kapitel 3.1.5). Je ungenauer die Lösung des Komplementaritätsproblems, d.h. je ungenauer die Zwangsimpulse ($h\vec{\lambda}$) die Zwangsbedingungen erfüllen, je größer ist dieser Fehler. Bei den Messungen wurde dazu die Fehlerkorrektur an den Gelenken vollständig abgeschaltet und eine Quadratsumme über alle positions- und orientierungsbezogenen Fehler in allen Gelenken gebildet. Als Messsystem kam ein IBM-kompatibler PC mit einem intel® core i7 Prozessor bei 2,66 GHz Taktfrequenz zum Einsatz. Es wurde je eine Zehntel Sekunde Echtzeit simuliert, die Schrittweite beträgt 12 ms. Das Gauß-Seidel Iterationsverfahren wurde mit 10, 100, 1000 und 10000 Iterationen gemessen. Die in Kapitel 3.5.2 beschriebene Implementierung der Dantzig Routine wurde mit unterschiedlich großen Diagonalaufschlägen auf die Systemmatrix ausgemessen, mit $1 \cdot 10^{-4}$, $1 \cdot 10^{-5}$ und $1 \cdot 10^{-9}$.

Ergebnis: Abbildung 3.14 zeigt die Ergebnisse der Messreihe. Aufgetragen ist der Gesamtfehler über der Zeit, die Skalierung der Fehlerachse ist logarithmisch. Der Wert „CFM“ in der Legende entspricht dem additiven Diagonalaufschlag auf die Systemmatrix.

Die Dantzig Routine benötigte zur Lösung des Problems durchschnittlich 6,3 ms. Um mit der oben beschriebenen Gauß-Seidel Routine (GS) (siehe 3.5.1) ungefähr in diesen Laufzeitbereich zu gelangen, sind etwa 70 Iterationen möglich. An der Grafik lässt sich gut ablesen, dass selbst mit 100 Iterationen der Fehler in der Gauß-Seidel Routine rund 100 Mal so groß ist wie bei der Dantzig Routine mit einem Diagonalaufschlag von $1 \cdot 10^{-4}$. Dies ist umso erstaunlicher, bedeutet dieser Wert doch die Möglichkeit, die

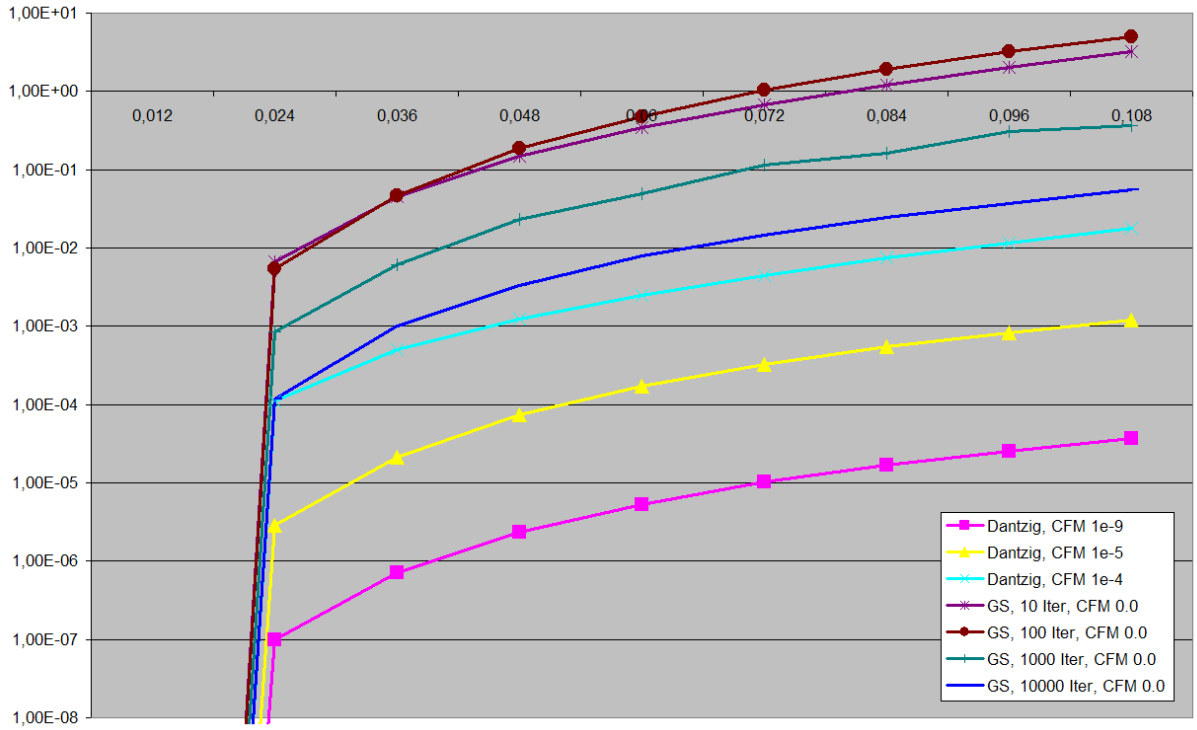


Abbildung 3.14.: Ergebnis der Fehlermessungen der unterschiedlichen Lösungsverfahren am Referenzmodell

Zwangsbedingungen relativ großzügig zu verletzen. Tatsächlich ist beim Referenzmodell schon ein deutlich kleinerer Diagonalaufschlag ausreichend, damit das Verfahren stets eine Lösung finden kann. Der angenommene Wert war jedoch auch für deutlich „problematischere“ Modelle mit mehr Kontakten und Zyklen ausreichend, so dass er als ein praxistauglicher Erfahrungswert gelten kann.

Diese Messung bestätigt die Erfahrungen während der Entwicklungen der im Rahmen dieser Arbeit untersuchten Anwendungen. Für komplexe kinematische Strukturen wie den Harvester ist die Genauigkeit der Gauß-Seidel Routine bei den möglichen Iterationzahlen von etwa 70 - 100 für eine interaktive Anwendung nicht ausreichend. In visualisierten 3D-Simulationen äußert sich die geringere Genauigkeit durch scheinbar „weiche“ Gelenkverbindungen. Gelenke biegen sich unter der Last der montierten Körper. Abbildung 3.15 zeigt die Visualisierung des Referenz-Simulationsmodells mit der Gauß-Seidel Routine und 100 Iterationen. Zwar ist hier die Fehlerkorrektur abgeschaltet, so dass die einmal entstandenen Fehler in den Gelenken nicht korrigiert werden. Allein die Größe des entstehenden Fehlers verdeutlicht aber die mangelhafte Genauig-

keit der Lösungsroutine. Die gleiche Simulationsdauer in dieser Szene lässt bei Lösung mit der Dantzig Routine mit bloßem Auge noch keinen Fehler erkennen.

Abbildung 3.15.: Wie wirkt sich der Fehler bei der Lösung der Komplementaritätsformulierung auf die Simulation des Referenzmodells aus? In dieser Sequenz wird die inverse Dynamik mit der Gauß-Seidel Routine und 100 Iterationen gelöst, die Fehlerkorrektur durch Stabilisierung ist vollständig abgeschaltet. Die Sequenz überdeckt eine Realzeit von einer Sekunde, die Zeitschrittweite der Simulation beträgt zwölf Millisekunden.

In solchen Anwendungen ist die Dantzig Routine daher klar im Vorteil. Während die Gauß-Seidel Routine gleichermaßen nach beschränkten und unbeschränkten Zwangsbedingungen löst, verläuft die Lösung des unbeschränkten Teilsystems mit der Dantzig Routine durch die $\mathbf{LDL}^T$-Zerlegung direkt und nicht iterativ. Hierdurch tritt in den unbeschränkten Zwangsbedingungen, also z.B. den klassischen Gelenkbedingungen, ein deutlich geringerer Fehler auf.

Bei der Simulation von Kontaktreibung jedoch muss die Dantzig Routine einen größeren Fehler als die simple Gauß-Seidel (GS) Routine produzieren. Der Grund hierfür liegt in der Art und Weise, wie und wann die Grenzwerte für die Reibungsimpulse bestimmt werden. In GS werden die Grenzwerte in jeder Iteration erneut an alle im System wirkenden Impulse angepasst. So kann jeder Reibungsimpuls eines jeden Kontakts Einfluss auf alle anderen Zwangsbedingungen nehmen. In der Dantzig Routine jedoch werden die Grenzwerte nur einmalig festgelegt, nachdem die Kontaktnormalenimpulse bestimmt wurden. Dies hat zur Folge, dass bei der Bestimmung der Grenzwerte eines Reibungsimpulses mögliche Einflüsse aller anderen Reibungsimpulse nicht berücksichtigt werden können. Die Beträge der Reibungsimpulse fallen bei Lösung mit Dantzig Routine tendenziell geringer aus, was sich z.B. in Fahrzeugsimulationen durch geringere Reibung der Reifen bemerkbar macht. Für solche Simulationen, bei denen der Fokus eher auf Kontaktsituationen denn auf kinematischen Strukturen liegt, kann daher gelegentlich die GS Routine die bessere Wahl darstellen.

Ein weiterer wesentlicher Vorteil der GS Routine ist die Möglichkeit, die Anzahl der Iterationen und damit die Laufzeit und die geforderte Genauigkeit an gegebene Anforderungen anpassen zu können. Sind keine interaktiven Bildwiederholraten gefragt, kann mit GS so eine große Genauigkeit bei der Simulation des Starrkörpermodells mit Reibung erreicht werden. Andererseits können in Modellen mit sehr vielen Kontakten (und wenigen Gelenken) noch interaktive Bildwiederholraten erreicht werden. In einigen Modellen mit extrem vielen Kontakten fand die Dantzig Routine nie eine Lösung, ist daher gar keine Alternative. Kapitel 4.4.1 behandelt die Simulation großer Stapel mit vielen Kontaktsituationen. Dies ist ein typisches Beispiel für ein Modell, in dem GS die bessere Wahl ist, da die Dantzig Routine hier versagte und keine Lösung liefern konnte. Gauß-Seidel im Gegensatz kann hier schon bei geringer Anzahl Iterationen visuell plausible Ergebnisse liefern.

Als Faustregel lässt sich festhalten: Liegt der Fokus auf Kontakten und Reibung, sollte GS eingesetzt werden. Sollen komplexe kinematische Strukturen und Kontakte simuliert werden, ist die Dantzig Routine eindeutig die bessere Wahl.

Kapitel 4.

Kontaktgraphenanalyse zur Optimierung von Verfahren und Modellierung

Kapitel 3 beschreibt ausführlich das Verfahren zur Bestimmung der inversen Dynamik mit Lagrange-Multiplikatoren und damit eines der Kernelemente des im Rahmen dieser Arbeit realisierten Mehrkörperdynamiksimulationssystems. Für dessen vielseitigen Einsatz ist eine flexible, effiziente und robuste Implementierung notwendig. Als ein besonders nützliches Werkzeug zur Erreichung dieses Ziels hat sich die Kontaktgraphenanalyse erwiesen, weshalb ihr dieses Kapitel gewidmet ist.

Der Kontaktgraph bildet die Konfiguration einer sich stetig verändernden Szene auf eine Graphenstruktur ab. Knoten des Graphen sind die Körper des Systems, seine Kanten werden durch Gelenke, Kontakte und weitere Zwangsbedingungen zwischen den Körpern impliziert. Als Pfadlänge zwischen zwei Körpern im Graphen wird in diesem Zusammenhang die Anzahl der Kanten eines kürzesten Pfades zwischen zwei Körpern genutzt.

Dieser Graph wird für die einfache und effiziente Erfüllung von vier grundlegenden Aufgaben im Simulationsablauf genutzt:

1. Die ***dynamische Rekonfiguration*** identifiziert Gruppen von Körpern, die aus Sicht der Dynamiksimulation als ein logischer Körper simuliert werden können. Dies tritt auf, wenn ein Modell vollständig starre Verbindungen enthält - solche können aus der Anwendungsmodellierung hervorgehen oder aber durch heuristische Verfahren zur Laufzeit einer Simulation hergestellt werden.

2. Die ***automatisierte Bestimmung von Kollisionsgruppen*** ermittelt Gruppen von Körpern, deren Mitglieder untereinander nicht auf Kollisionen hin überprüft werden müssen: Bei zwei direkt über ein Gelenk miteinander verbundenen Körpern sorgen z.B. Gelenkanschläge dafür, dass keine ungewollten Kollisionen zwischen ihnen auftreten können. Eine Kollisionserkennung zwischen diesen Körpern ist daher überflüssig. Da die Kollisionserkennung nur Körper aus unterschiedlichen Kollisionsgruppen auf Kollision hin überprüft, werden keine überflüssigen Überprüfungen durchgeführt. Damit ist diese Funktion wesentliche Grundlage der Grobphase der Kollisionserkennung.

3. Die ***Bildung von Simulationsclustern*** dient der Identifikation von Gruppen von Körpern, die vollständig unabhängig voneinander simuliert werden können, weil sie sich untereinander weder durch Gelenke noch durch Kontakte beeinflussen. Diese Funktion dient der Laufzeitoptimierung - aufgrund des superlinearen Laufzeitverhaltens der eingesetzten Lagrange-Multiplikatoren-Methode lässt sich die inverse Dynamik mehrerer kleiner Mehrkörpersysteme schneller bestimmen als die eines großen. Außerdem lässt sich die Berechnung der einzelnen Cluster parallel ausführen.

4. Die ***Bestimmung von Pfadlängen*** zwischen bestimmten Körpern der Mehrkörpersimulation. Diese Information dient dem Konzept der Stoßfortpflanzung und als Entscheidungskriterium in heuristischen Verfahren.

Alle vier Aufgaben können mithilfe klassischer Algorithmen für Tiefen- (engl.: *Depth-First Search*) und Breitensuche (engl.: *Breadth-First Search*) gelöst werden. Dies soll im Folgenden zunächst anhand eines einfachen Referenzmodells erläutert werden.

Abbildung 4.1 zeigt das Modell eines fiktiven, mobilen Roboters, der mithilfe seines Arms Stückgut in seinen Laderaum befördern kann. Abbildung 4.2 zeigt die Graphendarstellung dieser zwar fiktiven, aber durchaus repräsentativen Simulationsszene. Die roten Kanten werden durch Kontakte impliziert, die grünen durch Gelenke und die schwarzen durch starre Verbindungen. Die gestrichelte Linie symbolisiert eine temporäre, starre Verbindung.

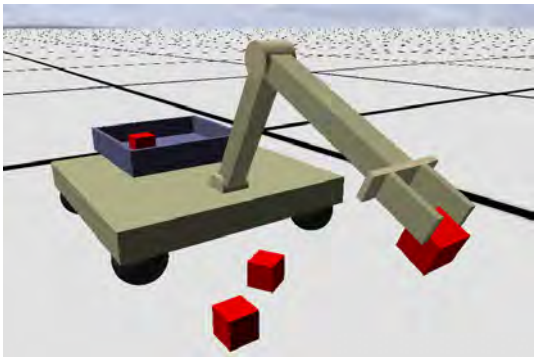


Abbildung 4.1.: 3D-Visualisierung des Modells eines fiktiven, mobilen Roboters

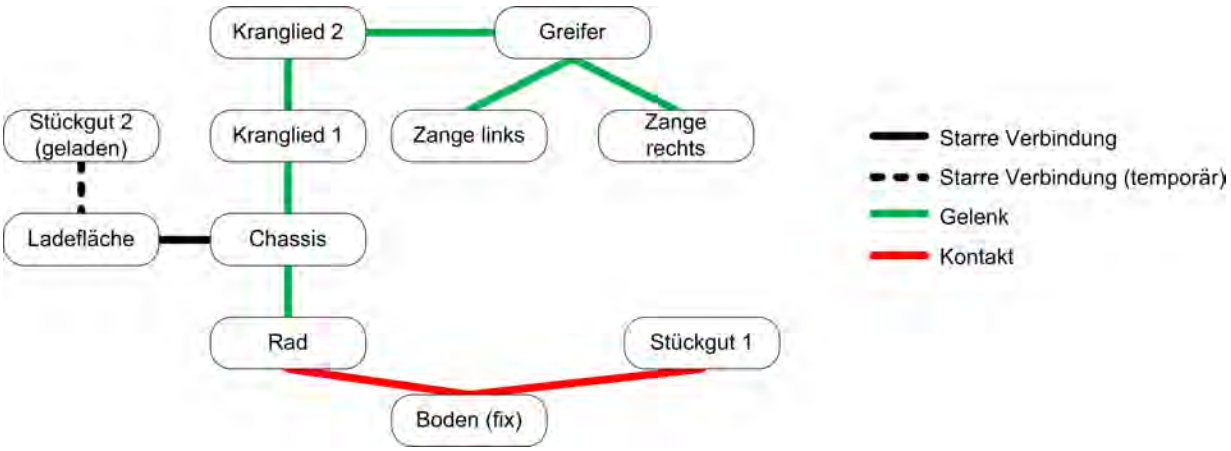


Abbildung 4.2.: Der Kontaktgraph der fiktiven Beispiel-Szene

4.1. Dynamische Rekonfiguration des Mehrkörpersystems

Im Verlauf vieler Simulationsszenarien treten Situationen auf, in denen unterschiedliche Körper eines Mehrkörpersystems Gruppen bilden, die ihrerseits als ein logischer, zusammengesetzter Körper betrachtet werden können. Am Beispiel des mobilen Roboters heißt das: Angenommen, der Roboter bietet die Möglichkeit, anstelle der Ladefläche einen zweiten Arm zu montieren. Während der Modellierung eines konkreten Anwendungsszenarios wird das eine oder andere Werkzeug ausgewählt und mittels einer starren Verbindung mit dem Chassis verbunden. Diese Verbindung könnte durch ein „starres Gelenk“ realisiert werden, das sechs Zwangsbedingungen impliziert und in der Folge alle sechs nötigen Zwangsimpulse aufbringt, um die Verbindung aufrecht

zu erhalten. Ebenso gut lassen sich Chassis und erstes Glied des Werkzeugs jedoch als *ein* logischer Körper simulieren. Dann muss die Mehrkörperdynamiksimulation nicht mit sechs Zwangsbedingungen „belastet“ werden, die nur der Erhaltung einer gänzlich starren Verbindung dienen.

Werden solche starren Verbindungen durch Kanten im Kontaktgraphen repräsentiert, entsprechen die resultierenden Zusammenhangskomponenten genau denjenigen Körpergruppen, die zu *einem* Starrkörper vereinigt werden können.

Doch starre Verbindungen, die aus der manuellen Modellierung resultieren, würden den Begriff „dynamische Rekonfiguration“ nicht rechtfertigen. Im Verlauf der Simulation des fiktiven mobilen Roboters kann es auch sinnvoll sein, klassische Gelenkverbindungen temporär durch starre Verbindungen zu ersetzen: Angenommen das Fahrzeug steht, während der Arm Stückgut aufnimmt. In dieser Zeit ist es legitim, die Räder des Fahrzeugs mit dem Chassis zu „verschmelzen“, so dass Räder und Chassis zusammen als ein Körper simuliert werden können. Der wesentliche Vorteil besteht darin, dass die durch das Radgelenk implizierten Zwangsbedingungen in der inversen Dynamik nicht berücksichtigt werden und keine entsprechenden Zwangsimpulse bestimmt werden müssen. Anhand einer Heuristik wird entschieden, dass die Gelenkverbindung vorübergehend als starr angesehen werden darf. Im Kontaktgraphen muss eine entsprechende Kante angelegt werden. Eine erneute Tiefensuche findet daraufhin veränderte Zusammenhangskomponenten und die Menge der zu simulierenden Starrkörper reduziert sich. Abbildung 4.3 zeigt die entsprechende Graphendarstellung des fiktiven Modellbeispiels. Grau hinterlegt sind jeweils die Körper, die in der gegenwärtigen Situation als ein logischer Körper betrachtet werden können und durch eine Tiefensuche als eine Zusammenhangskomponente identifiziert werden.

Ebenso kann es sinnvoll sein, Teile der Ladung mit der Ladefläche zu verschmelzen, wenn sie eine stabile Ruhelage angenommen hat, der mobile Roboter z.B. fährt und gerade nicht mit der Ladung interagiert. In diesem Fall können auch Arm und Fahrzeug logisch verschmolzen werden. Ein heuristisches Verfahren entscheidet für Teile der Ladung und die Armgelenke, wann eine Verschmelzung sinnvoll ist und legt zur Laufzeit der Simulation entsprechende, starre Verbindungen an. Eine erneute Tiefensuche findet neue Zusammenhangskomponenten, die dann je als ein Starrkörper betrachtet werden können. In Abbildung 4.4 ist die entsprechende Repräsentation im Kontaktgraphen dargestellt.

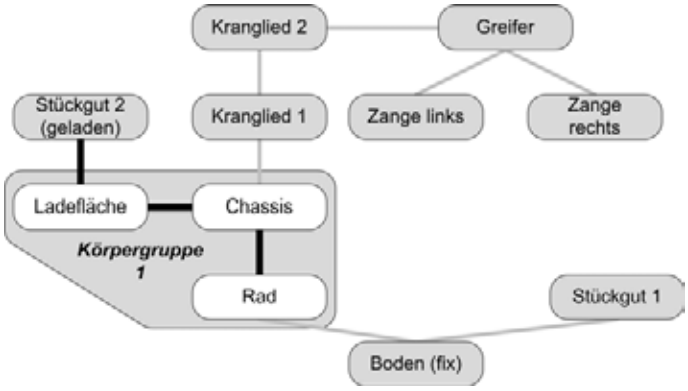


Abbildung 4.3.: Dynamische Rekonfiguration, 1-2: Steht der mobile Roboter und benutzt seinen Arm, können seine Räder mit dem Chassis verschmolzen werden, ohne dass die Realitätsnähe der Simulation beeinträchtigt wird.

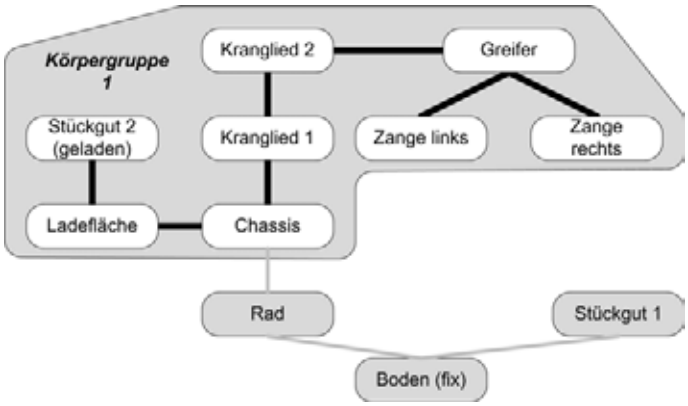


Abbildung 4.4.: Dynamische Rekonfiguration, 2-2: Fährt der mobile Roboter und benutzt den Arm gerade nicht, können große Teile des Systems, die nicht zum Fahrwerk gehören, als ein logischer Starrkörper betrachtet werden.

4.2. Automatisierte Bestimmung von Kollisionsgruppen

Nachdem die Menge der zu simulierenden Körper des Systems ermittelt ist, kann die Kollisionserkennung erfolgen. Noch vor der Grobphase der Kollisionserkennung (zum zeitlichen Ablauf vgl. Kapitel 5.2) müssen diejenigen Körpergruppen identifiziert werden, die strukturbedingt nicht auf Kollisionen hin überprüft werden müssen. Beispiele aus dem Referenzmodell hierfür sind Rad und Chassis oder direkt benachbarte Glieder des Arms. Kollisionen zwischen diesen Körpern sind bedingt durch die kinematische Struktur nicht möglich. Dementsprechend sollen Rad und Chassis derselben Kollisionsgruppe angehören. Die Kollisionserkennung betrachtet ausschließlich solche Körper-

paarungen, die unterschiedlichen Kollisionsgruppen angehören: Die Menge der Kollisionsgruppen ist die Ausgangsmenge, auf der die Kollisionserkennung arbeitet, nicht die Menge aller Körper im System.

Abbildung 4.5 zeigt die Graphendarstellung der Kollisionsgruppen des fiktiven Beispiels. Grau hinterlegt jetzt diejenigen Körper, die einer Kollisionsgruppe zugeordnet werden. Deutlich zu erkennen ist, dass die automatisierte Graphensuche eine im Vergleich zur Menge der simulierten Körper deutlich reduzierte Menge von Kollisionsgruppen findet. Der einzige Mehraufwand während der Modellierung besteht darin, das Gelenk zwischen „Kranglied 1“ und „Greifer“ mit der Zusatzinformation zu versehen, dass es die Kollisionsgruppen der angeschlossenen Körper trennt. Im Beispielmodell ist dies notwendig, weil Greifer und Fahrzeug kollidieren können sollen. In der vorliegenden Implementierung leistet dies die Eigenschaft „separateColliderGroups“ einer jeden Gelenkklasse.

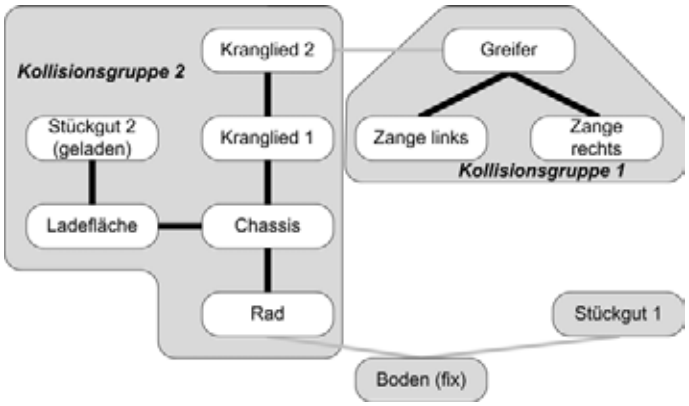


Abbildung 4.5.: Die Bestimmung von Kollisionsgruppen: Der Kontaktgraph erlaubt die automatische Bestimmung von Kollisionsgruppen, die die Grundmenge für die Grobphase der Kollisionserkennung bilden.

4.3. Bildung von Simulationsclustern

Ist die Kontaktpunktbestimmung abgeschlossen und sind alle Kontakte identifiziert, kann mit der Bestimmung der inversen Dynamik fortgefahren werden, d.h. mit der Berechnung von Zwangsimpulsen in Kontakten und Gelenken. Das eingesetzte Verfahren der inversen Dynamik mit Lagrange-Multiplikatoren besitzt ein superlineares Laufzeitverhalten in Bezug auf die Anzahl der Zwangsbedingungen. Deshalb lassen sich zwei

Mehrkörpersysteme der Größe $n/2$ schneller lösen als eines der Größe n.

Ein Mehrkörpersystem, das nicht weiter unterteilt werden kann, wird *Simulationscluster* genannt. Die Suche nach Simulationsclustern stellt die geringsten Anforderungen an eine Kante: Jeder Kontakt, jedes Gelenk und jede sonstige Verbindung zwischen Körpern erzwingt die Simulation der beteiligten Körper im selben Cluster. Ohne weitere Intelligenz müsste daher die Szene aus Abbildung 4.2 vollständig in einem einzigen Cluster simuliert werden.

Für die meisten Anwendungen ist es jedoch legitim, den Boden, auf dem sowohl der mobile Roboter als auch „Stückgut 1“ liegen, als völlig fix anzunehmen. Ein vollständig unbeweglicher Boden kann im Starrkörpermodell keine Auswirkung vom Roboter auf „Stückgut 1“ übertragen - entsprechend kann an dieser Stelle eben doch eine Auftrennung vorgenommen werden. Der fixe Boden wird schlicht in beiden Simulationsclustern berücksichtigt. Da er selbst als völlig unbeweglich angenommen wird, können keine Konflikte zwischen den Simulationsclustern auftreten. Abbildung 4.6 zeigt die entsprechende Repräsentation im Kontaktgraphen.

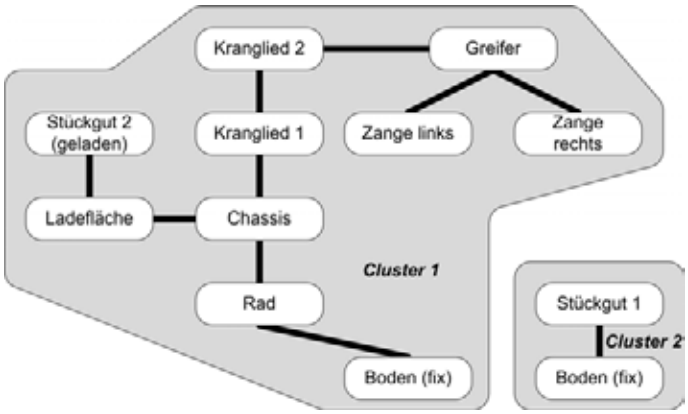


Abbildung 4.6.: Die Clusterbildung: In diesem Anwendungsbeispiel dient der Kontaktgraph der Findung unabhängiger Mehrkörpersysteme in einer Szene. Ein vollständig fixer Körper kann in mehrere Cluster eingefügt werden.

4.4. Bestimmung der Kontaktpfadlänge als Grundlage für Stoßfortpflanzung und Heuristiken

Die Länge des Kontaktpfades zwischen den Körpern A und B ist die Anzahl der Kanten eines kürzesten Pfades (also Gelenke oder Kontakte) im Kontaktgraphen zwischen den

Körpern A und B. Eine simple Breitensuche liefert sogar alle Pfadlängen von einem Knoten zu allen anderen im Graphen mit linearer Zeitkomplexität bezogen auf die Anzahl der Kanten und die Anzahl der Knoten im Graphen. Diese Information ist Grundlage für das Konzept der Stoßfortpflanzung (engl.: *Shock Propagation*). Sie dient darüber hinaus als Entscheidungskriterium in heuristischen Verfahren.

4.4.1. Optimierte Adaption des Prinzips der Stoßfortpflanzung

Die Simulation eines Stapels mit einem Dynamiksimulationsverfahren stellt eine besondere Herausforderung dar und wird daher gerne als Referenzmodell zu Vergleichszwecken herangezogen. Die Schwierigkeit: Kleine Fehler in der Behandlung der ruhenden Kontakte zwischen je zwei Körpern addieren sich über die Höhe eines Stapels auf - Körper weit oben auf einem Stapel führen daher in Folge kleiner Korrekturbewegungen zwischen je benachbarten Körpern schon große Bewegungen durch, die für sich betrachtet unrealistisch aussehen und schnell zum Einsturz des Stapels führen.

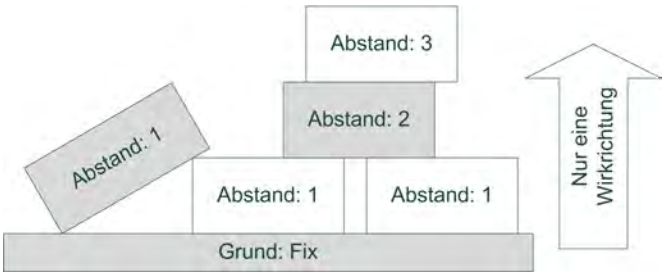
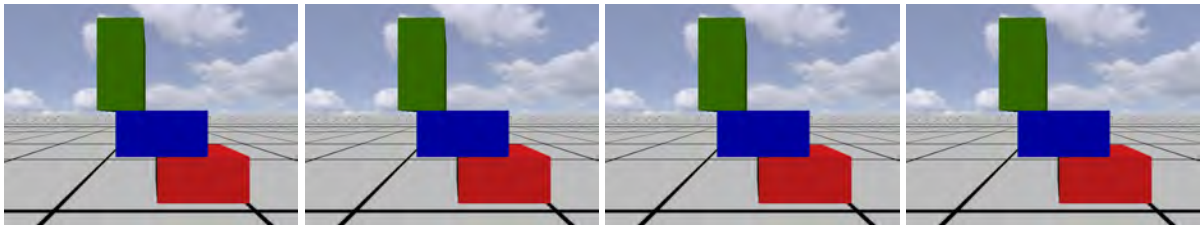


Abbildung 4.7.: Anwendung der Kontaktpfadlänge in der Stoßfortpflanzung: Weiter oben liegende Boxen haben keinen Einfluss auf weiter unten liegende.

Um die Simulation solcher Stapel zu stabilisieren, macht sich die Stoßfortpflanzung das Wissen um die Position der Körper auf dem Stapel und die Tatsache, dass es eine unbewegliche Basis gibt, zu Nutze. Abbildung 4.7 zeigt eine zweidimensionale Darstellung eines Stapelmodells. Die einzelnen Boxen sind je mit der Information der Kontaktpfadlänge von sich aus bis zum unbeweglichen Untergrund beschriftet. Die erste Annahme ist nun, dass die Boxen, die direkt auf dem Untergrund lagern, keinen Einfluss auf selbigen haben, weil seine Masse und Trägheit erheblich größer ist. Weiterhin wird angenommen, dass die Boxen der zweiten Ebene keinen Einfluss auf die Boxen der ersten Ebene haben, weil die ihrerseits durch den Boden gestützt werden, usw..

Guendelman et al. [57] und Erleben [43] passen ihre Formulierungen der inversen



$0,25s.$ $0,5s.$ $0,75s.$ $1,0s.$

Abbildung 4.8.: Das Problemmodell mit einfacher Stoßfortpflanzung: Der Stapel bleibt wider der Physik stabil.

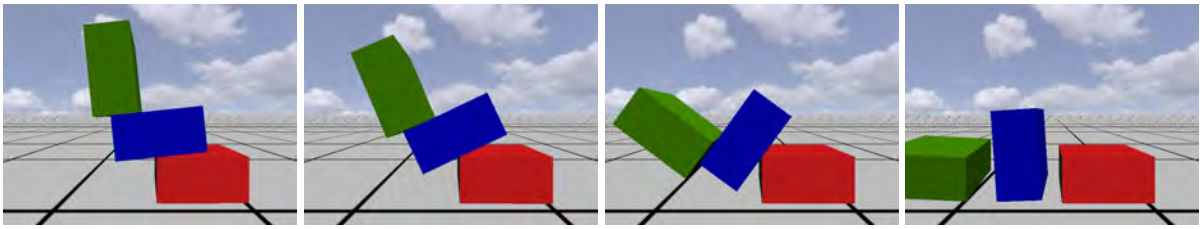
Dynamik an, um obige Annahmen einfließen zu lassen. Die Algorithmen werden so angepasst, dass die inverse Dynamik entsprechend der Stapelordnung von unten nach oben bestimmt wird. Das bedeutet, für einen im Stapel weiter unten liegenden Quader existieren alle über ihm liegenden Quader *gar nicht*. Die höher liegenden Körper reagieren lediglich auf tiefer liegende. Adaptiert man diesen Ansatz in ein System aus Zwangsbedingungen, müssen diese derart formuliert werden, dass sie keinen Einfluss auf im Stapel tiefer liegende Körper haben.

Sei im Folgenden Körper i der tiefer liegende Körper, Körper j der höher liegende. Dann hätten die Teil-Jacobimatrizen (vgl. mit Gleichung 3.70 auf Seite 106) die Form:

$$\begin{aligned} \mathbf{J}_i &= \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \\ \mathbf{J}_j &= \begin{pmatrix} n_1 & n_2 & n_3 & -(\vec{n} \times \vec{r}_j)_1 & -(\vec{n} \times \vec{r}_j)_2 & -(\vec{n} \times \vec{r}_j)_3 \end{pmatrix} \end{aligned} \tag{4.1}$$

Diese „kompromisslose“ Interpretation der Stoßfortpflanzung hat jedoch eine erhebliche Verfälschung der Simulation zur Folge. Abbildung 4.8 zeigt hierfür ein beeindruckendes Beispiel: Unter der Annahme, dass alle drei Quader die gleiche Masse besitzen, erwartet man, dass durch die Last des obersten Quaders der mittlere und der obere kippen werden. Wendet man jedoch die Stoßfortpflanzung wie oben dargestellt an, bleibt der Stapel stabil stehen, weil der oberste Quader entsprechend der Stapelordnung keinen Einfluss auf den mittleren hat.

Um diese Schwäche zu beseitigen ohne gänzlich auf den stabilisierenden Effekt verzichten zu müssen, lässt sich das Prinzip der Stoßfortpflanzung auf der Ebene von Zwangsbedingungen intelligenter einsetzen: Anstatt jeden Einfluss von im Stapel weiter oben liegenden Körpern auf weiter unten liegende zu verhindern, lässt sich auch nur der Anteil unterbinden, der direkt „gerade nach unten“ wirkt. Das heißt, die Zwangs-



$0,25s.$ $0,5s.$ $0,75s.$ $1,0s.$

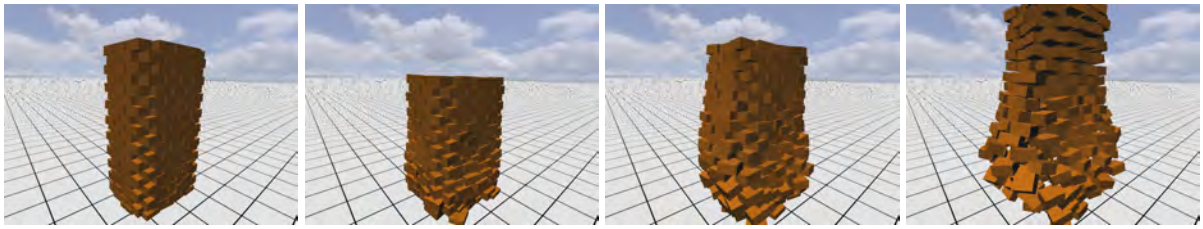
Abbildung 4.9.: Das Problemmodell mit optimierter Stoßfortpflanzung: Trotz Stoßfortpflanzung kippt der instabile Stapel, wie man es erwartet.

bedingung soll nur den z-Anteil (genau entlang der Gravitationsrichtung) der translatorischen Geschwindigkeit nicht an den tiefer liegenden Körper weiterleiten. Die Teil-Jacobimatrizen erhalten dann die Form:

$$\begin{aligned} \mathbf{J}_i &= \begin{pmatrix} -n_1 & -n_2 & 0 & (\vec{n} \times \vec{r}_i)_1 & (\vec{n} \times \vec{r}_i)_2 & (\vec{n} \times \vec{r}_i)_3 \end{pmatrix} \\ \mathbf{J}_j &= \begin{pmatrix} n_1 & n_2 & n_3 & -(\vec{n} \times \vec{r}_i)_1 & -(\vec{n} \times \vec{r}_i)_2 & -(\vec{n} \times \vec{r}_i)_3 \end{pmatrix} \end{aligned} \tag{4.2}$$

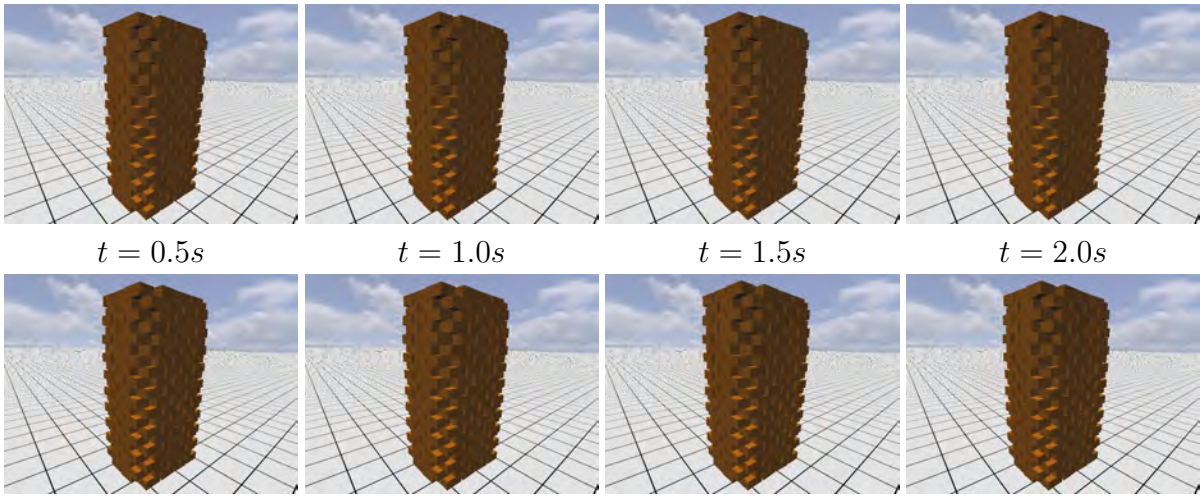
Mit dieser Interpretation der Stoßfortpflanzung verhält sich das Referenzmodell entsprechend den Erwartungen und kippt. Die Sequenz in Abbildung 4.9 belegt dies. Hierfür sorgt das Versatzdrehmoment, das die Gewichtskraft des obersten Quaders am mittleren hervorruft, weil sie seitlich versetzt vom Schwerpunkt des mittleren Körpers angreift.

Obwohl diese auf größere Realitätsnähe hin optimierte Interpretation der Stoßfortpflanzung eine weniger deutliche Verfälschung der physikalischen Gesetzmäßigkeiten in der Simulation zur Folge hat, bleibt der positive Einfluss auf die Stabilität von Stapelsimulationen erhalten. Die Abbildungen 4.10 und 4.11 zeigen die Simulation eines Stapels aus 240 Quadern, die erste ohne, die zweite mit eingeschalteter, optimierter Stoßfortpflanzung. Zu Beginn der Simulation treten 1982 Kontaktpunkte auf. Als Lösungsroutine kommt das Gauß-Seidel Verfahren mit nur 10 Iterationen zum Einsatz. Der Fehlerkorrekturparameter k_{er} hat jeweils den Wert 0.15, die Zeitschrittweite beträgt $1/100$ Sekunde. Ohne den Trick der Stoßfortpflanzung zeigt das Modell schon nach einer halben Sekunde Simulationszeit einen erheblichen Fehler - hier kommen auch die Schwächen der Constraintstabilisierung zur Fehlerkorrektur deutlich zur Geltung. Mit sonst identischen Parametern jedoch eingeschalteter, optimierter Stoßfortpflanzung, bleibt dasselbe Modell stabil stehen - übrigens auch noch nach drei und auch nach



$t = 0.5s$ $t = 1.0s$ $t = 1.5s$ $t = 2.0s$

Abbildung 4.10.: Simulation eines Stapels aus 240 Quadern ***ohne*** Stoßfortpflanzung. Nach spätestens 2 Sekunden bricht der Stapel völlig zusammen.



$t = 0.5s$ $t = 1.0s$ $t = 1.5s$ $t = 2.0s$

$t = 3.0s$ $t = 4.0s$ $t = 5.0s$ $t = 6.0s$

Abbildung 4.11.: Simulation eines Stapels aus 240 Quadern ***mit*** optimierter Stoßfortpflanzung. Der Stapel steht deutlich länger und stabiler.

sechs Sekunden.

Das „Zusammensacken“ des Stapels ohne Stoßfortpflanzung lässt sich so erklären: Die Kollisionserkennung generiert nur dann mehr als einen Kontaktpunkt zwischen zwei Quadern, wenn je zwei Flächen nahezu parallel zueinander liegen. Ist dies nicht der Fall, wird nur ein Kontaktpunkt an der Stelle der tiefsten Durchdringung generiert. Wird der notwendige Grad an Parallelität zwischen zwei Quadern einmal nicht mehr erreicht, ist z.B. die linke Seite eines Quaders tiefer in den unter ihn liegenden Quader eingesunken, so wird nur an dieser Stelle ein einzelner Kontaktpunkt generiert. Die Fehlerkorrektur durch Constraintstabilisierung wird die beiden Quader nun an dieser linken Seite auseinander drücken, so dass in nachfolgenden Schritten die rechte Seite tief einsinkt. Hierdurch tendieren Quader im Stapel zum „Hin- und Herwackeln“, sobald ein gewisser

Grad an Parallelität erstmalig unterschritten ist. Eine Kontaktpunktgenerierung, die auch im Falle von großer Nicht-Parallelität mehrere Kontaktpunkte generiert, würde daher vermutlich ebenfalls zu einer Stabilisierung beitragen. Die Tatsache, dass im Beispiel mit eingeschalteter Stoßfortpflanzung der notwendige Grad an Parallelität jedoch nie unterschritten wird, belegt eindrucksvoll den stabilisierenden Einfluss des Verfahrens.

Trotz der erhöhten Eigenstabilität des Stapels ist er äußeren Einflüssen gegenüber selbstverständlich unverändert empfindlich. Abbildung 4.12 zeigt, wie er auf „Beschuss“ durch eine schwere Kugel reagiert.

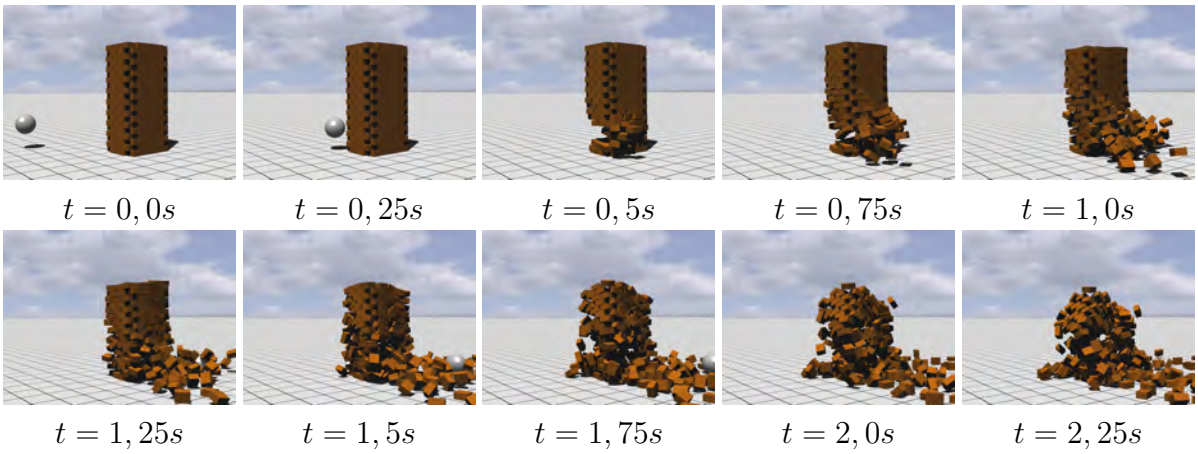


Abbildung 4.12.: Auch der durch Stoßfortpflanzung stabilisierte Stapel lässt sich selbstverständlich durch äußere Einwirkungen beeinflussen, wie hier durch „Beschuss“.

4.4.2. Die Kontaktpfadlänge als Entscheidungskriterium in Aktivierungsheuristiken

In interaktiven Anwendungen, in denen umfangreiche Szenarien simuliert werden müssen, ist es oft unerlässlich, Teile der Szene zu deaktivieren, d.h. solche Körper der Simulation, die gegenwärtig keinen relevanten Einfluss auf den Gesamtablauf haben, nicht zu simulieren. Hierdurch kann das Laufzeitverhalten deutlich verbessert werden, ohne dass der Anwender die Deaktivierung einzelner Körper der Simulation wahrnimmt. Als einfaches Beispiel kann wiederum das fiktive Roboter-Beispiel aus Abbildung 4.1 herangezogen werden: Durchfährt der Roboter eine große Szenerie mit viel herumliegendem Stückgut, ist es nicht notwendig, entfernte und ruhende Körper des Stückguts

zu simulieren. Typische Kriterien zur Deaktivierung eines Körpers sind z.B. seine kinetische Energie oder schlicht seine Geschwindigkeit.

Ein wertvolles Kriterium zur Reaktivierung kann jedoch auch der Pfadabstand sein, jetzt nicht nur zum fixen Untergrund, sondern z.B. zu den Teilen des Greifers des Roboters. Unterschreitet sie einen festgelegten Wert, wird der Körper reaktiviert und kann vom Greifer auch indirekt über mehrere Zwischen-Kontakte beeinflusst werden. Überschreitet sie den Grenzwert, verhält sich das entfernte Stückgut wie ein fixer Körper und damit wie ein unbewegliches Hindernis. Dieses Kriterium wurde vor allem bei der Realisierung des Forwardersimulators eingesetzt. Details zur dieser Anwendung beschreibt Kapitel 6.2.

4.5. Softwaretechnische Realisierung

Ein Kontaktgraph wird häufig zur Simulationslaufzeit modifiziert. So führt jeder neu auftretende Kontakt zwischen zwei Körpern eine neue Kante in den Graphen ein. Um den Synchronisierungsaufwand zwischen Szene und einer expliziten Datenstruktur für den Kontaktgraphen zu vermeiden, werden im hier realisierten Framework die ohnehin vorhandenen Klassen soweit wie möglich direkt für die Realisierung des Kontaktgraphen genutzt.

Abbildung 4.13 zeigt den relevanten Ausschnitt des Klassendiagramms. Knoten des Kontaktgraphen sind die Körper und damit Instanzen der Klassen `RBDRigidBody`. Entsprechend definiert diese Klasse die Methoden für Tiefen- und Breitensuche. Kanten werden immer dort impliziert, wo relative Freiheitsgrade zwischen zwei Körpern eingeschränkt werden. Die Klasse `RBDEdge` repräsentiert eine abstrakte Kante. Abstrakt deshalb, weil sie von einem der Graphenalgorithmen je nach Anwendung als Kante gewertet werden kann aber nicht muss. Gelenke und Kontakte, die genau zwei Körper referenzieren, beerben diese Klasse und repräsentieren damit selbst direkt eine abstrakte Kante. Andere Zwangsbedingungstypen wie z.B. Differenziale, die die relativen Bewegungen von mehr als zwei Körpern in Relation setzen, müssen explizit mehrere Instanzen der Klasse `RBDEdge` anlegen.

Damit aus einer abstrakten Kante eine für die jeweilige Anwendung konkrete wird, definiert die Klasse `RBDEdge` die Methode `bool requires(RBDRequirement req)`. Mögliche Requirements spiegeln die Anwendungen des Kontaktgraphen wider:

- OneBody: Eine abstrakte Kante liefert auf diese Anfrage „true“, wenn sie impliziert,

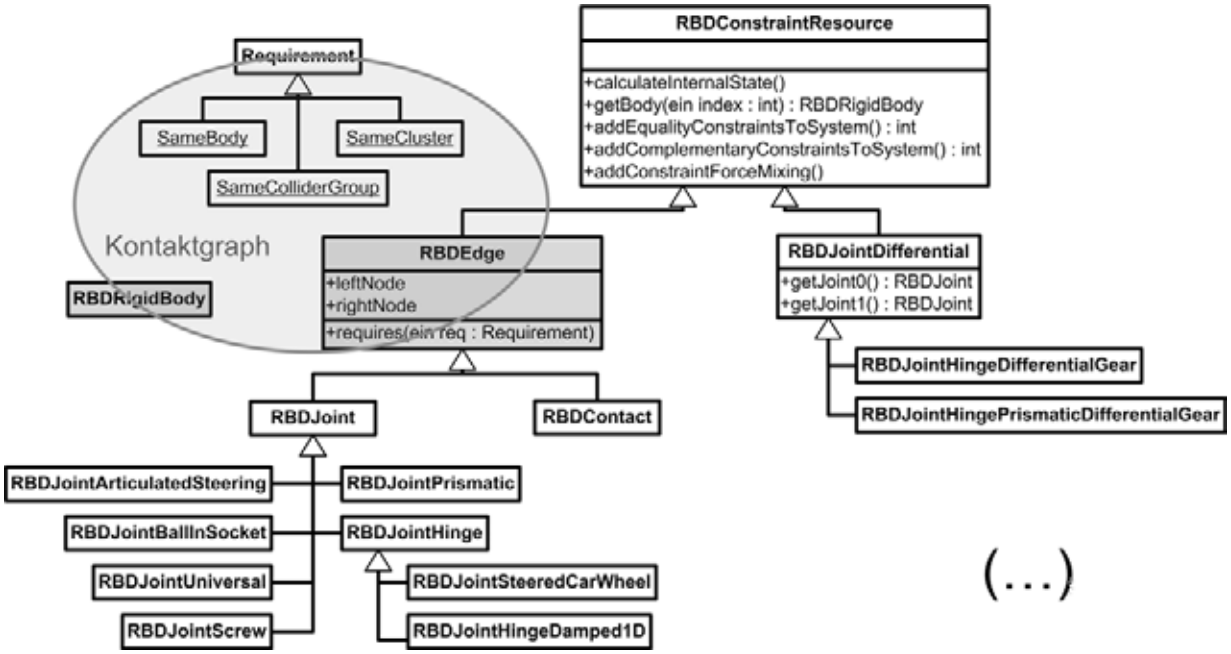


Abbildung 4.13.: Ausschnitt aus dem Klassendiagramm, das die für den Kontaktgraphen relevanten Klassen und Beziehungen zeigt.

dass die verbundenen Körper als ein logischer Körper simuliert werden können.

- OneColliderGroup: Eine Kante liefert auf diese Anfrage „true“, wenn sie impliziert, dass die verbundenen Körper zu einer Kollisionsgruppe gehören und deshalb nicht auf gegenseitige Kollision hin überprüft werden müssen.

- OneCluster: Eine Kante liefert auf diese Anfrage „true“, wenn sie impliziert, dass die verbundenen Körper im selben Simulationscluster behandelt werden müssen.

Die Methoden für Tiefen- und Breitensuchen nehmen als Parameter je eine Instanz der Klasse `Requirement` entgegen. Ein Graphendurchlauf nutzt dann nur diejenigen abstrakten Kanten, welche die jeweilige Anforderung („Requirement“) erfüllen.

Algorithmus 5 beschreibt die rekursive Tiefensuche auf abstrakten Kanten zur Bestimmung von Zusammenhangskomponenten in Pseudocode. Einziger Unterschied zum klassischen Algorithmus ist die Abfrage, ob eine betrachtete Kante die an sie gestellte Anforderung („requirement“) erfüllt.

Algorithmus 6 beschreibt eine iterative Breitensuche auf abstrakten Kanten. Auch die Breitensuche kann zur Bestimmung von Zusammenhangskomponenten genutzt werden. Im dargestellten Pseudocode werden jedoch die Pfadabstände zwischen einem

Algorithmus 5: `depthFirstSearch()`, rekursive Tiefensuche zur Bestimmung der Zusammenhangskomponenten im Kontaktgraphen mit abstrakten Kanten

Eingabe : Ein Körper der Mehrkörperszene „node“ und eine Anforderung an die Kanten („requirement“)
Ausgabe: Die Zusammenhangskomponente des eingegebenen Körpers als Menge von Knoten („connectedComponent“)

```
if node.myColor != WHITE then
    return;
myColor ← BLACK;
connectedComponent ← connectedComponent ∪ node;
foreach edge ∈ getEdgesOf(node) do
    if edge enforces(requirement) then
        otherNode ← getOtherNode (edge, node) ;
        if otherNode.myColor == WHITE then
            depthFirstSearch(otherNode, connectedComponent, requirement) ;
        end
    end
end
```

Anfangsknoten (bzw. -körper) „node“ und allen anderen Knoten bestimmt.

4.6. Bewertung der Verfahren der Kontaktgraphenanalyse

Die Werkzeuge „dynamische Rekonfiguration“ und „automatische Bestimmung von Kollisionsgruppen“ dienen vor allem der Automatisierung von Modellierungsarbeiten. Beide Arbeiten könnten teilweise auch manuell durchgeführt werden. Die Verfügbarkeit eines Kontaktgraphen und seine systematische und automatisierte Analyse erlauben jedoch ihre Automatisierung. So lassen sich z.B. aus mehreren Bestandteilen zusammengesetzte Modelle ohne wesentlichen Mehraufwand automatisiert mit optimaler physikalischer Struktur simulieren. Ein Beispiel hierfür sind die Forstmaschinensimulatoren aus Kapitel 6: Die Modelle werden typischerweise mindestens aus den drei Teilen Fahrwerk, Kran und Aggregat oder Greifer zusammengesetzt. Durch die automatische dynamische Rekonfiguration werden der erste Körper des Krans und die Kranbasis im Fahrgestell automatisch als ein logischer Körper erkannt und simuliert, unabhängig davon, welcher konkrete Kran gerade auf dem Fahrzeug „montiert“ ist. Darüber hinaus bietet

Algorithmus 6: `breadthFirstSearch()`, Breitensuche zur Bestimmung des Pfadabstandes zwischen einem gegebenem und allen anderen Körpern im Kontaktgraphen mit abstrakten Kanten

Eingabe : Ein Körper der Mehrkörperszene „node“ und eine Anforderung an die Kanten („requirement“)
Ausgabe: Für alle Körper / Knoten in derselben Zusammenhangskomponente wie „node“ werden die Pfaddistanzen zu diesem bestimmt.

```
if node.myColor != WHITE then
    return;
node.distance ← 0 ;
Q ← ∅ ;
Q ← Q ∪ node;
while !isEmpty(Q) do
    activeNode ← firstElementOf(Q) ;
    foreach edge ∈ getEdgesOf(node) do
        if edge enforces(requirement) then
            otherNode ← getOtherNode (edge, node) ;
            if otherNode.myColor == WHITE then
                otherNode.myColor ← GRAY ;
                otherNode.distance ← activeNode.distance +1 ;
                Q ← Q ∪ otherNode;
            end
        end
    end
    Q ← Q- activeNode;
    activeNode.myColor ← BLACK;
end
```

sie ein wertvolles Werkzeug für externe heuristische Verfahren, um Körper sehr einfach miteinander zu vereinigen und wieder aufzuteilen. Ein Beispiel hierfür sind die Fahrwerke der Forstmaschinensimulatoren: Die Fahrzeuge besitzen sechs oder acht Räder. Eine anwendungsspezifische Heuristik sorgt dafür, dass bei stehendem Fahrzeug die Räder fest mit ihrer Aufhängung verbunden werden. Hierdurch lassen sich 30 (6 Räder) bzw. 40 (8 Räder) Zeilen in der Jacobimatrix direkt einsparen. Weil als Folge hieraus auch die betroffenen Differenziale wegfallen, fallen zusätzlich vier bzw. sechs Zeilen aus Differenzialbedingungen weg. Ebenso können unbewegte Gelenke am Kran fixiert werden. So kann auf Seiten der Anwendungsentwicklung durch gezieltes Anlegen von Kanten im Kontaktgraphen in Form von „starren Gelenken“ dafür gesorgt werden, dass zu jeder Zeit immer nur eine minimale Menge notwendiger Gelenke tatsächlich simuliert werden muss. Dies unterstützt das Erreichen von hohen Bildwiederholraten der Simulation, die für die Akzeptanz von Anwendungen wie den Forstmaschinensimulatoren im Schulungsbereich von großer Bedeutung sind.

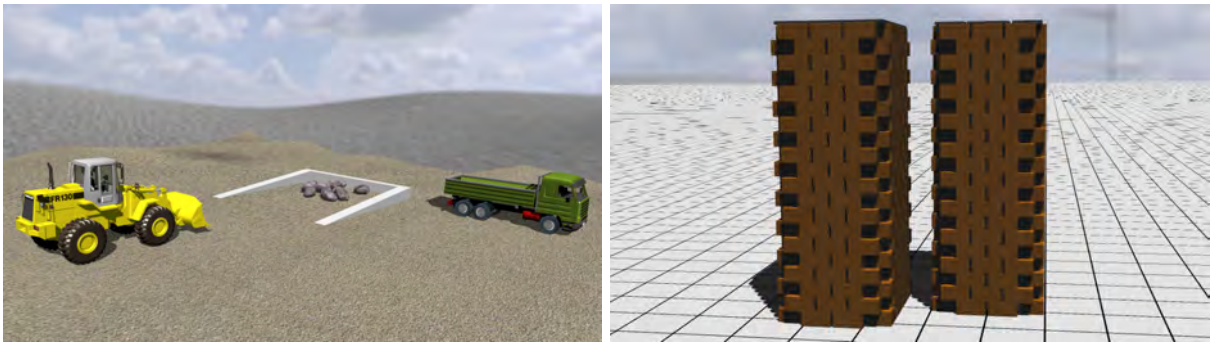


Abbildung 4.14.: Referenzmodelle zur Bewertung der Clusterbildung: Links eine Simulation bestehend aus Radlader, LKW und Ladegut, rechts zwei Stapel bestehend aus je 240 Quadern

	Modell	ohne Clusterbildung	mit Clusterbildung
Inverse Dynamik:	Radlader	5,06 ms.	1,09 ms.
	Stapel	3745 s.	2464 s.
Clusterbildung:	Radlader	0 ms.	0,18 ms.
	Stapel	0 s.	1,97 s.

Tabelle 4.1.: Rechenzeiten zur Bestimmung der inversen Dynamik in zwei Referenzmodellen jeweils ohne und mit Clusterbildung

Abbildung 4.14 zeigt zwei Referenzmodelle, an denen sich die Vorteile der Bildung

von Simulationsclustern belegen lassen. In Tabelle 4.1 sind die Rechenzeiten für das einmalige Lösen der resultierenden Komplementaritätsprobleme (inverse Dynamik) und für das Bestimmen der Cluster aufgeführt. Es ist deutlich zu erkennen, dass der Vorteil der Zerlegung des Problems in mehrere kleinere Simulationscluster den Nachteil durch den zusätzlichen Schritt der Clusterbildung klar überwiegt. Natürlich ist der Vorteil nur vorhanden, solange sich die Szene tatsächlich günstig zerlegen lässt. Das Stapelmodell lässt sich in zwei gleichgroße, das Radladermodell in mindestens drei Cluster gleicher Größenordnung zerlegen. In dynamischen Modellen kann diese gute Zerlegbarkeit natürlich nicht immer gewährleistet werden: Verbindet man z.B. beide Stapel durch ein gemeinsames Dach oder belädt man den Kipper mit allen Steinen und berührt ihn gleichzeitig mit einem Körper des Radladers, ist keine Zerlegung mehr möglich. Da die Bildung von Simulationsclustern selbst in Relation zur Bestimmung der inversen Dynamik nur sehr geringe Laufzeiten aufweist, ist der Mehraufwand in solchen Fällen mit negativem Ergebnis (keine trennbaren Cluster vorhanden) hierfür allerdings vertretbar. Für solche Modelle, in denen klar ist, dass nur selten oder nie eine Zerlegung möglich sein wird, lässt sich die Funktion in der vorliegenden Implementierung zudem abschalten.

Die Kontaktpfadlänge als Grundlage von Heuristiken wird in der Anwendung Forwardersimulator (vgl. Kapitel 6.2) eingesetzt und bietet hier eine sinnvolle Ergänzung anderer heuristischer Entscheidungsregeln für Aktivierung und Deaktivierung von Teilen der Simulation. Die Stoßfortpflanzung bietet zwar beeindruckend stabile Simulationen von extremen Stapeln, hat aber dennoch nur geringe Praxisrelevanz: In den in Kapitel 6 untersuchten Anwendungen treten derart riesige Stapel nicht auf. In Modellen, in denen z.B. wie im fiktiven Beispielmodell mit einem Greifer Dinge von einem Stapel aufgehoben werden, lässt sich die Stapelhöhe eines Körpers außerdem nicht mehr sinnvoll bestimmen, weil der Greifer zwar „von oben“ auf einen Stapel zugreifen kann, selbst aber nur über einen einzelnen Kontakt (Rad - Boden) mit dem Boden verbunden ist. Daher ist das Verfahren zumeist eher von theoretischem Interesse.

Kapitel 5.

Implementierung in einem VR-System

Die letzten beiden Kapitel haben detailliert die Verfahren behandelt, die zur Realisierung einer Mehrkörperdynamiksimulation eingesetzt werden können, die die Grundlage vielseitiger Anwendungsentwicklungen sein soll. Der Schwerpunkt lag dabei auf dem Aspekt der effizienten und robusten Bestimmung der inversen Dynamik.

In diesem Kapitel werden nun weitere wesentliche technische Aspekte des im Rahmen dieser Arbeit entstandenen Frameworks behandelt. Die Beschreibungen dienen als Dokumentation für Anwender des Systems, die auf dieser Grundlage weitere Anwendungen realisieren möchten. Sie dienen aber auch Entwicklern, die Weiterentwicklungen an der vorliegenden Implementierung vornehmen möchten. Daher werden grundlegende Aspekte der Modellierung an der grafischen Oberfläche ebenso behandelt wie sehr grundlegende objektorientierte Softwarestrukturen, einige algorithmische Details einer laufzeiteffizienten Implementierung und natürlich die Integration in die zeitliche Simulationssteuerung. Die zugrunde gelegte Softwareplattform ist das am Institut für Mensch-Maschine-Interaktion der RWTH Aachen mitentwickelte 3D-Robotersimulations- und Virtual-Reality-System (VR-System) VEROSIM®.

5.1. Struktur der Implementierung

Die realisierte Mehrkörperdynamiksimulation ist auf drei Basis-Bibliotheken aufgeteilt: Die Bibliothek VSLibRBDynamX ist der Kern der Implementierung. Sie behandelt Kontaktpunktberechnung, Kontaktgraphenanalyse, inverse Dynamik und die Animation der Bewegungsgleichungen. VSLibRBDynMath stellt einige spezialisierte Mathematikklassen und -funktionen zur Verfügung. Hierzu gehören spezialisierte Klassen für Matrizen und Vektoren sowie entsprechende Funktionen, geometrische Funktionen und Funktio-

nen und Klassen für räumliche Transformationen. Das VEROSIM®-Plugin VSPluginRB-DynamX bildet die Schnittstelle zwischen VR-System und Mehrkörperdynamikkomponente. Das Plugin hat die Aufgabe, die Mehrkörperszene entsprechend den Modellvorgaben zu initialisieren und zur Laufzeit Änderungen zwischen VR-System und Dynamikkomponente in beide Richtungen zu kommunizieren. Durch eine spezielle Struktur von VEROSIM®, aus der eine direkte Beziehung zwischen Softwareklassen und Elementen eines Modells resultiert, kommen auch Anwender der grafischen Benutzeroberfläche, wenn auch unbewusst, direkt in Kontakt mit den Klassen aus diesem Plugin.

Neben den drei Kernbibliotheken existieren einige anwendungsspezifische Bibliotheken, die zusätzliche Logik implementieren und ggf. Funktionen der Dynamiksimulation nutzen. Zwei solche Bibliotheken sind VSPluginRBHarvester und VSPluginRBForwarder. Diese Bibliotheken implementieren zusätzliche Logik für die Anwendungen „Harvestersimulator“ und „Forwardersimulator“ und nutzen dazu Features der Dynamiksimulation z.B. aus dem Bereich der Kontaktgraphenanalyse. Details dieser anwendungsspezifischen Funktionalität liefern die Kapitel 6.1 und 6.2.

5.2. Der zeitliche Ablauf der Simulation

Das eingesetzte VR-System VEROSIM® ist ein modernes und vielseitiges Simulationswerkzeug, das in einer Vielzahl praktischer Anwendungen verwendet wird. Entsprechend der Breite der Anwendungen, vom Einsatz als 4D-Geoinformationssystem (4D-GIS)[1] bis hin zum Werkzeug für Fabriksimulationen, ist die dynamikbasierte Mehrkörpersimulation nur eines von vielen unterstützten Simulationsverfahren. Da unterschiedliche Simulationsverfahren gemeinsam eingesetzt werden, muss sich die zeitliche Steuerung nach den Vorgaben eines Simulations-*Schedulers* [103] richten, den VEROSIM® vorgibt.

Jeder zu simulierende Prozess in VEROSIM® muss sich als *Task* beim Scheduler anmelden. Dieser ruft die einzelnen Tasks, u.a. eben auch die Dynamiksimulation auf und übermittelt die Zeitdifferenz (Δt), um welche die Simulation vorangetrieben werden muss. Für interaktive Anwendungen kommt ein „VisuCycle“ genannter Betriebsmodus des Schedulers zum Einsatz: Der Scheduler startet die Simulation zum Real-Zeitpunkt t_0 und zeichnet initial einmal die Szene. Dabei protokolliert er die vergangene Realzeit.

[1] Die vierte Dimension beschreibt die Zeit: Eingesetzt als GIS bietet VEROSIM® die Möglichkeit, Geoinformationen zu unterschiedlichen Zeitpunkten auszuwerten und darzustellen.

Danach werden die Tasks mit der jeweils vergangenen Realzeit aufgerufen. Ein Zyklus endet mit dem erneuten Zeichnen der Szene. Interaktive Anwendungen erfordern kurze und möglichst konstante Ausführungszeiten der Dynamiksimulation, damit eine gleichmäßig hohe Bildwiederholrate erreicht wird und die Echtzeit eingehalten werden kann. Echtzeitfähigkeit bedeutet, dass die Simulation und Darstellung einer Sekunde Modellzeit nie mehr als eine Sekunde Realzeit erfordern darf. Die Anforderungen an die Ausführungszeit lassen praktisch nur sogenannte Time-Stepping-Verfahren (vgl. Abschnitt 2.2.4) zu, d.h. die Zeitsteuerung erfolgt mit konstanter Schrittweite h. Wird die Dynamiksimulation vom Scheduler mit einer Zeitdifferenz $\Delta t > h$ aufgerufen, werden mehrere Schritte der Länge h ausgeführt. Dabei überschreitet die Summe der internen Zeitschritte nie Δt. Bleibt nach n ganzen Schritten der Länge h eine Restzeit $t_{\mathrm{rest}} = \Delta t - n \cdot h$ übrig, wird dieser Wert bis zum nächsten Aufruf der Dynamiksimulation durch den Scheduler vorgehalten und dann auf das zu simulierende Δt aufgeschlagen.

Es wird bewusst kein Zeitschritt der Länge $t_{\mathrm{rest}} < h$ direkt ausgeführt, da die inverse Dynamik verschiedentlich aufeinanderfolgende Zeitschritte gleicher Länge erwartet, so z.B. bei der Stabilisierung der Zwangsbedingungen (vgl. Kapitel 3.1.5).

Der Ablauf eines Simulationsschrittes der Länge h innerhalb der reinen Mehrkörpersimulation ist damit wie folgt:

1. **Dynamische Rekonfiguration:** Hat es eine Änderung in der Konfiguration der Mehrkörperszene gegeben, das heißt hat sich die Menge der simulierten Körper oder ihre Zusammensetzung im Verlauf der Simulation verändert, wird anhand des Kontaktgraphen die Menge der zu simulierenden Körper neu bestimmt. Details hierzu beschreibt Kapitel 4.1

2. **Kollisionserkennung und Bestimmung von Kontaktpunkten:** Das System unterstützt derzeit geometrische Primitive, Dreiecksnetze, Höhenfelder und Sphere-Trees. Die Kollisionserkennung an Dreiecksnetzen basiert auf dem BVH-Verfahren [55] (siehe Kapitel 2.3.2 ab Seite 29) der VEROSIM®-Bibliothek VSLibCollision.

3. **Auswertung des Kontaktgraphen I/II:** Bestimmung von Pfadabständen für Stoßfortpflanzung und Aktivierungs-Heuristiken.

4. **Auswertung des Kontaktgraphen II/II:** Bestimmung der Simulationscluster. Gibt es in der Szene mehrere unabhängige Gruppen zusammenhängender Körper, die untereinander nicht interagieren, lassen sich diese leicht im Kontaktgraphen

identifizieren. Die Bestimmung der inversen Dynamik wird dann für jeden Simulationscluster einzeln ausgeführt.

5. **Kollisionsbehandlung:** Soll eine explizite Kollisionsbehandlung durchgeführt werden, so muss sie an dieser Stelle erfolgen. Sollen Modelle wie z.B. ein Billardspiel oder Newtons Kugelstoßpendel simuliert werden, ist eine explizite Kollisionsbehandlung notwendig. Für die im Rahmen dieser Arbeit betrachteten Anwendungen spielen Kollisionen im Gegensatz zu ruhenden Kontakten keine Rolle. Auf eine Kollisionsbehandlung wurde daher verzichtet. Sie lässt sich jedoch unabhängig von der inversen Dynamik als eigener Schritt realisieren, z.B. entsprechend einem iterativen Verfahren wie bei Bender [17].

6. **Bestimmung der inversen Dynamik bzw. Berechnung aller Zwangsimpulse:** Dies ist der Schlüsselschritt der Mehrkörperdynamiksimulation. An seinem Ende sind alle im Mehrkörpersystem wirkenden Kräfte und Momente (bzw. Impulse und Drehimpulse) bekannt und die Simulation kann durch Animation der Bewegungsgleichungen in der Zeit vorangetrieben werden.

7. **Geschwindigkeitsupdate:** Update der linearen und rotatorischen Körper-Geschwindigkeiten $\vec{v}(t+h) = \vec{v}(t) + \vec{\Phi}(\vec{f}_{\text{ext}}(t), \vec{i}(t, t+h))$ unter Berücksichtigung aller externer Kräfte $\vec{f}_{\text{ext}}(t)$ zum Zeitpunkt t und aller Zwangsimpulse $h\vec{\lambda}(t, t+h)$, die für die Einhaltung der Zwangsbedingungen im Zeitpunkt $t+h$ sorgen.

8. **Positions- und Orientierungsupdate:** Update von Positionen und Orientierungen der Körper auf Basis der neuen Geschwindigkeiten $\vec{x}(t+h) = \vec{x}(t) + \vec{\rho}(\vec{v}(t+h))$. Hierbei wird außerdem der inverse Trägheitstensor Θ^{-1} in Weltkoordinaten jedes Körpers neu bestimmt, der während der Bestimmung der inversen Dynamik zur Formulierung der inversen Massenmatrix $\mathbf{M}^{-1}$ benötigt wird.

Die oben aufgeführten acht Schritte finden sich in der Methode simulate(...) der zentralen Klasse `RBDScene` der Bibliothek VSLibRBDynamX. Für Entwickler ist dies der zentrale Einstiegspunkt in die vorliegende Implementierung.

5.3. Modellierung dynamikbasierter Simulationen

Soll ein Simulationssystem Grundlage einer Vielzahl von Anwendungen sein, kommt der Modellierung entsprechender Modelle besondere Bedeutung zu. Bei vielen im Rah-

men dieser Arbeit untersuchten Anwendungen besteht die folgende, typische Ausgangssituation: 3D-Geometriedaten, typischerweise importiert aus CAD-Daten, stehen in VEROSIM® zur Verfügung. Die Geometriedaten werden in der VEROSIM®-eigenen Datenbank („VSD“) vorgehalten. Die VSD erlaubt die Anordnung von Geometrie- und anderen Daten in kinematischen Bäumen. Jeder Knoten eines kinematischen Baumes (Instanzen von `VSD3D::Node`) kann eine homogene Transformation (siehe z.B. [23]) seines Koordinatensystems in Relation zu dem seines Elternknotens definieren. Diese Datenstruktur wird in VEROSIM® klassischerweise direkt zur Modellierung kinematischer Ketten bei rein kinematischer Simulation genutzt. Eine kinematische Kette entspricht einem Pfad von der Wurzel bis zu einem Blatt des kinematischen Baumes. Die Anordnung der Daten in Baumform spiegelt somit die unidirektionale Natur einer kinematischen Kette wider: Wird die Basis bewegt, bewegt sich die gesamte Kette rückwirkungsfrei mit ihr. Wird nur das Blattelement bewegt, bleiben die übrigen Elemente der kinematischen Kette hiervon unbeeinflusst. Diese Struktur widerspricht der Natur eines dynamikbasierten Mehrkörpersystems, in dem sich, entsprechend Newtons drittem Axiom „Actio et reactio“, immer alle Körper gegenseitig beeinflussen. Die Modellierung der Elemente eines Mehrkörpersystems geschieht daher nicht hierarchisch. Alle Körper besitzen eine Position und Orientierung in Form einer homogenen Koordinatentransformation im selben Weltkoordinatensystem.

Auch Gelenke können unabhängig voneinander und von Körpern angelegt werden. Ihre eigene homogene Transformation beschreibt implizit ihre Position bzw. die Lage ihrer Achsen zu Beginn der Simulation. So nutzt z.B. das Drehgelenk die z-Achse seines lokalen Koordinatensystems als Drehachse. Die Relationen zu den Körpern, die ein Gelenk verbindet, werden über explizite Referenzen am Gelenk hergestellt. Dabei ergibt sich die Lage des Gelenks zu einem beliebigen Zeitpunkt der Simulation aus den Lagen und Positionen der Körper im Weltkoordinatensystem und den relativen Lagen des Gelenks bzw. seiner Achsen in den Koordinatensystemen der verbundenen Körper. Diese relativen Lagen sind konstant im Verlauf der Simulation.

Die Modellierung dynamikbasierter Modelle folgt einem durch VEROSIM® vorgegebenen Entwurfsmuster. Anstatt für unterschiedliche Arten von Knoten in der VSD unterschiedliche Klassen zu definieren, ergänzen sogenannte *Extensions* (zu Deutsch: Erweiterungen) bestehende Knoten um neue Eigenschaften. Dieser Idee folgend bietet die Softwarebibliothek VSPluginRBDynamX eine Menge von Extensions an, mit denen Knoten der VSD um physikalische Eigenschaften ergänzt werden, zum Beispiel denen

von Starrkörpern und Gelenken. Die folgenden Abschnitte beschreiben detailliert, wie einfach die Modellierung dynamikbasierter Simulationen auf der Grundlage der VSD und den Erweiterungen des Dynamiksimulationssystems möglich ist.

5.3.1. Modellierung eines Starrkörpers

Der erste Schritt bei der Erstellung eines dynamikbasierten Modells ist die Definition eines Körpers. Zunächst benötigt ein Körper ein lokales Koordinatensystem, das seine Position (i.A. nicht sein Schwerpunkt!) und seine Orientierung beschreibt. Diese Informationen kann ein Knoten vom Typ `VSD3D::Node` vorhalten, im Folgenden Körperknoten genannt. Für die Ergänzung eines solchen Knotens um die Eigenschaft „Starrkörper“ im Sinne der Dynamiksimulation wird er mit einer `ExtensionRigidBody` ausgestattet. Bis hierher besitzt er jedoch noch keine Form, keine Masse und keine Trägheit. Diese Informationen ergeben sich aus der Akkumulation physikalischer Grundformen, die im Sinn der VSD-Hierarchie unterhalb des Körperknotens angelegt sind.

Geometrie wird in der VSD durch Instanzen von `VSD3D::HullNode` vorgehalten. Um die reine Geometrie um physikalische Eigenschaften zu ergänzen, wird sie mit einer Instanz von einer von `ExtensionRigidBodyShape` abgeleiteten Klasse erweitert. Diese Extensions erweitern Geometrien zu physikalischen Grundformen, im Folgenden auch *Shape* genannt. Typischerweise werden Geometrien mit dazu passenden Shape-Klassen ergänzt, z.B. ergänzt eine `ExtensionRigidBodyShapeCylinder` eine Zylinder-Geometrie. Dieses Verhältnis ist jedoch nicht verbindlich: Shape-Klassen können selbst alle für die Simulation erforderlichen Informationen vorhalten. Die Trennung von visualisierter und physikalischer Geometrie dient dazu, hoch detaillierte Modelle der Visualisierung durch einfachere und aus Sicht der Kollisionserkennung rechenzeiteffizientere Geometrien zu approximieren. Kapitel 6 liefert einige Beispiele für die Simulation mit vereinfachten physikalischen Ersatzmodellen.

Für jeden Shape-Typen steht eine eigene Klasse zur Verfügung. Abbildung 5.1 zeigt eine VEROSIM®-Exploreransicht[2] der Datenbank und die zugehörige 3D-Visualisierung eines Starrkörpers bestehend aus zwei physikalischen Grundformen, einem Box-Typ und einem Zylinder-Typ. Im Beispiel stimmen Visualisierung und physikalisches Modell überein.

Zur Modellierung von Masse und Trägheitstensor eines Starrkörpers sind zwei Wege

[2]Der VEROSIM®-Explorer ist ein dem Windows-Explorer verwandtes grafisches Oberflächenelement zur Bearbeitung der VSD

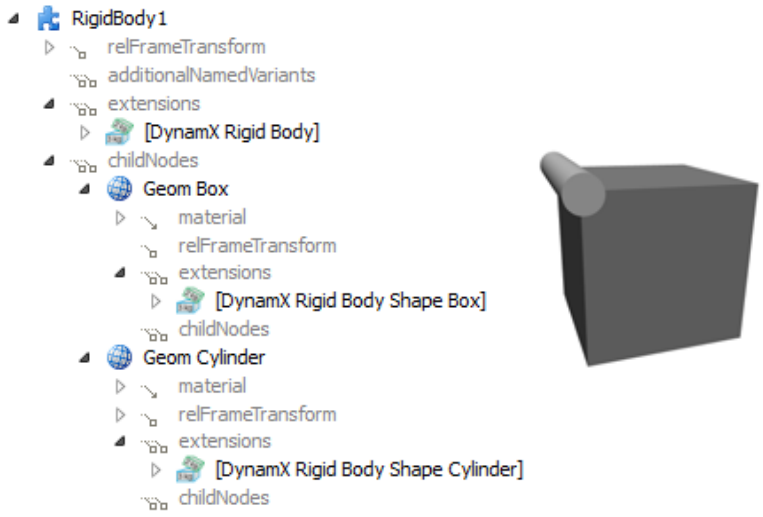


Abbildung 5.1.: Modellierung eines einzelnen Starrkörpers bestehend aus zwei Shapes. Links eine Ansicht der Datenbank im Explorer, rechts die zugehörige Visualisierung.

vorgesehen: Beide Eigenschaften können explizit an der `ExtensionRigidBody` hinterlegt werden. In diesem Fall dient das Koordinatensystem des übergeordneten Knotens als Referenzsystem für diesen Körper, sein Ursprung ist dann der Schwerpunkt. Für die Prototypenentwicklung ist es hilfreich, wenn diese Eigenschaften automatisch aus der Konfiguration der zugehörigen Grundformen abgeleitet werden. Jede Grundform besitzt eine Masse und einen definierten Schwerpunkt. Sein Trägheitstensor wird automatisch unter Annahme einer homogenen Massenverteilung berechnet. Gesamtmasse und -trägheitstensor und Gesamtschwerpunkt werden dann aus den jeweiligen Teilgrößen akkumuliert. Die Akkumulation von Trägheitstensoren behandelt Anhang A.1.2.

5.3.2. Modellierung von Gelenken

Der nächste Schritt ist die Modellierung eines Drehgelenks zwischen zwei Starrkörpern. Abbildung 5.2 zeigt jetzt zwei Starrkörper, die über ein Gelenk miteinander verbunden sind. Das linke Bild zeigt eine Ansicht der Datenbank, das rechte die zugehörige 3D-Visualisierung. Der zweite Starrkörper (oben, „DynamX Rigid Body 2“) wurde analog zum ersten modelliert, besteht jedoch nur aus einem einzelnen Quader. Beide Körper liegen auf derselben Hierarchieebene in der VSD - dies spiegelt die bidirektionale Natur

ihrer Interaktion wider. Sie beeinflussen sich demnach gegenseitig.

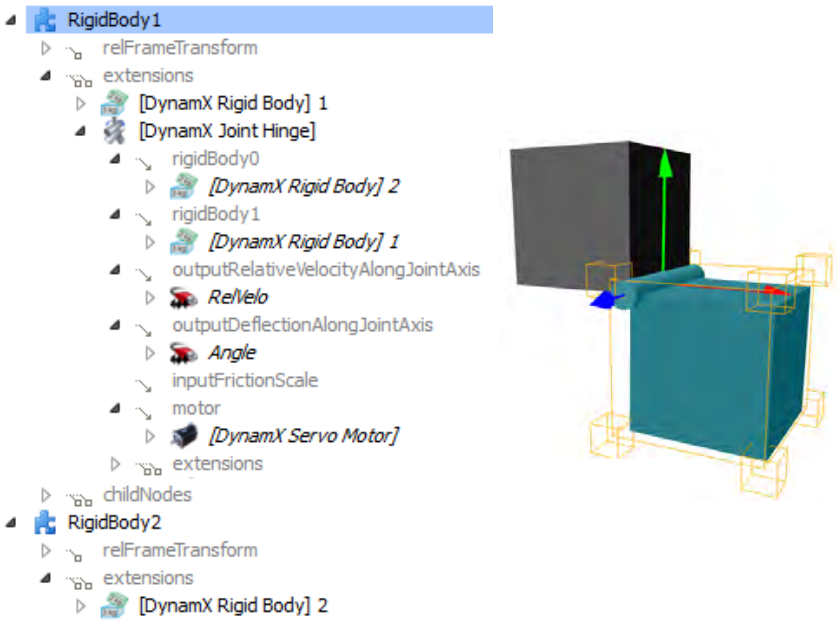


Abbildung 5.2.: Modellierung eines Drehgelenks zwischen zwei Starrkörpern. Das Drehgelenk referenziert die Körper, die es verbindet, einen Motor, der es antreibt und zwei Ausgänge, über die kinematische Sensorwerte des Gelenks ausgegeben werden.

Der Knoten des ersten Starrkörpers („DynamX Rigid Body 1“) wird mit der Extension „[DynamX Joint Hinge]“ vom Typ `ExtensionJointHinge` erweitert. Diese Erweiterung verleiht ihm die Eigenschaft, als Gelenk zu fungieren. Das Gelenk referenziert die beiden Körper, die es verbindet (Referenzen „rigidBody0“ und „rigidBody1“). Initiale Orientierung und Position des Gelenks (vgl. Kapitel 3.2.1) definiert das Koordinatensystem des Knotens, zu dem es gehört: Sein Ursprung ist die Position, die z-Achse (blau) wird zur Drehachse des Gelenks. Im Verlauf der Simulation ergeben sich Position und Orientierung des Gelenks aus ihrer initialen, relativen Lage zu den verbundenen Körpern sowie deren absoluter Lage und Position. Im Beispiel nutzen die Extensions „[DynamX Joint Hinge]“ und „[DynamX Rigid Body] 1“ denselben Knoten „RigidBody1“ als Koordinatensystem. Dies ist nur eine Möglichkeit der Modellierung. Soll die nicht-hierarchische Natur des Mehrkörpersystems deutlicher hervorgehoben werden oder kann das Koordinatensystem eines Körperknotens nicht entsprechend ausgerichtet werden, so kann die Gelenk-Extension auch an einem beliebigen anderen Knoten hängen, der dann frei

Gelenktyp	**Klassenname**	**Freiheitsgrade**
Drehgelenk	ExtensionJointHinge	rot. (z)
Lineargelenk	ExtensionJoint-Prismatic	transl. (z)
Kardangelenk	ExtensionJoint-Universal	rot. (z) rot. (x)
Kugelgelenk	ExtensionJointBall-InSocket	rot. (x) rot. (y) rot. (z)
Schraubengelenk	ExtensionJointScrew	rot./transl. (z)
Gedämpftes Drehgelenk	ExtensionJointWheel-Suspension	rot. (z) transl. fd (x)
Gedämpftes, gelenktes Drehgelenk	ExtensionJoint-SteeredWheel-Suspension	rot. (z) rot. (x) transl. fd (x)
Dämpfer linear	ExtensionJoint-PrismaticSpring-Damper	transl. fd (z)

Tabelle 5.1.: Übersicht über verfügbare Gelenktypen, die jeweils als Freiheitsgrade genutzten Achsen (**rot**atorisch/**transl**atorisch, x, y, z) des Koordinatensystems und ggf. die Information, welche Achse als Feder-Dämpfer-System (fd) ausgelegt ist

ausgerichtet werden kann. Unter pragmatischen Gesichtspunkten bietet es sich allerdings häufig an, für Gelenk und Körper denselben Knoten zu nutzen.

Andere Gelenktypen werden analog zum Drehgelenk modelliert. Sie nutzen als Achsen ggf. mehr oder andere Basisvektoren ihres jeweiligen Koordinatensystems. Neben den frei beweglichen Achsen bietet das System auch Gelenke mit solchen Achsen, entlang derer die Bewegung einem Feder-Dämpfer-Modell entspricht. Tabelle 5.1 liefert eine Auflistung der meist genutzten Gelenktypen und der jeweils genutzten Basisvektoren zur Definition der Freiheitsgrade. Es lassen sich viele weitere, auch abstrakte Gelenktypen, wie z.B. ein „Ebenengelenk“ realisieren - für die hier betrachteten Anwendungen bestand jedoch keine entsprechende Anforderung.

Über Ausgänge (`VSDIO::Output`) werden kinematische Sensorwerte bereitgestellt, die an Gelenken oder Körpern während der Simulation ohnehin bestimmt und häufig benötigt werden. Über das Input/Output-Framework in VEROSIM® können diese Werte sehr einfach und direkt anderen Simulationskomponenten zur Verfügung gestellt werden.

5.3.3. Modellierung von Antrieben

Das modellierte Gelenk soll nun noch einen Antrieb erhalten, um eine Bewegung erzwingen zu können. Hierzu stehen während der Modellierung drei Motortypen zur Verfügung: Der kraftbasierte Motor („DynamX Force Based Motor“, Klasse `ExtensionForceBasedMotor`) wendet im Mehrkörpersystem explizit ein vorgegebenes Drehmoment bzw. eine vorgegebene Kraft auf. Diese Kräfte oder Momente sind dann Teil des Vektors $\vec{f}$ der wirkenden Kräfte, vgl. Gleichung 3.21 auf Seite 86. Die geschwindigkeits- und positionsgestellten Motoren („DynamX Velocity Based Motor“, Klasse `ExtensionVelocityBasedMotor` und „DynamX Servo Motor“, Klasse `ExtensionServoMotor`) werden beide durch Einbringung kinematischer Zwangsbedingungen realisiert. Während der geschwindigkeitsbasierte Motor direkt eine Sollgeschwindigkeit als rechte Seite b der Zwangsbedingung nutzt, verwendet der Positionssteller eine PID-Regelung, um aus der Abweichung zwischen Soll- und Istwert der Auslenkung eines Gelenks eine Sollgeschwindigkeit zu bestimmen und sie als rechte Seite der Zwangsbedingung einzusetzen. Entsprechende Parameter sind am Motor anzugeben. Die Realisierung von Motoren durch Zwangsbedingungen behandelt Kapitel 3.2.8. Ist eine Motorinstanz angelegt, muss sie lediglich am Gelenk referenziert werden, um aus einem freilaufenden ein angetriebenes Gelenk zu machen. Am Motor selbst können alle notwendigen Eigenschaften des Antriebs vorgenommen werden, wie zum Beispiel das maximal aufzubringende Moment, die Referenzierung eines Sollwerteingangs, Skalierungsfaktoren usw.. Gelenktypen, die mehr als einen Motor unterstützen können, zum Beispiel das Gelenk für die kombinierte Vorderradaufhängung (Klasse `ExtensionSteeredCarWheelSuspension`, vgl. Kapitel 3.4.4), können auch mehrere, unabhängige Motoren referenzieren.

Dasselbe Modellierkonzept wird auf jedes beliebige Modell angewendet. Abbildung 5.3 zeigt einen Screenshot von der Ausmodellierung der dynamischen Komponenten eines Laufroboterbeins.

5.3.4. Modellierung von Differenzialgetrieben

Differenzialgetriebe beziehen sich auf mehrere Gelenke und kommen in verschiedenen Anwendungen zum Einsatz. Im Folgenden werden zwei typische Anwendungsszenarien beschrieben.

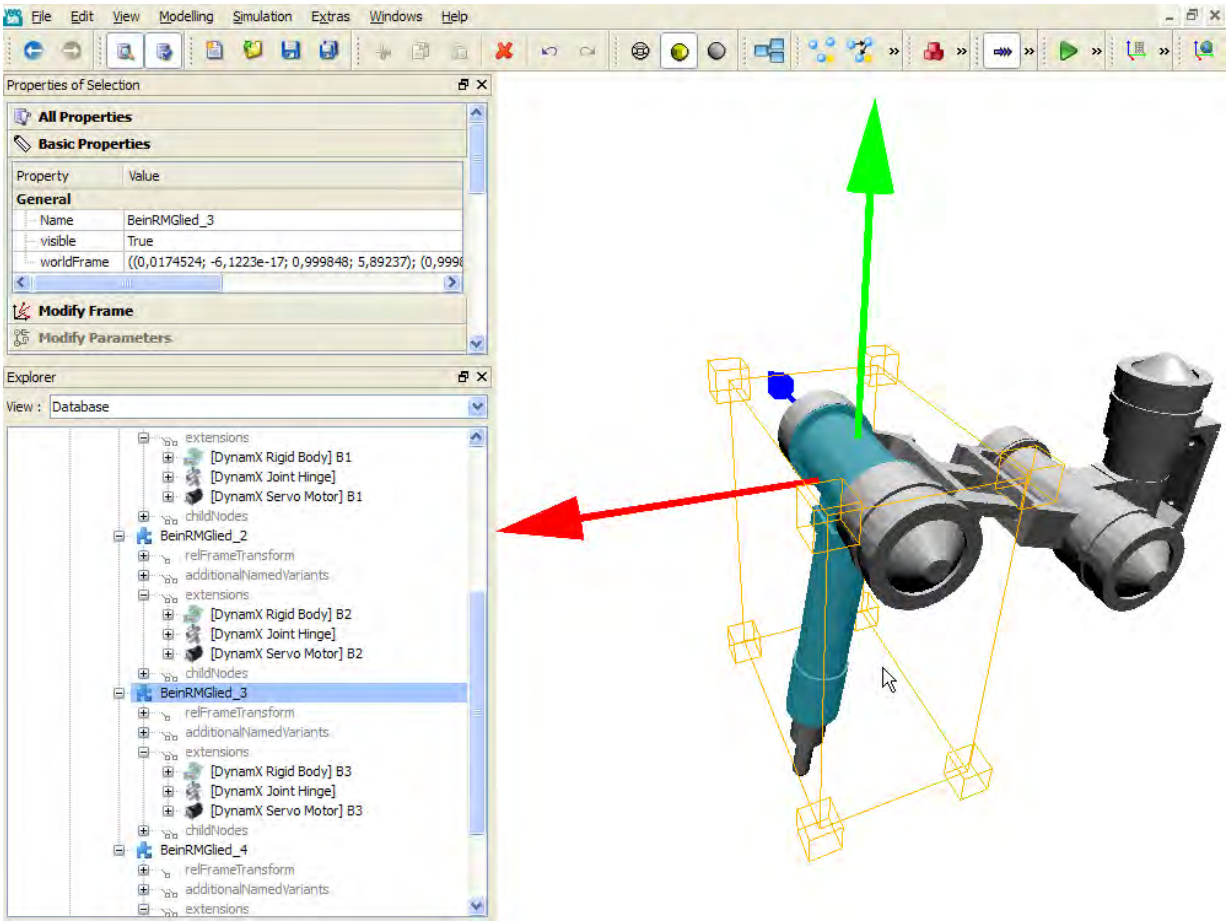


Abbildung 5.3.: Ausmodellierung eines Laufroboterbeins mit dem vorgestellten Modellierkonzept in VEROSIM®

Direkte Kopplung zweier Drehgelenke

Die erste Anwendung eines Differenzialgetriebes behandelt die direkte Kopplung zweier Drehgelenke. Ein typisches Anwendungsbeispiel zeigt Abbildung 5.4, rechts: Zu sehen ist der Greifer eines Holz-Forwarders. Die beiden gelben Zangen sind je über ein Drehgelenk mit der grauen Basis verbunden. In Realität sorgt eine mechanische Kopplung dafür, dass linke und rechte Zange immer die jeweils gleiche Auslenkung zur Basis haben.

In der Simulation wird dies durch ein Differenzialgetriebe nachgebildet, Abbildung 5.4 links zeigt die Exploreransicht der Datenbank. Wie im vorherigen Beispiel sind linke und rechte Zange je über ein Drehgelenk („DynamX Joint Hinge“) mit der Basis verbunden. Nur das Gelenk der großen Zange (links) verweist auf einen Antriebsmotor. An

der kleinen Zange findet sich eine Instanz von „DynamX Joint Differential“, die auf beide Drehgelenke verweist. Die Eigenschaft „transmissionRatio“ (Übersetzungsverhältnis) des Differenzials hat den Wert 1,0 oder -1,0, je nach Orientierung der Drehgelenke. So wird die kleine Zange immer analog zur großen Zange mitbewegt. Der Motor an der großen Zange muss durch die zusätzliche differenzielle Relation nun auch die kleine Zange mitbewegen. Sein maximales Moment muss also für die Bewegung beider Zangen ausreichen.

Bei dieser Anwendung handelt es sich um ein positionsbezogenes Differenzial: Tatsächlich sollen nicht nur die Geschwindigkeiten der beiden Zangen relativ zur Basis gleich groß sein, sondern auch ihre Auslenkungen. Daher ist bei diesem Differenzial die Korrektur eines potenziellen Fehlers notwendig. Um sie zu aktivieren, besitzen alle von `ExtensionJointDifferential` abgeleiteten Klassen die Eigenschaft „correctPositionalConstraint“. Wird diese Eigenschaft „true“ gesetzt, sorgt die Fehlerkorrektur am Differenzial (vgl. Kapitel 3.4.2) dafür, dass nicht nur die geschwindigkeitsbezogene, sondern auch die positionsbezogene Zwangsbedingung eingehalten wird.

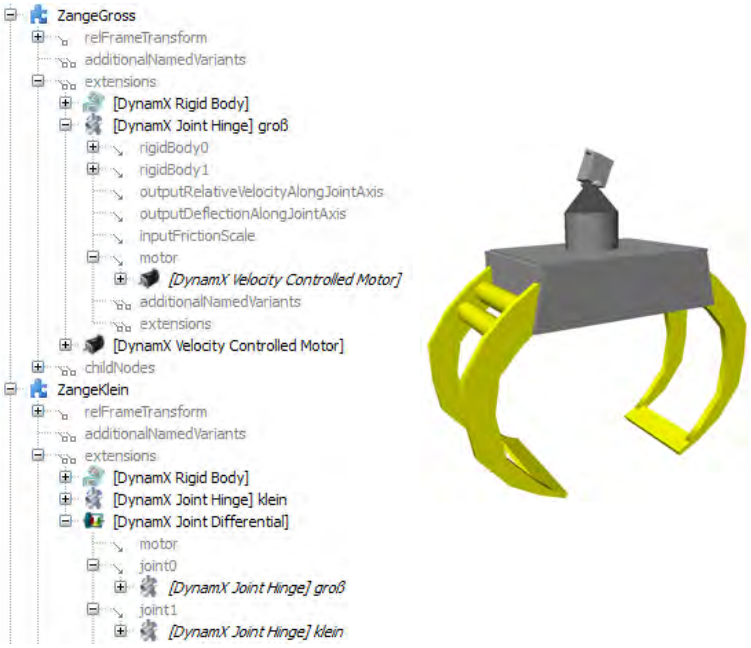


Abbildung 5.4.: Modellierung eines Differenzialgelenks zur direkten Kopplung zweier Drehgelenke.

Radantrieb über freilaufendes Querdifferenzial

Abbildung 5.5 zeigt einen Ausschnitt der Exploreransicht eines PKW-Modells. Der Ausschnitt stellt nur die beiden Differenziale für vorne und hinten dar. Jedes Differenzial verbindet ebenfalls je zwei Drehgelenke links und rechts. Statt eines Motors an einem der Drehgelenke, wie in den vorhergehenden Beispielen, sind die Gelenke jetzt ohne Antrieb, dafür besitzt jedes Differenzial einen eigenen Motor. Diese Modellierung bildet ein mechanisches, freilaufendes Querdifferenzial nach. Die Zwangsbedingung, die durch das Differenzial impliziert wird, lautet: Die Summe der Geschwindigkeiten von linkem und rechtem Rad ist gleich der Geschwindigkeit des Motors. Als Übersetzungsverhältnis am Differenzial wird wiederum 1,0 bzw. -1,0 angesetzt.

Hier kann auf die Korrektur des positionsbezogenen Fehlers verzichtet werden. Da die Räder eines Autos in ihrer Auslenkung um die Rotationsachse nicht begrenzt sind, fallen kleine Abweichungen zwischen den Auslenkungswinkeln nicht auf. Die Eigenschaft „correctPositionalConstraint“ kann daher auf „false“ gesetzt werden.

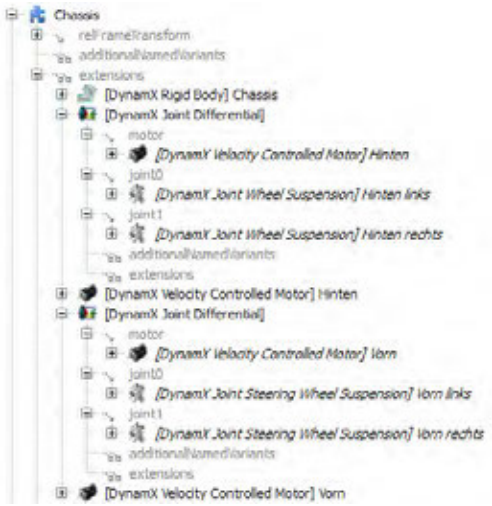


Abbildung 5.5.: VEROSIM®-Exploreransicht der Modellierung von zwei angetriebenen Querdifferenzialen in einem PKW-Modell.

Weitere Anwendungen von Differenzialen

Viele weitere Anwendungen sind möglich. So können mithilfe der Eigenschaft „transmissionRatio“ auch beliebige Getriebeübersetzungen simuliert werden. Auch kann die Klasse „DynamX Joint Differential“ unterschiedliche Gelenktypen referenzieren, bei-

spielsweise ein Linear- und ein Drehgelenk. Mit dieser Konstruktion lässt sich die Umsetzung von rotatorischer auf lineare Bewegung und umgekehrt simulieren, dank der direkten Abbildung auf Zwangsbedingungen (vgl. Kapitel 3.4.2) immer unter korrekter Berücksichtigung der dynamischen Zusammenhänge.

5.3.5. Modellierung dynamikbezogener Sensoren

Zwangsimpulse, die während der Dynamiksimulation bestimmt werden, können auch für andere Anwendungen neben der reinen Bewegungsanimation interessant sein. Da sie zur Simulation ohnehin bestimmt werden müssen, können sie anderen Simulationskomponenten ohne zusätzlichen Rechenaufwand zur Verfügung gestellt werden. Hierzu wurden für die Mehrkörperdynamiksimulation in VEROSIM® in der Bibliothek VSPluginSensorSimRBDynamX einige Sensorsimulationen implementiert, die diese Größen einfach und flexibel verfügbar machen.

Der Motorkraftsensor (`ExtensionMotorForceTorque`, Abbildung 5.6, links) erweitert einen geschwindigkeitsbasierten Motor und liefert dessen aktuelles Drehmoment bzw. dessen aktuelle Kraft. Der Kontaktkraftsensor (`ExtensionContactForce`, Abbildung 5.6, mitte) erweitert einen Starrkörper und liefert die Summe aller Kontaktkräfte, die gegenwärtig auf ihn einwirken. Der Kraft-/Momentsensor (`ExtensionForceTorque6DOF`, Abbildung 5.6, rechts) schließlich erweitert eine starre Verbindung (`ExtensionJointRigid`) zwischen zwei Starrkörpern und liefert an sechs Ausgängen drei Kraft- und drei Momentwerte, die zur Aufrechterhaltung der starren Verbindung notwendig sind. Selbstverständlich kann dieser Sensor nur dann Kraft- und Momentwerte liefern, wenn die starr verbundenen Körper ***nicht*** durch die Dynamische Rekonfiguration (vgl. Kapitel 4.1) verschmolzen werden. Dazu bietet die Klasse (`ExtensionJointRigid`) die Eigenschaft „forbidUnification“, die diesen Vorgang unterbindet.

Die Verbindung von Sensorsimulationen und Visualisierungsmetaphern liefert spektakuläre Darstellungen: Abbildung 5.7 zeigt ein Modell des DFKI [2] Scarabaeus Laufroboters. Die Ausgaben der Motorkraftsensoren werden genutzt, um die entsprechenden Motoren je nach gegenwärtiger Last einzufärben. Analog dazu repräsentiert die Farbe der Füße die gerade auf sie wirkenden Kontaktkräfte.

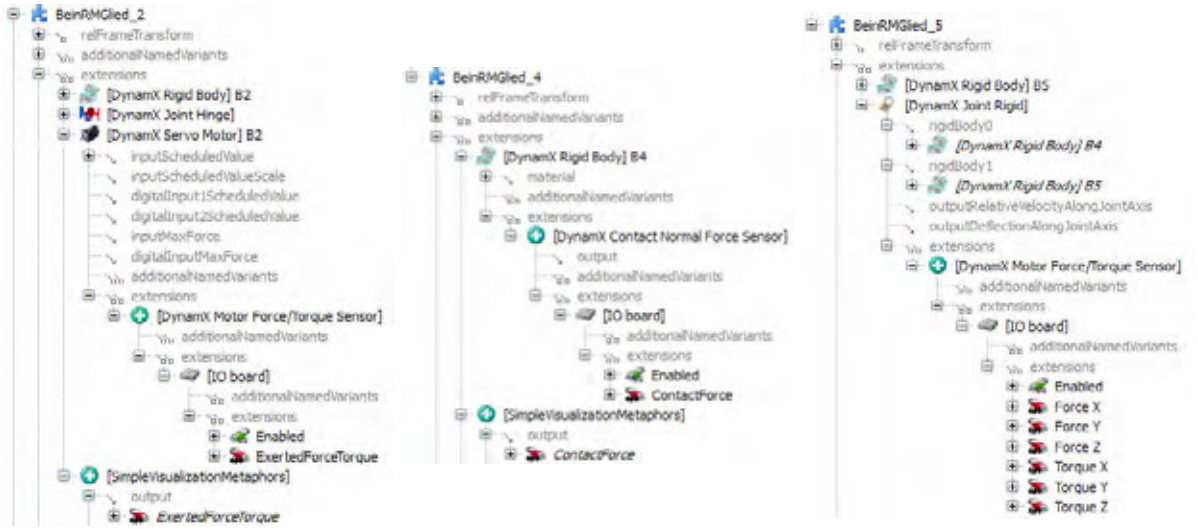


Abbildung 5.6.: Anordnung von Kraft-/Momentsensoren in der VEROSIM®-Datenbank, v.l.: Motordrehmoment, Kontaktkraft, 6-DOF Kraft-/Momentsensor

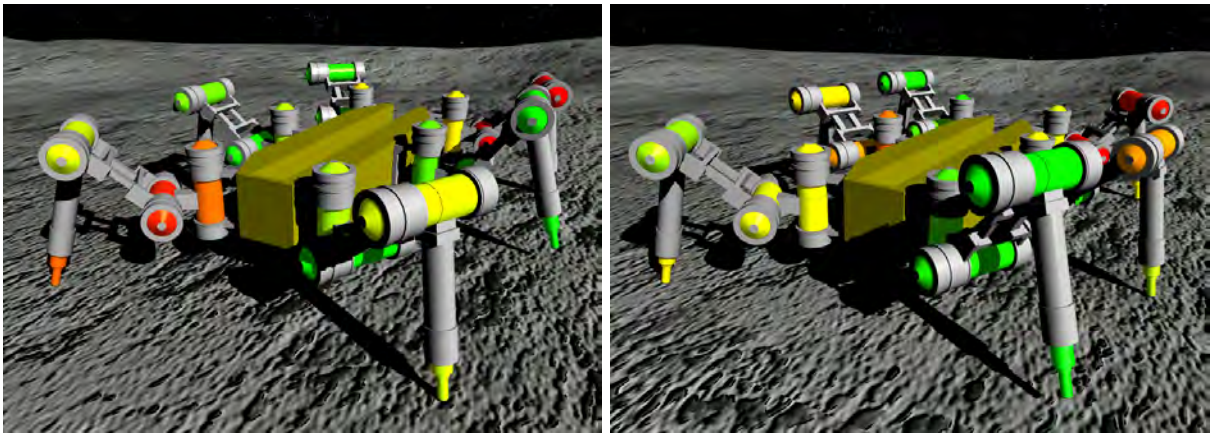


Abbildung 5.7.: Farbliche Kodierung der auftretenden Motormomente und Fußkontaktkräfte am Modell des Scarabaeus Laufroboters des DFKI [2]

5.4. Objektorientierte Softwarestrukturen des Systems

Im Rahmen der Realisierung des Mehrkörperdynamiksimulationssystems in VEROSIM® ist mit VSLibRBDynamX eine umfassende Softwarebibliothek entstanden. Die folgenden Abschnitte erläutern einige grundlegende Strukturen und Konzepte dieser Bibliothek.

5.4.1. Klassenstruktur für die Kollisionserkennung

Abbildung 5.8 zeigt den Ausschnitt des Klassendiagramms, der die wichtigsten Klassen und Zusammenhänge für den Bereich Kollisionserkennung darstellt.

Ausgangspunkt der Kollisionserkennung sind Kollisionselemente und damit Instanzen der Klasse `RBDCollisionDetectionElement` und ein Grobphasenalgorithmus, seinerseits eine Instanz der Klasse `RBDBroadPhaseCD`, zum Beispiel ein gleichmäßiges Gitter (Klasse `RBDCollisionGrid`). Natürlich verweist ein Grobphasenalgorithmus auf mehrere Kollisionselemente. Jedes Kollisionselement kann jedoch auch mehreren Grobphasenalgorithmen zugeordnet werden. Diese Möglichkeit nutzt z.B. die Anwendung „Harvestersimulator“ zur Reduzierung des relevanten Umgebungsmodells, vgl. Kapitel 6.1.1. Das Kollisionselement deklariert alle notwendigen Methoden für einen Grobphasenalgorithmus, zum Beispiel zur Abfrage einer achsenparallelen Bounding Box (AABB) und triggert ggf. die Aktualisierung der Grobphasenalgorithmeninstanzen, denen es zugeordnet ist.

Drei Klassen sind vom Kollisionselement abgeleitet: Das `RBDSceneElement6DOF` beschreibt ein Element der Simulationsszene, welches eine Masse, einen Trägheitstensor, eine Position und eine Orientierung besitzt. Hierunter fallen die Starrkörper selbst (Instanzen von `RBDRigidBody`) sowie die physikalischen Grundformen der Körper (Instanzen von `RBDShape`). Während der Modellierung werden Starrkörper aus physikalischen Grundformen zusammengesetzt. Zur Laufzeit kann ein Starrkörper jedoch auch andere, ebenfalls zusammengesetzte Starrkörper enthalten. Damit erfüllen die drei Klassen `RBDSceneElement6DOF` („Component“), `RBDRigidBody` („Composite“) und `RBDShape` („Leaf“) das klassische Entwurfsmuster „Composite“ [51]. Diese Struktur ist eine der Grundlagen zur Implementierung des Mechanismus der Dynamischen Rekonfiguration aus Kapitel 4.1. Sollen mehrere Starrkörper als ein logischer Körper simuliert werden, wird zunächst eine neue Instanz von `RBDRigidBody` erzeugt, die die logische Vereinigung repräsentiert. Dann werden die zu vereinigenden Körper der neuen Instanz als

Kinder hinzugefügt. Aus Sicht der Mehrkörperdynamiksimulation wird dann nur noch die neu erzeugte Instanz simuliert, die ursprünglichen Körper werden durch ihr Eltern-Element lediglich „mitbewegt“. Aus Sicht der Kollisionserkennung enthält die logische Vereinigung alle Geometrien ihrer Kind-Elemente, unabhängig davon, ob es sich um Starrkörper oder nur um physikalische Grundformen handelt.

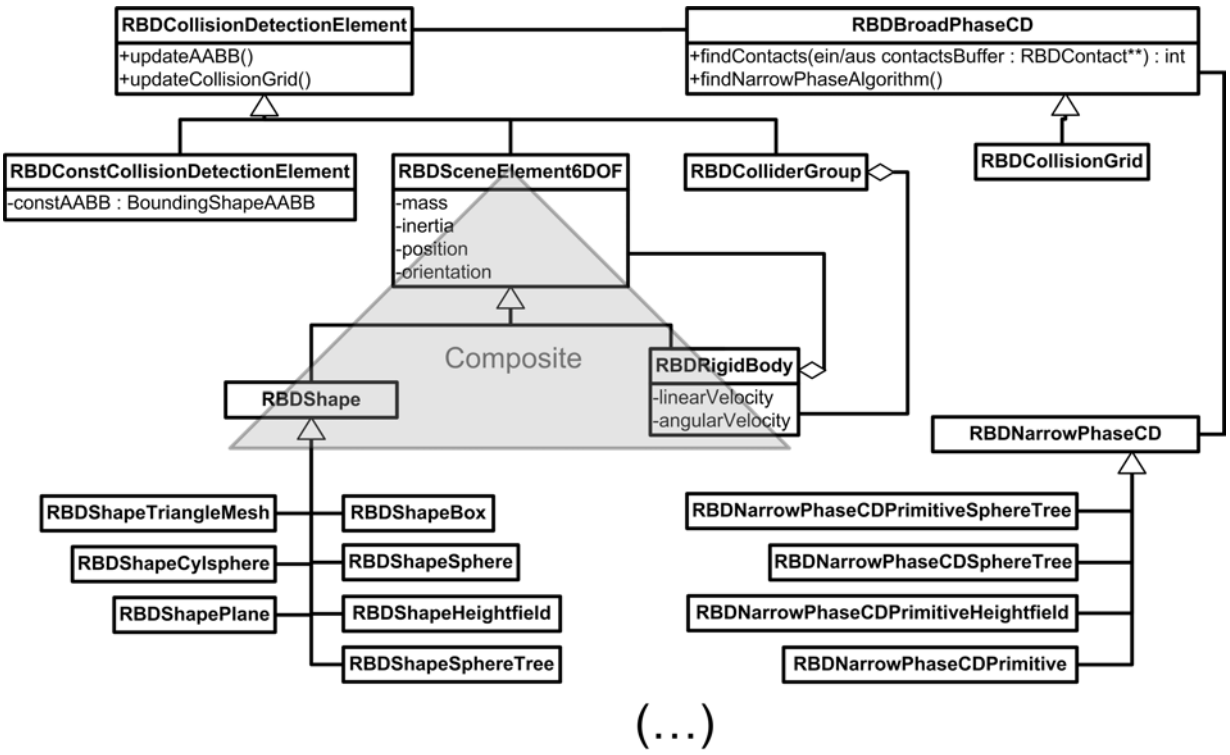


Abbildung 5.8.: Ausschnitt des Klassendiagramms mit Fokus auf die relevanten Elemente für die Kollisionserkennung

Instanzen der Klasse `RBDColliderGroup` gruppieren solche Starrkörper, die zu einer kinematischen Struktur gehören, wie einem Roboterarm oder z.B. Chassis und Räder eines Fahrzeugs, so dass diese nicht untereinander auf Kollision hin überprüft werden. Selbstverständlich hat der Anwendungsentwickler Einfluss auf diese Gruppierung: Wenn Selbstkollisionen erwünscht sind, kann diese Gruppierung unterbunden werden. An der VEROSIM®-Benutzeroberfläche bietet jede Instanz eines Gelenks hierzu die Eigenschaft „separateColliderGroups“.

Das `RBDConstCollisionDetectionElement` dient der einmaligen Abfrage von Kollisionskandidaten zu einer konstanten, achsenparallelen *Bounding Box* und kommt z.B. in der Anwendung Harvestersimulator zum Einsatz.

Die Detailphase der Kollisionserkennung übernehmen Instanzen der von RBDNarrowPhaseCD abgeleiteten Klassen. Jede dieser Klassen bietet Algorithmen für die Detailphase der Kollisionserkennung zwischen bestimmten Typen physikalischer Grundformen, z.B. zwischen geometrischen Primitiven oder zwischen Sphere-Tree und Heighfield. Im abstrakten Grobphasenalgorithmus (RBDBroadPhaseCD) liefert die Methode „findNarrowPhaseAlgorithm“ den jeweils passenden Detailphasen-Algorithmus für ein auf Kontakte hin zu untersuchendes Paar von Grundformen (RBDShape).

Der Großteil der Kollisionskörper in den bis heute realisierten Anwendungen basiert auf geometrischen Primitiven, also Boxen, Kugeln und Zylindern mit Halbkugelkappen, genannt „Cylspheres“. Die Detailphase der Kollisionserkennung an solchen Primitiven ist relativ schnell und robust. Eine Erweiterung um weitere geometrische Grundkörper ist jedoch relativ aufwändig, da für jede mögliche Kollisionspaarung eine eigene Routine entwickelt werden muss. Auch die Entwicklung einzelner Routinen ist durchaus nicht trivial: Während für die o.g. drei Grundtypen noch relativ einfache Ansätze zur Bestimmung von Kontaktpunkten gefunden werden können, ist z.B. die Bestimmung von Kontaktpunkten, Kontaktnormalen und Durchdringungstiefe zwischen zwei beliebig orientierten Zylindern kein triviales Problem.

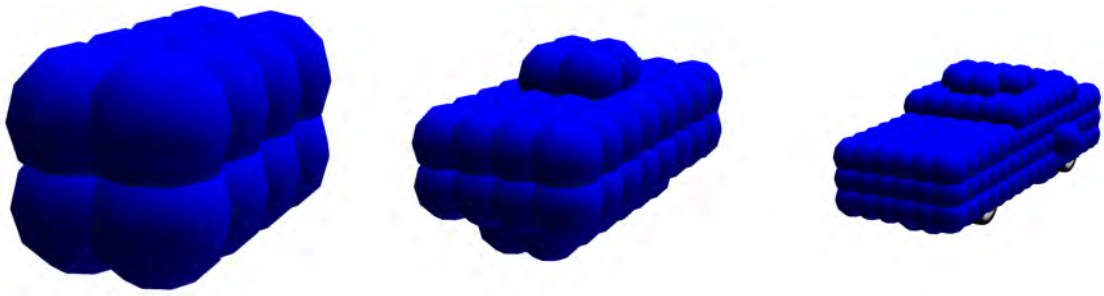
Für beliebige Polyeder steht daher eine einfache Variante der Sphere-Trees (Klasse ExtensionRigidBodyShapeOctSphereTree) zur Verfügung. Hiermit werden Polyeder durch einen hierarchischen Baum aus Kugeln approximiert, wodurch eine relativ einfache aber effiziente Kontaktpunktbestimmung möglich wird. Das Verfahren entspricht dem von O’Sullivan und Dingliana [98].

Beispiele für Modelle aus geometrischen Primitiven finden sich in Kapitel 6. Abbildung 5.9 zeigt ein Beispiel der Approximation einer Karosserie durch einen Sphere-Tree. Die Eigenschaft LOD (engl.: *Level of detail*) beschreibt jeweils die Tiefe des Sphere-Trees.

5.4.2. Klassenstruktur zur Einbringung von Zwangsbedingungen

Abbildung 5.10 zeigt den Ausschnitt des Klassendiagramms, der alle für die Einbringung von Zwangsbedingungen relevanten Klassen und Zusammenhänge darstellt.

Ausgangspunkt hierfür ist die Gesamtszene (Instanz von RBDScene), die alle Starrkörper (RBDRigidBody) und alle Quellen für Zwangsbedingungen (Instanzen der Klasse RBDConstraintResource) enthält. Anhand der Zusammenhänge im Kontaktgraphen werden zur Laufzeit Simulationscluster (RBDCluster) erzeugt. Jedem Simulationscluster



LOD=2 LOD=3 LOD=4

LOD=5 LOD=6 LOD=7

Abbildung 5.9.: Approximation des Polyeder-Modells einer Fahrzeugkarosserie durch einen *Oct-Spheretree* für die Kontaktpunktberechnung bis zur Baumtiefe LOD=7

werden die ihm zugehörigen Starrkörper und Zwangsbedingungen zugeordnet.

Im Falle eines Clusters, der mit dem Lagrange-Multiplikatoren-Verfahren arbeitet (`RBDClusterLagrangeMultiplier`), muss hierin die Jacobimatrix der Zwangsbedingungen formuliert werden. Hierzu deklariert die Klasse `RBDConstraintResource` die beiden Methoden addEqualityConstraints(...) und addComplementaryConstraints(...). Innerhalb dieser Methoden fügt jede `RBDConstraintResource` individuell Zeilen zur Jacobimatrix und zu den zugehörigen rechten Seiten der Gleichungen (Vektor $\vec{b}$) hinzu. In addComplementaryConstraints(...) werden außerdem Grenzwerte für die aufzubringenden Zwangsimpulse ($\vec{\lambda}_{\mathrm{low}}, \vec{\lambda}_{\mathrm{high}}$) definiert. Die Unterteilung in *Equality*- und *Complementary*-Zwangsbedingungen ist erforderlich, da die eingesetzten Lösungsroutinen annehmen, dass die vorderen Zeilen der Jacobimatrix die kraftmäßig unbeschränkten, also Equality-Constraints enthalten und die hinteren Zeilen die kraftmäßig beschränkten, also Complementary-Constraints.

Alle Klassen, welche die Einbringung von Zwangsbedingungen erlauben, sind von `RBDConstraintResource` abgeleitet:

- `RBDJoint` ist die Oberklasse aller „klassischen“ Zweikörpergelenke wie Dreh-

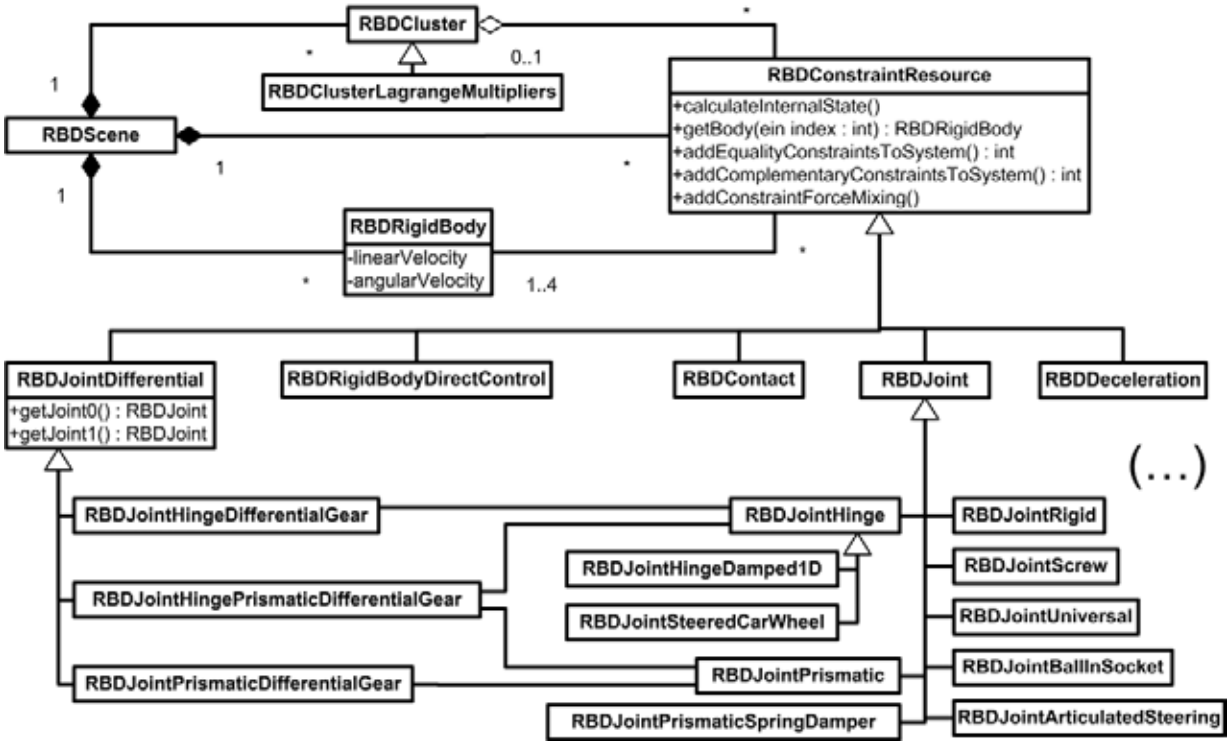


Abbildung 5.10.: Ausschnitt des Klassendiagramms mit Fokus auf die relevanten Elemente für die Einbringung von Zwangsbedingungen

gelenk (`RBDJointHinge`), Lineargelenk (`RBDJointPrismatic`), Kardangelenk (`RBDJointUniversal`) oder Kugelgelenk (`RBDJointBallInSocket`). Vom Drehgelenk existieren zwei weitere Spezialisierungen: Der `RBDJointHingeDamped1D` fügt dem Drehgelenk orthogonal zur Drehachse eine lineare, Feder-Dämpfer-Achse hinzu. Dieses Gelenk kommt als Hinterradaufhängung in Fahrzeugsimulationen zum Einsatz. Der `RBDJointSteeredCarWheel` ist die Klasse der in Kapitel 3.4.4 beschriebenen integrierten Vorderradaufhängung.

- Instanzen von `RBDContact` repräsentieren einen ruhenden Kontakt.
- `RBDJointDifferential` ist die Oberklasse aller Differenzialgelenke, z.B. zwischen zwei Drehgelenken (`RBDJointHingeDifferential`), zwischen zwei Lineargelenken (`RBDJointPrismatic`) oder zwischen einem Linear- und einem Drehgelenk (`RBDJointHingePrismaticDifferential`).
- `RBDRigidBodyDirectControl` dient der direkten Kontrolle eines Starrkörpers in allen sechs Freiheitsgraden, z.B. mittels einer 3D-Maus oder eines Trackingsy-

stems. Anwendungsbeispiele liefert Kapitel 6.4.2.

5.5. Laufzeitoptimale Ausführung notwendiger Matrixmultiplikationen

Die Formulierung für die inverse Dynamik in Gleichung 3.21 auf Seite 86 enthält einige große Matrixmultiplikationen. Die betroffenen Matrizen besitzen spezielle dünn besetzte Strukturen, die sich während ihrer Multiplikation zur Optimierung der notwendigen Rechenzeit ausnutzen lassen.

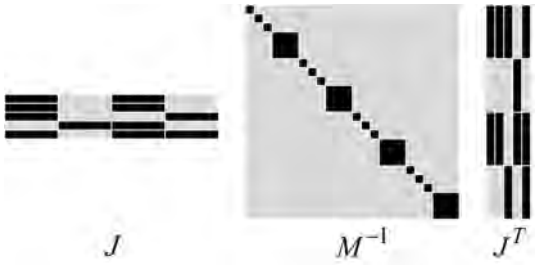


Abbildung 5.11.: Struktur der dünnbesetzten Matrizen $\mathbf{J}$, $\mathbf{M}^{-1}$ und $\mathbf{J}^T$. Nur die schwarz eingefärbten Bereiche enthalten Werte ungleich Null.

Den größten Rechenaufwand unter den Matrixmultiplikationen erfordert die Bestimmung der Systemmatrix $\mathbf{A} = \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$. Abbildung 5.11 zeigt eine grafische Darstellung dieses Produkts. Die Jacobimatrix der Zwangsbedingungen $\mathbf{J}$ enthält eine Zeile für jede Zwangsbedingung. Jede Zeile enthält für jeden Körper, auf den sie sich bezieht, je einen Eintrag mit maximal sechs von Null verschiedenen Werten. Die Massenmatrix $\mathbf{M}$ enthält entlang ihrer Hauptdiagonalen je drei Mal die Masse eines Körpers, dann den zugehörigen 3×3-Trägheitstensor.

Sei n die Anzahl der Körper im System und m die Anzahl der Zwangsbedingungen, dann gilt für die Dimensionen der Matrizen:

$$\begin{aligned} \mathbf{M}^{-1} &\in \mathbb{R}^{6n \times 6n} \\ \mathbf{J} &\in \mathbb{R}^{m \times 6n} \end{aligned} \tag{5.1}$$

Nutzt man die Dünnbesetztheit der Matrizen nicht aus, besitzt die Multiplikation von $\mathbf{M}^{-1} \cdot \mathbf{J}^T$ eine Komplexität von $O(mn^2)$, die von $\mathbf{J} \cdot \left(\mathbf{M}^{-1}\mathbf{J}^T\right)$ eine Komplexität von $O(nm^2)$.

Unter Ausnutzung der Dünnbesetztheit lassen sich diese Komplexitäten jedoch deutlich reduzieren.

Zunächst sollten beide Matrizen speicherplatzeffizient abgelegt werden. Tatsächlich wird die Massenmatrix gar nicht explizit als eigene Instanz vorgehalten. Jeder Körper speichert seine inverse Masse ($1/m$) und seinen inversen Trägheitstensor ($\mathbf{\Theta}^{-1}$). Da sich die inverse Massenmatrix $\mathbf{M}^{-1}$ des Mehrkörpersystems wegen ihrer Bandstruktur auf analogem Wege wie die nicht inverse konstruieren lässt (vgl. Kapitel 3.1.1), erlaubt eine Liste aller Körper des Systems daher implizit den Zugriff auf ihre Elemente.

Die Jacobimatrix (Klasse `RBMJacobeanMatrix` in Bibliothek VSLibRBDynMath) wird zeilenweise abgelegt, wobei jede Zeile eine Liste von Einträgen â sechs Elementen enthält. Neben den sechs Werten gehört zu einem Eintrag seine Spaltennummer, in der das erste Element des Eintrags in einer vollständigen Matrix stünde. In den Algorithmen 7 und 8 liefert die Funktion `colOffsetOf()` diesen Wert zurück. Jeder dieser Einträge verweist zusätzlich direkt auf denjenigen Körper, auf den bzw. auf dessen Geschwindigkeit er sich bezieht. Mit diesen Datenstrukturen lässt sich das Produkt $\mathbf{M}^{-1} \cdot \mathbf{J}^T$ mit der in Algorithmus 7 dargestellten Routine berechnen.

Gelenke und Kontakte erzeugen in der Jacobimatrix Zeilen mit zwei Einträgen. Nur Differenziale bewirken Einträge mit mehreren, höchstens vier Einträgen. Ein Differenzial erzeugt jedoch nur eine Zeile, ein typisches Gelenk mit einem Freiheitsgrad hingegen mindestens fünf, ein Kontakt mindestens drei Zeilen. Man darf daher die Annahme machen, dass die mit Abstand größte Zahl der Zeilen der Jacobimatrix nur zwei Einträge enthält. Keine besitzt mehr als vier Einträge. Die Anzahl der Durchläufe der inneren Schleife ist daher durch eine konstante Zahl begrenzt. Damit hat die Routine aus Algorithmus 7 eine Komplexität von nur noch $O(m)$ im Gegensatz zu $O(mn^2)$.

Unter den gleichen Voraussetzungen lässt sich die zweite nötige Multiplikation optimieren. Algorithmus 8 zeigt eine entsprechende Routine. Hier kann die Komplexität von $O(nm^2)$ auf $O(m^2)$ reduziert werden.

Im Ganzen kann die Komplexität der Bestimmung des Matrixproduktes $\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$ damit von $O(nm^2 + n^2m)$ auf $O(m + m^2)$ reduziert werden.

Algorithmus 7: Algorithmus zur Multiplikation der Matrizen $\mathbf{M}^{-1}$ und $\mathbf{J}^T$ mit linearer Komplexität.

```
integer iColumn = 0;
foreach jacobeanRow ∈ J do
    foreach entry ∈ jacobeanRow do
        // Anzahl Durchläufe durch konst. Schranke (4) begrenzt!
        body = bodyOf(entry);
        result [colOffsetOf(entry)+0][iColumn ] += 1/massOf(body) · entry [0];
        result [colOffsetOf(entry)+1][iColumn ] += 1/massOf(body) · entry [1];
        result [colOffsetOf(entry)+2][iColumn ] += 1/massOf(body) · entry [2];

        invInertia ← invInertiaOf(body) ;
        result [colOffsetOf(entry)+3] += invInertia [0][0] · entry [3]
                    + invInertia [0][1] · entry [4]
                    + invInertia [0][2] · entry [5] ;

        result [colOffsetOf(entry)+4] += invInertia [1][0] · entry [3]
                    + invInertia [1][1] · entry [4]
                    + invInertia [1][2] · entry [5] ;

        result [colOffsetOf(entry)+5] += invInertia [2][0] · entry [3]
                    + invInertia [2][1] · entry [4]
                    + invInertia [2][2] · entry [5] ;
    end
    iColumn += 1 ;
end
```

Algorithmus 8: Algorithmus zur Multiplikation der Matrizen $\mathbf{J}$ und $\mathbf{M}^{-1}\mathbf{J}^T$ mit quadratischer Komplexität.

```
integer iRow = 0;
foreach jacobeanRow ∈ J do
    foreach entry ∈ jacobeanRow do
        // Anzahl Durchläufe durch konst. Schranke (4) begrenzt!
        for integer iCol = 0 to columnsOf(M⁻¹Jᵀ) do
            result [iRow ][iCol ] += entry [0] · M⁻¹Jᵀ [colOffsetOf(entry)+0][iCol ]
                        + entry [1] · M⁻¹Jᵀ [colOffsetOf(entry)+1][iCol ]
                        + entry [2] · M⁻¹Jᵀ [colOffsetOf(entry)+2][iCol ]
                        + entry [3] · M⁻¹Jᵀ [colOffsetOf(entry)+3][iCol ]
                        + entry [4] · M⁻¹Jᵀ [colOffsetOf(entry)+4][iCol ]
                        + entry [5] · M⁻¹Jᵀ [colOffsetOf(entry)+5][iCol ] ;
        end
    end
    iRow += 1;
end
```

Kapitel 6.

Anwendungen

Die in dieser Arbeit beschriebenen systematischen Entwicklungen und Optimierungen führten zu einem leistungsfähigen und flexiblen Mehrkörperdynamiksimulationssystem, dessen Fähigkeiten im Folgenden anhand einer Reihe anspruchsvoller Anwendungen belegt werden sollen. Im Rahmen der Beschreibung einzelner Anwendungen werden dabei Einsatz und Nutzen spezieller Funktionen der entwickelten Dynamiksimulationskomponente hervorgehoben.

Obwohl diese wichtige Teile der Anforderungen einzelner Anwendungen direkt erfüllen kann, sind in vielen Fällen anwendunsspezifische Erweiterungen notwendig. Die Möglichkeiten zu deren Integration sind ein wichtiger Aspekt bei der Beurteilung des Ansatzes, Mehrkörperdynamiksimulation der Anwendungsentwicklung zu Grunde zu legen. Deshalb werden zu jeder der im Folgenden beschriebenen Anwendungen auch anwendungsspezifische Erweiterungen an oder auf Basis der Dynamiksimulation skizziert.

6.1. Harvestersimulatoren für die Maschinenführerausbildung

Harvester oder Holzvollernter sind wesentlicher Bestandteil der modernen Holzwirtschaft. Die Fahrzeuge sind extrem geländegängig und können mit dem an einem Kranausleger befestigten Harvesteraggregat einen Baum in wenigen Sekunden fällen, entasten und in unter ökonomischen Gesichtspunkten optimale Segmente zerteilen. Der Vorgang des Zerteilens wird Sortimentierung genannt. Abbildung 6.1 zeigt den Harvester des Forstlichen Bildungszentrums (FBZ) für Waldarbeit und Forsttech-

nik des Landes NRW in Neheim, Arnsberg (links) und einen Harvester des Herstellers Ponsse (rechts). Für die optimale Sortimentierung der aufgearbeiteten Segmente sorgt ein Bordcomputer. Anhand von Sensorwerten des Aggregats bestimmt er Längen und Durchmesser des gerade gefällten Baumes. Diese vergleicht er mit aktuell erreichbaren Marktpreisen und wählt dann ein optimales Sortiment. Der Fahrer muss nur die Baumart des gerade gefällten Baumes bestimmen, der Bordcomputer übernimmt dann die Sortimentierung.

In vielen Ländern existieren zentrale Ausbildungsstätten, in denen das Führen solcher Forstmaschinen in Ausbildungskursen erlernt werden kann. Aufgrund hoher Betriebskosten der realen Maschinen und hoher Folgekosten durch fehlerhafte Bedienung durch unerfahrene Maschinenführer rentiert sich der Einsatz von Simulatoren, auf denen Schüler ihre ersten Fahrstunden absolvieren können. Auf dem Simulator können alle Bedienelemente für Fahrzeug, Kran und Aggregat kennengelernt, trainiert und der Umgang mit dem Bordcomputer geübt werden. Steht dann die erste Fahrstunde auf einer realen Maschine an, sind viele grundlegende Fähigkeiten zur Bedienung des Systems bereits vorhanden.

Abbildung 6.1.: Links: Timberjack (heute John Deere) Harvester 1470D des Forstlichen Bildungszentrums NRW. Rechts: Ponsse Ergo Harvester im Einsatz, Bildquelle: Ponsse [100].

Die Realisierung entsprechender Simulatoren mit relativ geringem Aufwand ist eines der praktischen Ergebnisse dieser Arbeit. Den hier beschriebenen Entwicklungen ist eine rein kinematische Realisierung einer Harvestersimulation durch Freund et al. [50, 49] vorausgegangen. Kinematische Simulation bedeutet, dass Kräfte, Momente, Massen und Trägheiten bei der Simulation nicht berücksichtigt werden. Es findet eine rein phänomenologische Simulation auf Grundlage von Bewegungsmustern statt. Je-

der mögliche Bewegungsablauf muss im Vorhinein definiert werden. In der natürlichen Umgebung eines Holzvollernters, dem Wald mit seinen vielen Varianten, erscheint die Simulation auch dem Laien daher schnell unnatürlich. Die kinematische Simulation ist jedoch deutlich weniger rechenaufwändig als eine auf Grundlage eines physikalischen Modells. Zum Zeitpunkt der Erstellung des Simulators auf kinematischer Grundlage bot dieses Vorgehen daher einen optimalen Kompromiss aus verfügbarer Rechenleistung und nötiger Realitätsnähe.

Mit der Verfügbarkeit der hier vorgestellten, vielseitig einsetzbaren Mehrkörpersimulation und der notwendigen, höheren Rechnerleistung liegt es nah, die Harvestersimulation auf diese neue Grundlage zu stellen. Erhofftes Ziel ist eine realitätsnähere Simulation bei geringerem Realisierungsaufwand.

6.1.1. Anwendungsspezifische Anforderungen

Große Waldmodelle

Ein Waldmodell enthält typischerweise eine große Anzahl Bäume, die sich auf einer weiten Fläche verteilen. Um sowohl die Kollisionserkennung als auch die Dynamiksimulation realzeitfähig zu halten, sollen zu jedem Zeitpunkt immer nur die Teile des Gesamtmodells in der Simulation berücksichtigt werden, die aktuell mit dem Harvester und damit dem Hauptakteur interagieren können.

Fällen und Sägen

Wenn der Harvester einen Baum fällt und zersägt, müssen zur Simulationslaufzeit Körper aus der Mehrkörpersimulation gelöscht und neue hinzugefügt werden.

Simulation vieler kleiner, stabiler Holzstapel

Holzsegmente, die bereits gefällt und / oder aufgearbeitet wurden, werden typischerweise entsprechend ihrer Sortimente auf kleine sogenannte Polter gestapelt. In Realität nehmen diese Stapel sehr stabile Konfigurationen ein, die sie erst dann wieder aufgeben, wenn mit dem Harvesteraggregat solche liegenden Segmente aufgenommen werden sollen.

Harvesterfahrwerke

Die Fahrwerke heute üblicher, radgebundener Harvester haben einige typische gemeinsame Merkmale. So kommen zur Radaufhängung häufig sogenannte Tandem- oder Bogieachsen zum Einsatz. Eine Bogieachse hängt zwei Räder einer Seite an einer Pendelachse am Chassis auf. Durch diese Pendelaufhängung wird die Auswirkung einer überfahrenen Unebenheit auf die Bewegung des Chassis und damit der Basis des Krans oder der Kabine um die Hälfte reduziert. Weiterhin handelt es sich bei Harvestern um sogenannte Knicklenker. Das bedeutet das Fahrzeug wird gelenkt, indem hinterer und vorderer Fahrzeugteil gegeneinander „geknickt“ werden. Die Längsachse der Knicklenkung kann zusätzlich entriegelt werden, so dass vorderer und hinterer Fahrzeugteil in Grenzen, z.B. $-15,0° \cdots +15,0°$, um die Längsachse des Fahrzeugs verdreht werden können. Um bei stehendem Fahrzeug und Arbeit mit dem Kranausleger die Stabilität zu erhöhen, wird diese Achse automatisch verriegelt.

Komplexe Krankinematiken

Die Drehgelenke an einem Harvesterkran werden typischerweise durch parallel zum Drehgelenk liegende Hydraulikzylinder aktuiert. Bei einfachen Kränen, wie z.B. dem Ponsse Forwarderkran K70, handelt es um eine 1:1-Beziehung, vgl. Abbildung 6.13 auf Seite 202. Viele moderne Kräne jedoch besitzen komplexere Kinematiken, um die Bedienung zu vereinfachen. Solche Kinematiken werden häufig „Parallelkran“ genannt. So ist z.B. der Wippzylinder am Ponsse HN125 parallel zu Hub- **und** Wipparm montiert. Hierdurch kann das Harvesteraggregat durch eine Bewegung des Hubzylinders und damit durch eine eindimensionale Bewegung nur eines Knüppels auf einer nahezu linearen, horizontalen Linie bewegt werden, obwohl hierfür implizit zwei Drehgelenke bewegt werden.

6.1.2. Anwendungsspezifische Verfahren

Behandlung großer Waldmodelle

Da der Harvester selbst einziger Akteur in der Simulation ist, bietet es sich an, zu jedem Zeitpunkt nur seine unmittelbare Umgebung tatsächlich in der Mehrkörperdynamiksimulation zu berücksichtigen. Hierzu verwaltet der simulierte Harvester zwei vergrößerte *Bounding Boxes*, eine kleinere und eine größere (vgl. Abbildung 6.2). Initial

werden all jene Bäume für Kollisionserkennung und Dynamiksimulation aktiviert, die die äußere Box berühren oder innerhalb liegen. Bewegt sich das Fahrzeug, wird die innere Box relativ zu seiner Position mitbewegt, die äußere hält ihre Position. Erst wenn die innere die Grenzen der äußeren Box überschreitet, muss die Menge der aktiven Bäume aktualisiert werden. Hierzu wird die äußere Box so platziert, dass die innere genau in ihrem Zentrum liegt. Alle derzeit aktiven Bäume werden deaktiviert und alle jetzt von der äußeren Box berührten Bäume werden aktiviert. Um die Anzahl der *Bounding-Box*-Vergleiche für die Aktivierung der betroffenen Bäume gering zu halten, kommt ein Gitterraster mit konstanter Zellgröße zum Einsatz, wie es häufig auch in der Grobphase der Kollisionserkennung eingesetzt wird, vgl. Kapitel 2.3.1. Die gesamte Szene wird unterteilt durch ein einfaches zweidimensionales Gitter mit fest vorgegebener Zellengröße von z.B. 10 Metern $\times$ 10 Metern. Zu jeder Zelle gehört eine Liste der Stammsegmente, die innerhalb liegen oder sie berühren. Ebenso verweist jedes Stammsegment auf alle Zellen, die es berührt bzw. in denen es liegt.

Muss nun eine Aktualisierung der Menge der aktiven Bäume vorgenommen werden, so müssen nur all die Bäume auf Überschneidung mit der äußeren Box überprüft werden, die dieselben Zellen berühren, die auch die äußere Box berührt. Hierdurch wird die Anzahl der notwendigen *Bounding-Box*-Vergleiche gering gehalten.

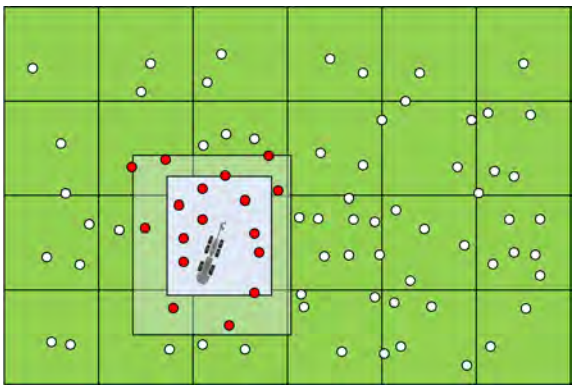


Abbildung 6.2.: Aktivierung relevanter Bäume: Verlässt der innere (variabel) Umgebungsbereich den äußeren (fest) Umgebungsgbereich, wird die Position des äußeren Bereichs aktualisiert, so dass der innere genau in seinem Zentrum liegt und alle Bäume im äußeren Bereich für die Simulation aktiviert.

Um den Aufwand für die Realisierung dieses Mechanismus gering zu halten, nutzt er bereits vorhandene Mechanismen der Kollisionserkennung des Dynamik-

Simulationssystems aus: Die Kollisionserkennung unterstützt ein Grobphasenverfahren mit Raumunterteilung durch ein Gitter mit konstanter Zellgröße. Eine Implementierung eines solchen Gitters ist daher bereits vorhanden. Die Klasse `RBDCollisionDetectionElement`, von der auch die Klasse für einen Starrkörper `RBDRigidBody` abgeleitet ist, bietet die Funktionen, um in ein oder mehrere solcher Gitter eingefügt werden zu können. Der Mehraufwand für die Implementierung der anwendungsspezifischen Funktionalität beschränkt sich daher auf die beiden vergrößerten Bounding Boxes für den Harvester, anhand derer die jeweils relevanten Bäume bestimmt werden können.

Löschen und Hinzufügen von Starrkörper zur Simulationslaufzeit

Bei der Zerteilung eines existierenden Stammsegments verändert sich die Menge der simulierten Körper im Mehrkörpersystem. Softwaretechnisch muss daher sichergestellt sein, dass die Dynamiksimulationskomponente auf einer variablen Menge von Körpern arbeiten kann.

Weiterhin muss bei der Zerteilung berücksichtigt werden, dass die neuen Segmente des aufgearbeiteten Baumes von Beginn an eine Geschwindigkeit besitzen, die i.A. ungleich Null ist und von der Bewegung des ursprünglichen Segments zum Zeitpunkt der Zerteilung abhängt. Die rotatorische Geschwindigkeit beider Körper ist identisch mit der des ursprünglichen Segments. Die translatorische Geschwindigkeit der Schwerpunkte bestimmt sich aus der Geschwindigkeit des Schwerpunkts des ursprünglichen Segments und der relativen Lage der neuen Schwerpunkte hierzu. Seien $\vec{v}, \vec{\omega}$ translatorische und rotatorische Geschwindigkeit des zerteilten Segments und seien weiterhin $\vec{r}_i, \vec{r}_j$ die Vektoren vom Schwerpunkt des zerteilten Segments zu den Schwerpunkten der neuen Segmente i und j, dann ergeben sich die translatorischen Geschwindigkeiten der neuen Segmente $\vec{v}_i, \vec{v}_j$ aus der Vorschrift:

$$\begin{aligned} \vec{v}_i &= \vec{v} + \vec{\omega} \times \vec{r}_i \\ \vec{v}_j &= \vec{v} + \vec{\omega} \times \vec{r}_j \end{aligned} \tag{6.1}$$

Simulation vieler kleiner, stabiler Holzstapel

Obwohl die Methode der Stoßfortpflanzung die stabile Simulation von Stapeln ermöglicht, ist es häufig sinnvoll, auf eine detaillierte Simulation von Stapeln gänzlich zu verzichten: In Realität wird ein Stapel von Stämmen, der einmal eine stabile Konfiguration

eingenommen hat, diese ohne äußere Einflüsse nicht wieder verlassen können. Selbst wenn ein weiterer Stamm auf den Stapel fällt, bleibt der Stapel selber in den meisten Fällen unbewegt. Um dieses Wissen im Sinne hoher Bildwiederholraten nutzen zu können, werden Körper in solchen Stapeln und allgemein solche Körper, die nicht unmittelbar wesentlich für die Dynamiksimulation sind, am Untergrund fixiert. Abbildung 6.3 zeigt einen entsprechenden Screenshot. Die Methoden des Kontaktgraphen bieten die notwendigen Werkzeuge: Ist ein Körper als nicht unmittelbar für die Dynamiksimulation relevant identifiziert, wird ein starres Gelenk (`RBDJointRigid`) zwischen Grund und Körper angelegt. Zu Beginn des nächsten Simulationszyklus sorgt eine Ausführung der dynamischen Rekonfiguration (vgl. Kapitel 4.1) dafür, dass Körper und Untergrund als ein logischer Starrkörper betrachtet werden. In der Kollisionserkennung und Kontaktpunktberechnung werden die Stämme selbstverständlich dennoch weiterhin berücksichtigt: Sie fügen sich, genau wie die geometrischen und physikalischen Grundkörper des ursprünglichen Bodenkörpers (*Shapes*), in die Hierarchie des Kollisionserkennungselements ein, vgl. hierzu Kapitel 5.4.1.

Abbildung 6.3.: Viele am Boden liegende Stammsegmente werden physikalisch mit dem Untergrund „verschmolzen“, solange das Aggregat nicht in unmittelbarer Nähe ist.

6.1.3. Anwendung generischer Verfahren und Modellierung

Stammsegmente

Die Bäume werden in der gegenwärtigen Ausbaustufe im physikalischen Modell nur durch ihre Stämme repräsentiert, vgl. Abbildung 6.4. Ausgehend von drei bis fünf Referenzdurchmessern werden die Stämme durch eine Menge von *Cylspheres*[1] approximiert, deren Durchmesser nach oben hin in abschnittsweise linearem Verlauf abnimmt.

Abbildung 6.4.: Das physikalische Ersatzmodell einer Harvestersimulation. Links: Visualisierungsmodell. Rechts: Visualisierung des physikalischen Ersatzmodells.

Harvesteraggregat

Das Aggregat ist aus mechanischer Sicht vermutlich die diffizilste Komponente des Systems Harvester. Ein Harvesteraggregat besitzt typischerweise ein oberes und ein unteres Zangenpaar zum Greifen eines Stammes, eine Vorschubwalze direkt am Aggregat sowie zwei weitere, an beweglichen Armen montierte Vorschubwalzen, vgl. Abbildung 6.5. Das Aggregat kann aufgestellt und abgekippt werden, so dass es dem gegriffenen, fallenden Baum folgen kann. Der Rotator erlaubt die Drehung des Aggregats um die Hochachse. Abbildung 6.6 zeigt das Visualisierungsmodell sowie die Visualisierung des physikalischen Ersatzmodells eines Ponsse Harvester Aggregats. Bemerkenswerterweise ist für die realitätsnahe Simulation des Aggregats keinerlei Sonderbehandlung

[1] *Cylsphere*: Zylinder mit Halbkugelkappen

notwendig: Das dargestellte Aggregat wird schlicht vollständig ausmodelliert mit insgesamt 14 Drehgelenken, sieben Motoren (Rotator, Kippgelenk, Walzen, Arme, Zangen oben und unten, Sägeschwert) sowie fünf Differenzialen (Zangen oben links und rechts, Zangen unten links und rechts, Arme links und rechts, Walzen links, rechts, und Mitte). Das Aggregat allein impliziert also 82 Zwangsbedingungen und damit Zeilen in der Jacobimatrix der Zwangsbedingungen. Um einen robusten Vorschub zu gewähren, sind die Walzen mit relativ großem ($\mu = 1.0$), die Zangen mit relativ kleinem Reibungskoeffizienten ausgestattet ($\mu = 0.1$). Das ausmodellierte Aggregat erlaubt bereits das Greifen und Vor- und Zurückschieben von Stämmen. Lediglich das Zersägen des gegriffenen Stammsegments erfordert eine kleine Sonderbehandlung. Diese umfasst die Erzeugung zweier neuer Stammsegmente und die Löschung des ursprünglichen Segments.

Abbildung 6.5.: Das Ponsse Harvesteraggregat H7, Quelle: Ponsse [100]

.

Fahrwerk eines Harvesters

Auch die Bogieachsen bedürfen keiner speziellen Behandlung: Jedes Rad wird mit einem Drehgelenk an der Bogieachse aufgehängt, die Bogieachse mit einem Drehgelenk am Chassis. Abbildung 6.7, links verdeutlicht die mechanische Konfiguration. Die blauen Achsen der Koordinatensysteme sind jeweils die Drehachsen am linken Bogie. Lediglich die Antriebskonfiguration ist besonders: In Realität sind beide Räder eines Bogies mechanisch gekoppelt, d.h. sie drehen sich immer mit gleicher Geschwindigkeit. Hierfür wird in der Simulation ein Differenzial mit Übersetzungsverhältnis $r = 1,0$ eingesetzt. Für eine realitätsnahe Kurvenfahrt müssen linkes und rechtes Räderpaar über ein weiteres Differenzial, ein Querdifferenzial, angetrieben werden. Dies ist ein Beispiel

Abbildung 6.6.: Das physikalische Ersatzmodell eines Harvesteraggregats. Links: Visualisierungsmodell. Rechts: Visualisierung des physikalischen Ersatzmodells.

für eine Differenzialrelation, in der die in Relation gesetzten Gelenke keinen gemeinsamen Körper besitzen. Die zugehörige Zeile in der Jacobimatrix enthält entsprechend vier Blöcke mit Einträgen ungleich Null.

Abbildung 6.7.: Links: Die drei Drehachsen (blau) an einem Bogiegelenk. Rechts: Ein Gelenk zur Realisierung eines Knicklenkers. In blau die Lenkachse, in Rot die freie Achse mit begrenzter Auslenkung.

Für die Knicklenkung steht ein eigener Gelenktyp bereit, repräsentiert durch die Klasse `RBDJointArticulatedSteering`. Abbildung 6.7, rechts visualisiert die relevanten Freiheitsgrade. Entlang der Längsachse (rot) können die verbundenen Körper innerhalb vorgegebener Maximalausschläge frei verdreht werden. Die Lenkung erfolgt um die z-Achse (blau). Hier wird typischerweise ein geschwindigkeitsbasierter Motor eingesetzt, aber auch momentbasierte Motoren und Positionssteller können modelliert werden. Die Lenkachse besitzt außerdem zusätzlich einen Maximalausschlag.

Eine weitere wichtige Eigenschaft der Knicklenkung an den Forstarbeitsmaschinen ist die Möglichkeit der Verriegelung der freien Längsachse. Die Verriegelung ist notwendig, um die Standfestigkeit des Gesamtfahrzeugs bei Arbeit mit weit ausgelegtem Kran zu erhöhen. Sie passiert typischerweise automatisch in Abhängigkeit von der Fahrgeschwindigkeit: Steht das Fahrzeug, wird das Gelenk automatisch verriegelt. Der Gelenktyp besitzt zur Realisierung dieses Mechanismus eine zusätzliche Eigenschaft: Er kann den Eingang referenzieren, der die Sollfahrgeschwindigkeit entgegennimmt. Unterschreitet dieser Wert einen Grenzwert, wird das Gelenk verriegelt. Natürlich wird das verriegelte Knickgelenk durch eine starre Verbindung im Kontaktgraphen repräsentiert, so dass die Verriegelung in der logischen Verschmelzung von vorderem und hinterem Fahrzeugteil resultiert. Dies wiederum führt durch die Reduzierung der Anzahl der Körper im System und die Entfernung der Zwangsbedingungen, die das Knickgelenk impliziert, zu einer Verbesserung des Laufzeitverhaltens der Simulation.

Komplexe Krankinematiken

In einer kinematischen Realisierung einer Krankinematik, wie der im Ponsse HN125-Kran, muss eine Zusatzlogik für die stimmige Stellung von Hydraulikzylindern und parallelen Drehgelenken sorgen. Auf Grundlage der Dynamiksimulation ist die Lösung dieses Problem ebenso einfach wie genial: Modelliere die Modelldynamik genauso, wie die reale Dynamik. Im Simulationsmodell werden tatsächlich die Hydraulikzylinder und nicht die parallel verlaufenden Drehgelenke aktuiert. In den Hydraulikzylindern werden Linearmotoren durch Zwangsbedingungen realisiert. Hierdurch ist keine Zusatzlogik zur Realisierung solcher Kinematiken nötig. Das Modell bildet sie und sogar ihre Schwächen automatisch nach: Wird z.B. beim Ponsse HN125-Kran der Hubarm sehr weit nach vorn oder sehr weit nach hinten ausgelenkt, nähert sich die Kinematik einer Singularität, wenn das Drehgelenk am Wipparm nahezu $0°$ bzw. $180°$ erreicht, Abbildung 6.8 verdeutlicht beide Extremalstellungen. In Realität wie in der Simulation bemerkt man diese Nähe durch eine Abnahme der „gefühlten“ Stellkräfte am Kran, ohne dass hierfür eine Zeile Code notwendig wäre.

6.1.4. Erfahrungen und Ergebnisse

Unter Einsatz der oben beschriebenen, anwendungsspezifischen und generischen Verfahren konnte die Harvestersimulation derart realisiert werden, dass sie mit interaktiven

Abbildung 6.8.: Die Extremalstellungen des Ponsse HN125 Harvesterkrans

Bildwiederholraten robust und in Echtzeit abläuft. Abbildung 6.9 zeigt je ein Foto eines Simulators im Forstlichen Bildungszentrum für Waldarbeit und Forsttechnik des Landes NRW, im Forstlichen Bildungszentrum Münchehof des Landes Niedersachsen, im Forstlichen Bildungszentrum Magdeburgerforth des Landes Sachsen-Anhalt und in der Forstlichen Ausbildungsstätte Ort, Gmunden in Österreich.

Die Bedienung der simulierten Maschine erfolgt über einen Bedienstuhl, der baugleich auch in den realen Maschinen verbaut wird. Ebenso ist ein unveränderter Bordcomputer angebunden, der u.a. die Steuerung und Regelung der Vorschubfunktion des Harvesteraggregats übernimmt. Bordcomputer und Simulation bilden hier eine Regelschleife, um den Stamm in Segmente vorgegebener Länge zu zerteilen. Der Bordcomputer stellt Sollgeschwindigkeiten der Vorschubwalzen, die Simulation liefert tatsächliche Vorschublänge und -geschwindigkeit zurück. Die Visualisierung der Simulation geschieht in den vorgestellten Installationen über ein Stereo-Rückprojektionssystem der Dortmunder Initiative für rechnerintegrierte Fertigung (RIF) e.V. [3]. Die Bilder für linkes und rechtes Auge werden mittels Linearpolarisationsfilter getrennt. Der Tiefeneindruck ist den Fahrern eine große Hilfe bei der Platzierung des Harvesteraggregats am Baum.

Die Bewertung der Realitätsnähe der Simulation ist nicht einfach. Nach Aussagen der den Simulator einsetzenden Ausbilder, die über umfangreiche Erfahrungen auf realen Maschinen verfügen, erreicht sie aber einen hohen Grad. Hierfür sorgen z.B. variierende Fallmuster gefällter Bäume, die hohe Kippgefahr des Fahrzeugs, die „spürbaren“ Auswirkungen eines schweren Baumes im Aggregat am weit ausgelegten Kran und viele weitere kleine Details im Bewegungsverhalten der Maschine, die mit einer rein phänomenologischen, kinematischen Simulation nur unzureichend nachgebildet werden können.

Abbildung 6.9.: Die Harvestersimulatoren im Einsatz in unterschiedlichen Ausbildungsstätten in Deutschland und Österreich, v.l.o.: FBZ Magdeburgerforth (Sachsen-Anhalt), FBZ Münchehof (Niedersachsen), FAST Ort, Gmunden in Österreich, FBZ Neheim (NRW)

Bewertung der Entwicklungsaufwände

Der größte Teil der Logik der Harvestersimulation beruht auf der reinen Mehrkörperdynamiksimulation und hierauf aufbauenden Verfahren. Dennoch musste einiges an anwendungsspezifischer Logik in Form einer zusätzlichen Softwarebibliothek ergänzt werden. Interessant ist daher die Frage, wie groß der Anteil der anwendungsspezifischen Software am Gesamtaufwand ist. Um dies zumindest grob einschätzen zu können, wird als Softwaremetrik die Anzahl der aktiven Codezeilen der eingesetzten Bibliotheken verglichen. Für die rein kinematische Realisierung steht die Bibliothek „VSPluginHarvester“. Sie behandelt die unterschiedlichen Bewegungsmuster fallender Stammsegmente, spezielle Kran-Kinematiken, das Fahren der Maschine auf ebenem Grund,

Bibliothek	Funktion	Aktive Codezeilen
VSLibRBDynamX	Dynamiksimulation, Kollisionserkennung & -behandlung	20906
VSLibRBDynMath	Dynamikbezogene Mathematikklassen	3395
VSPluginRBDynamX	Schnittstelle zwischen Dynamiksimulation und VEROSIM®	8004
VSPluginRBHarvester	Anwendungsspezifische Erweiterungen der Dynamiksimulation für den Harvestersimulator	4989
VSPluginHarvester	Rein kinematische Realisierung einer Harvestersimulation	46312
VSPluginRBForwarder	Anwendungsspezifische Erweiterungen der Dynamiksimulation für den Forwardersimulator	2011

Tabelle 6.1.: Anzahl aktiver Codezeilen der Bibliotheken zur Realisierung einer Harvestersimulation

Greifen und Vorschub des Aggregats und das Sägen der Bäume. Die hier realisierte Mehrkörpersimulation ist in drei Bibliotheken gekapselt: VSLibRBDynamX umfasst den Kern der Mehrkörperdynamiksimulation und die Kollisionserkennung und -behandlung, VSLibRBDynMath kapselt einige dynamikrelevante Mathematikklassen und VSPluginRBDynamX dient als Schnittstelle zum Simulationssystem VEROSIM®. Die Bibliothek VSPluginRBHarvester kapselt alle anwendungsspezifische Logik, um welche die generische Mehrkörpersimulation zur Realisierung der Harvestersimulation ergänzt werden musste.

Tabelle 6.1 zeigt die Auswertung nach aktiven Codezeilen. Bemerkenswert an den Ergebnissen ist bereits die Tatsache, dass die Realisierung auf Basis der Dynamiksimulation insgesamt (≈37000) eine kleinere Anzahl aktiver Codezeilen aufweist, als die rein kinematische (≈46000). Diese große Anzahl ist in erster Linie auf die zahlreichen notwendigen Fallunterscheidungen zurückzuführen, die notwendig sind, um den gegenwärtigen Zustand der Simulation zu identifizieren. Vor allem aber haben die anwendungsspezifischen Erweiterungen der dynamikbasierten Harvestersimulation (VSPluginRBHarvester) einen um eine Größenordnung kleineren Umfang (≈5000 Zeilen) als die vollständig phänomenologische, kinematische Realisierung. Dass die Mehrkörperdynamiksimulation eine hervorragende Grundlage zur Realisierung solcher Anwendungen ist, wird hieran deutlich.

6.2. Forwardersimulatoren für die Maschinenführerausbildung

Der Forwarder dient als Ergänzung zum Harvester im Holzernteprozess: Nachdem Bäume gefällt und sortimentiert worden sind, transportiert der Forwarder das Holz zur nächsten LKW-befahrbaren Straße. Forwarder sind Harvestern im Aufbau sehr ähnlich, vor allem besitzen sie annähernd identische Fahrwerkskonstruktionen. Zusätzlich zum Kran besitzt der Forwarder einen sogenannten Rungenkorb, in den er bis zu 10 Tonnen Holz laden kann. Anstelle des Harvesteraggregats ist an der Kranspitze ein Greifer montiert, mit dem Stammsegmente vom Boden aufgehoben werden können. Abbildung 6.10 zeigt zwei Fotos von Forwardern im Einsatz.

Abbildung 6.10.: Forwarder im Einsatz: Links der Ponsse Wisent Forwarder des Forstlichen Bildungszentrums NRW. Rechts: Foto eines Ponsse Wisent Forwarders, Bildquelle: Ponsse[100].

6.2.1. Anwendungsspezifische Anforderungen

Auch für den Forwarder existiert bereits eine rein kinematische Realisierung. Bei dieser Anwendung, die in noch größerem Maße von „dynamischem Verhalten“ aller Komponenten der Simulation profitiert, waren die Unzulänglichkeiten einer kinematischen Implementierung für den Anwender jedoch noch deutlicher sprübar als beim Harvester. Umso mehr profitiert diese Anwendung von der physikalischen Grundlage des neuen Simulationsmodells.

Aufgrund der Ähnlichkeiten zur Harvestersimulation gleichen viele Anforderungen ei-

ner Forwardersimulation an die Dynamiksimulation denen der Harvestersimulation. Eine Anforderung ist dennoch absolut charakteristisch für den Forwarder:

Stabile Lagerung von Transportgut auf einer fahrenden Plattform

Ein Forwarder kann bis zu 10 Tonnen Holz in seinen Rungenkorb laden und transportieren. Aufgrund der großen Reibung zwischen Holzstämmen in der Realität durch sehr raue Oberflächen des Holzes erreicht dieser bewegte Stapel große Stabilität. Am Ziel angekommen, kann der Greifer in den Stapel „eingestochen“ werden, um das Holz neben dem Fahrzeug wieder abzuladen oder direkt auf einen LKW aufzuladen. Ein solcher, großer und beweglicher Stapel ist eine besondere Herausforderung für die Simulation.

6.2.2. Anwendungsspezifische Verfahren

Heuristik zur Verschmelzung von Ladung und Fahrzeug

Während der Modellierung einer Forwardersimulation wird schnell deutlich, dass eine vollständige Simulation der Ladung mit dem eingesetzten Simulationsverfahren kaum möglich sein wird. Einerseits ist die Approximation des Reibungsverhaltens durch das Coulombsche Modell offensichtlich nicht ausreichend, um große statische Reibungskräfte zwischen geladenen Holzstämmen ausreichend genau zu simulieren. Dies äußert sich in dem Phänomen, dass die Stämme permanent verrutschen und während der Fahrt schließlich vom Wagen fallen. Andererseits erzeugt der Stapel Holz im Rungenkorb eine sehr große Anzahl Kontaktpunkte, so dass mit dem hier vorgestellten Verfahren keine Echtzeitfähigkeit der Simulation mehr gewährleistet werden kann.

Um dieses Problem zu umgehen und dennoch eine realitätsnahe Simulation zu erreichen, werden die im Korb befindlichen Stämme mit dem Fahrzeug physikalisch „verschmolzen“, solange die Simulation des individuellen Segments nicht notwendig ist. Hierdurch wird die Ladung absolut stabil, gleichzeitig ändern sich Gesamtmasse und Trägheitsverhalten des Fahrzeugs genau so, als würde jedes Stück der Ladung einzeln simuliert.

Um eine Entscheidung treffen zu können, wann ein Stammsegment mit dem Fahrzeug verschmolzen werden kann bzw. wann die Verbindung wieder aufzuheben ist, kommen mehrere Kriterien zum Einsatz:

- Befindet sich das Segment geometrisch innerhalb des Laderaums? Hierzu werden je zwei Referenzpunkte innerhalb eines Segments daraufhin überprüft, ob sie innerhalb eines den Laderaum überdeckenden Quaders liegen.
- Ist das Segment noch in der Luft? Besitzt ein Segment gerade keinen Kontaktpunkt, kann es nicht auf der Ladung bzw. der Ladefläche aufliegen.
- Bewegt sich das Segment noch? Natürlich muss die Geschwindigkeitsdifferenz zwischen Segment und Laderaum einen gegebenen Grenzwert unterschreiten, bevor eine Verschmelzung stattfinden kann.
- Wie groß ist die Pfaddistanz zum Greifer? Wird das Segment oder einer seiner Kontaktnachbarn vom Greifer berührt, wird das Segment nicht verschmolzen. Zur Auswertung dient eine Abfrage im Kontaktgraphen, wie sie in Kapitel 4.4.2 behandelt werden.
- Liegt das Segment „ungefähr waagerecht“? Stabile Konfigurationen der Ladung in der Realität zeichnen sich dadurch aus, dass der Stamm in der Ebene der Ladefläche liegt, nicht etwa leicht aufrecht stehend oder gar steil an eine der Wände gelehnt. Diese Bedingung kann durch einen einfachen Orientierungsvergleich überprüft werden.

Je nach gegenwärtigem Zustand eines Stammsegments kommen unterschiedliche Kombinationen o.g. Kriterien zum Einsatz. Diese stellen effektiv sicher, dass eine kleinstmögliche Menge der Segmente im Korb tatsächlich individuell simuliert wird. Da Kran und Greifer die einzigen Interaktionsmedien zwischen Anwender und Rungenkorb sind, merkt der Anwender von dieser Vereinfachung der Simulation nichts. Immer, wenn er mit dem Greifer in den Rungenkorb „hineingreift“, werden die betroffenen Segmente vom Fahrzeug gelöst.

Um den Rechenaufwand gering zu halten, werden auch in der Forwardersimulation nur diejenigen Stammsegmente außerhalb des Rungenkorbs in der Dynamiksimulation berücksichtigt, die in einer gewissen örtlichen Nähe zum Fahrzeug stehen. Das Verfahren ist dasselbe wie in der Harvestersimulation, siehe hierzu Kapitel 6.1.2.

Der eigentliche Verschmelzungsvorgang nutzt die Methode der dynamischen Rekonfiguration (vgl. Kapitel 4.1): Wird für ein Stammsegment festgestellt, dass es mit dem Fahrzeug verschmolzen werden kann, wird eine entsprechende Kante im Kontaktgraphen in Form einer starren Verbindung (`RBDJointRigid`) angelegt. Vor dem nächsten

Zeitschritt in der Dynamiksimulation ordnet eine Tiefensuche diesen Stamm dann dem Körper zu, zu dem der Rungenkorb und ggf. schon andere Stämme gehören.

6.2.3. Anwendung generischer Verfahren und Modellierung

Abbildung 6.11 zeigt eine Gegenüberstellung von Visualisierungs- und physikalischem Ersatzmodell eines Ponsse S10 Forwarders. Die Stammsegmente sind durch *Cylspheres* approximiert. Forwarder-Fahrwerke entsprechen größtenteils denen von Harvestern. Auch hier werden Knicklenkung und Bogieachsen verbaut.

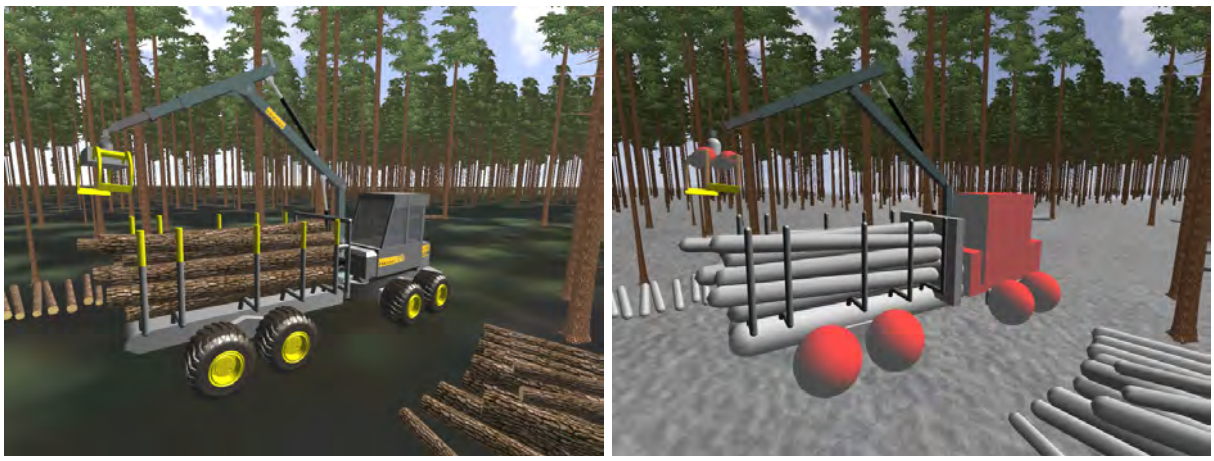


Abbildung 6.11.: Modellierung eines Ponsse S10 Forwarders: Links die Visualisierung, rechts das physikalische Ersatzmodell.

Für das Greifen von Stämmen wird wiederum keine Speziallösung eingesetzt, stattdessen ist der Greifer vollständig dynamisch ausmodelliert worden. Einzige Besonderheit hierbei ist die Differenzialzwangsbedingung, die dafür sorgt, dass sich linke und rechte Zange immer gleich in Relation zur Greiferbasis bewegen, denn sie werden auch in Realität nur durch einen Motor und eine Kopplungsmechanik bewegt.

6.2.4. Erfahrungen und Ergebnisse

Der Forwardersimulator konnte unter Ausnutzung der oben beschriebenen heuristischen Verfahren echtzeitfähig und mit interaktiven Bildwiederholraten realisiert werden [70, 104]. Es ist möglich, den Rungenkorb bis an die Kapazitätsgrenze zu befüllen. Dabei bleibt die Stabilität der Simulation erhalten und auch die Bildwiederholrate sinkt

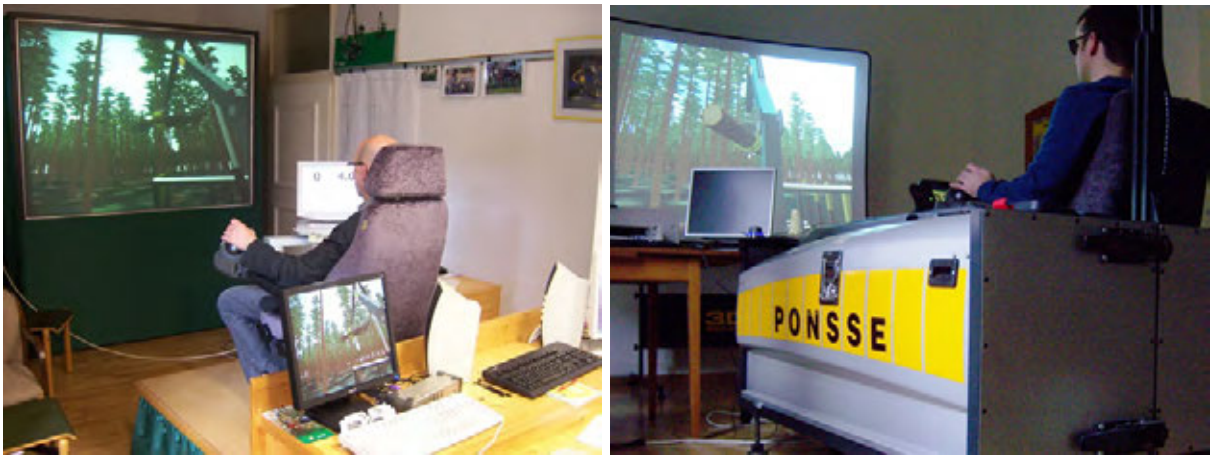


Abbildung 6.12.: Der Forwarder-Simulator im Einsatz, v.l.: In der Forstlichen Ausbildungsstätte Ort, Gmunden in Österreich und im Forstlichen Bildungszentrum Magdeburgerforth.

nur geringfügig. Dies wird vor allem durch die für den Anwender intransparente „Verschmelzung“ von Ladung und Fahrzeug erreicht. Der Forwardersimulator ist in denselben Ausbildungsstätten wie der Harvestersimulator im Einsatz, Abbildung 6.12 zeigt die Installation in der Forstlichen Ausbildungsstätte Ort, Gmunden in Österreich und im Forstlichen Bildungszentrum Magdeburgerforth. Die Bedienung der Forwardersimulation geschieht über dieselben Bedienelemente, mit denen auch die Harvestersimulation gesteuert wird. Ein Bordcomputer spielt im Forwarder eine weniger wichtige Rolle, auf eine eigene Anbindung wurde daher verzichtet. Die Bedienelemente, d.h. Knüppel, Tasten und Pedale, sind auch in Realität identisch.

Der Zusatzaufwand für die Heuristiken ist relativ gering. Die Bibliothek VSPluginRB-Forwarder, die die Dynamiksimulation um die anwendungsspezifische Logik ergänzt, zählt gerade einmal 2011 aktive Codezeilen (vgl. Tabelle 6.1). Diese Zahl belegt eindrucksvoll, dass die zugrunde gelegte Mehrkörperdynamiksimulation durch relativ kleine Ergänzungen für ein breites Anwendungsspektrum eingesetzt werden kann.

Die Realitätsnähe der Simulation wird von erfahrenen Maschinenführern als gut beurteilt. Gerade an der Forwardersimulation bemerken erfahrene Fahrer viele kleine Details im Bewegungsablauf, die nur auf einer physikalischen Grundlage möglich sind. Hierzu zählt zum Beispiel das Bündigschlagen von mehreren gegriffenen Stämmen am Boden (vgl. Abbildung 6.13, rechts). Hierzu wird der Kran einmal hin- und hergedreht, so dass sich Stämme und Greifer in die Vertikale drehen. Dann werden die Stämme am Boden aufgestützt und der Greifer leicht geöffnet, so dass alle Stämme im Greifer bündig zu

Boden rutschen. Danach wird der Greifer wieder geschlossen. Ähnlich funktioniert das Bündigschlagen an der Rückwand des Rungenkorbs. Auch hier wird der Greifer leicht geöffnet und gegen die Rückwand bewegt, so dass die Stämme im Greifer bündig mit der Rückwand abschließen. Solche Vorgänge sind in phänomenologischen Simulationen mit programmierten Bewegungsabläufen nur mit großem Aufwand zu realisieren, auf Grundlage einer Dynamiksimulation erhält man sie hingegen ohne jeden Mehraufwand.

Abbildung 6.13.: Links: Der simulierte Forwarder mit hoch beladenem Rungenkorb. Rechts: „Bündigschlagen“ der gerade gegriffenen Stammsegmente am Boden.

6.3. Radlader- und Schüttgutsimulation

Die Radladersimulation ist eine weitere Arbeitsmaschinensimulation, die mit dem hier vorgestellten Dynamiksimulationssystem einfach und schnell realisiert werden konnte. Sie kommt in zwei Varianten: In der ersten Variante hat der Bediener die Aufgabe, Felsbrocken auf einen Kipplaster zu laden. Diese Anwendung basiert auf der reinen Mehrkörperdynamiksimulation und bedurfte keinerlei anwendungsspezifischer Ergänzungen. In der zweiten Variante wird der Radlader mit einer Schüttgutsimulation auf Basis zellulärer Automaten kombiniert. Schüttgutsimulation in interaktiven Szenarien ist bis heute eine große Herausforderung. Verfahren auf Grundlage physikalischer Modelle wie Partikel- ([145, 97]) oder gar Starrkörperdynamik ([64, 60]) sind für Echtzeitfähigkeit und interaktive Bildwiederholraten immer noch zu rechenaufwendig.

Roßmann et al. [110] stellen ein realitätsnahes Verfahren zur Schüttgutsimulation auf Basis zellulärer Automaten vor. Hierfür wird der potenziell mit Schüttgut gefüllte Raum

durch ein gleichmäßiges, dreidimensionales Gitter in Zellen unterteilt. Das Verhalten jeder dieser Zellen wird durch einen Automaten beschrieben, dessen Zustände im Wesentlichen zwischen „gefüllt“ und „nicht gefüllt“ unterschieden werden. Anhand eines einfachen Regelsatzes ändert jede Zelle ihren Zustand. Wesentlich für die hohe Effizienz des Verfahrens ist, dass jede Zelle nur mit einer begrenzten Anzahl Nachbarzellen interagiert, d.h. die Regeln zum Wechsel ihres Zustands werten lediglich den eigenen und die Zustände der direkten Nachbarzellen aus. Zur Interaktion mit Schüttgut werden in [110] bereits Verfahren beschrieben, wie z.B. mit einer Schaufel das Schüttgut bearbeitet oder transportiert werden kann.

6.3.1. Anwendungsspezifische Anforderungen

Um eine realitätsnahe Verbindung zwischen Dynamik- und Schüttgutsimulation herzustellen, ist eine bidirektionale Interaktion notwendig: Zwar unterstützt das Schüttgutsimulationsverfahren bereits die Möglichkeit, das Schüttgut auf Basis rein kinematischer Informationen wie Geschwindigkeit und Position einer Schaufel zu bearbeiten. Ist die Schaufel ihrerseits Teil einer Mehrkörperdynamiksimulation, müssen für ein realitätsnahes Bewegungsverhalten auch entsprechende Dämpfungs- und Gewichtskräfte aus dem Schüttgut in die Dynamiksimulation zurückgeliefert werden.

Für diesen Rückweg werden drei Effekte gefordert:

1. **Masse**: Je nach Füllgrad der Schaufel des Radladers soll die Gesamtmasse der Schaufel variieren. Dies äußert sich in verändertem Fahrverhalten und vergrößerter Kippgefahr.

2. **Widerstand**: Sticht die Schaufel in einen Sandhaufen, müssen große Dämpfungskräfte wirken, die das gesamte Fahrzeug zum Stehen bringen können.

3. **Unebener Untergrund**: Für andere Körper der Dynamiksimulation neben der Schaufel soll sich das Schüttgut wie ein unebener, fester Untergrund verhalten.

6.3.2. Anwendungsspezifische Verfahren

Variable Masse innerhalb der Schaufel

Hierzu wird im Starrkörperdynamikmodell des Radladers innerhalb der Schaufel ein weiterer Starrkörper modelliert, dessen Masse zur Laufzeit der Simulation durch den

zellulären Automaten in der Schaufel variiert werden kann. Unter Annahme einer konstanten Dichte und Masse des Schüttguts wird die anzusetzende Masse proportional zur Anzahl der gefüllten Zellen gewählt. Einen interessanten Effekt der erhöhten Masse zeigt Abbildung 6.14: Durch den stark nach vorn verschobenen Schwerpunkt neigt der Radlader beim Bremsen dazu, vorn über zu kippen. Das bewusste Herbeiführen dieses Vorgangs wird „Stoppie“ genannt.

Beschleunigen... ...Bremsen... ...Kippen

Abbildung 6.14.: Der simulierte Radlader führt mit schwer beladener Schaufel einen „Stoppie“ aus.

Robustes Aufbringen von Widerstandskräften und -momenten

Der zweite wichtige Effekt sind Widerstandskräfte und -momente, die die Schaufel erfährt, wenn sie in einen Sandhaufen eingestochen wird. Typisch für solche Kräfte ist, dass sie große Beträge annehmen und natürlich auch, dass sie nur bremsend, niemals beschleunigend wirken. Rekapituliert man noch einmal die Eigenschaften der hier eingesetzten Mehrkörperdynamiksimulationskomponente wird schnell klar, dass solche Kräfte und Momente nicht in Form externer Kräfte in das System eingebracht werden sollten (vgl. auch Kapitel 3.4.3). Einerseits kann die automatenbasierte Schüttgutsimulation nur sehr grob die auftretenden Kontaktkräfte schätzen. Andererseits variieren diese Kräfte schon innerhalb der im hier eingesetzten Verfahren üblichen Zeitschrittweiten von etwa $1/100s$ sehr stark, so dass durch die implizite Annahme, die Kräfte seien über einen Zeitschritt hinweg konstant, große Fehler auftreten. Diese äußern sich dann z.B. in Phänomenen, wie einem Sandhaufen, der die Schaufel aktiv „von sich weg schiebt“.

Stattdessen bietet es sich an, solche dämpfenden Kräfte implizit durch Zwangsbedingungen im System zu repräsentieren. Eine solche Zwangsbedingung ähnelt der eines Kontaktpunktes. Als Kontaktnormale bzw. Richtung der Dämpfungskraft wird eine lo-

kale Oberflächennormale $\vec{n}_S$ der Sandoberfläche genutzt. Der Kontakt soll sich nicht ganz starr verhalten - ein heuristisch in der Schüttgutsimulation bestimmter Kontaktkraftbetrag f_W wird daher als oberer Grenzwert für den korrespondierenden Lagrange-Multiplikatoren angesetzt. Für die Dämpfungszwangsbedingung gilt wie für die Kontaktnormalenzwangsbedingung, dass sie drücken, aber nicht ziehen darf. Der untere Grenzwert des Lagrange-Multiplikators ist daher gleich Null. Im Gegensatz zum Kontakt betrifft die Zwangsbedingung hier jedoch nur einen Körper, die Schaufel.

Sei $\vec{n}_S$ also die Kraftrichtung, $\vec{p}$ der Angriffspunkt der dämpfenden Kraft an der Schaufel, i der Index des Schaufelkörpers in der Mehrkörpersimulation, $\vec{s}_i$ sein Schwerpunkt, $\vec{v}_i$ und $\vec{\omega}_i$ seine translatorische bzw. rotatorische Geschwindigkeit. Dann lautet die notwendige Zwangsbedingung:

$$\begin{aligned} \vec{n} \cdot (\vec{v}_i + \vec{\omega}_i \times (\vec{p} - \vec{s}_i)) \quad &= b + w, \quad b = 0 \\ \Leftrightarrow \vec{n} \cdot \vec{v}_i + \vec{n} \cdot (\vec{\omega}_i \times (\vec{p} - \vec{s}_i)) \quad &= b + w \\ \Leftrightarrow \vec{n} \cdot \vec{v}_i - \vec{n} \times (\vec{p} - \vec{s}_i) \cdot \vec{\omega} \quad &= b + w \end{aligned} \tag{6.2}$$

Mit $\vec{r}_i = \vec{p} - \vec{s}_i$ sind die Einträge der Teil-Jacobimatrizen und die zugehörigen Grenzwerte der Lagrange-Multiplikatoren folglich:

$$\begin{aligned} \mathbf{J}_{i,\mathrm{trans}} = \begin{pmatrix} \vec{n}_1 & \vec{n}_2 & \vec{n}_3 & -(\vec{n} \times \vec{r}_i)_1 & -(\vec{n} \times \vec{r}_i)_2 & -(\vec{n} \times \vec{r}_i)_3 \end{pmatrix} \\ 0 \leq \lambda \leq h \cdot f_W \end{aligned} \tag{6.3}$$

Mit solchen Zwangsbedingungen können damit an einer beliebigen Anzahl Punkte der Schaufel beliebig große Widerstandskräfte aufgebracht werden, wobei die inverse Dynamik sicherstellt, dass nie mehr Kraft aufgebracht wird als nötig ist, um den betreffenden Punkt an der Radladerschaufel zum Stehen zu bringen. Die Erzeugung solcher Zwangsbedingungen erledigen Instanzen der Klasse `RBDDeceleration`. Abbildung 6.15 zeigt einige Eindrücke der Kombination von Mehrkörper- und Schüttgutsimulation.

Weitere Details, vor allem zum hier angebundenen Verfahren der Schüttgutsimulation selbst, sind [110, 111] zu entnehmen.

Abbildung 6.15.: Zusammenspiel von automatenbasierter Schüttgutsimulation und Radlader auf Basis der Mehrkörperdynamiksimulation

Schüttgut als unebener Untergrund

Die Interaktion zwischen Schüttgut und Schaufel des Radladers ist eine Speziallösung - für die Interaktion zwischen allen anderen Körpern der Mehrkörperdynamiksimulation und dem Schüttgut wird noch eine Repräsentation des Schüttguts in der Mehrkörperdynamiksimulation benötigt. Aufgrund der Ähnlichkeiten der internen Darstellungen bietet sich hierfür ein Höhenfeld (engl.: *Heightfield*) an. Das Höhenfeld speichert Höhenwerte über einem zweidimensionalen, äquidistanten Gitter. In der Bibliothek VSLibRBDynamX wird das Höhenfeld durch die Klasse `RBDShapeHeightField` repräsentiert. Die Methoden zur Kollisionserkennung und Kontaktpunktberechnung sind in der korrespondierenden Klasse `ExtensionRigidBodyShapeHeightField` der VEROSIM®-Schnittstellenbibliothek VSPluginRBDynamX implementiert, da die Daten des Höhenfeldes selbst in einer anderen VEROSIM®-Bibliothek für die Schüttgutsimulation generiert und verwaltet werden.

Im Gegensatz zum Anwendungsbeispiel im nächsten Kapitel, in dem durch ein anderes Verfahren auch eine Verformung einer nicht starren Oberfläche durch einwirkende Starrkörper unterstützt wird, haben bei dieser Anwendung alle anderen Körper außer der Schaufel keinen Einfluss auf das Schüttgut.

6.3.3. Anwendung generischer Verfahren und Modellierung

Alle dynamischen Komponenten des Radladers wurden vollständig ausmodelliert, inklusive der Hydraulikzylinder und der Z-Kinematik zur Kippung der Schaufel. Auch in diesem Modell werden wie bei Forwarder und Harvester direkt die Hydraulikzylinder über geschwindigkeitsbezogene Zwangsbedingungen aktuiert. Abbildung 6.16 zeigt Vi-

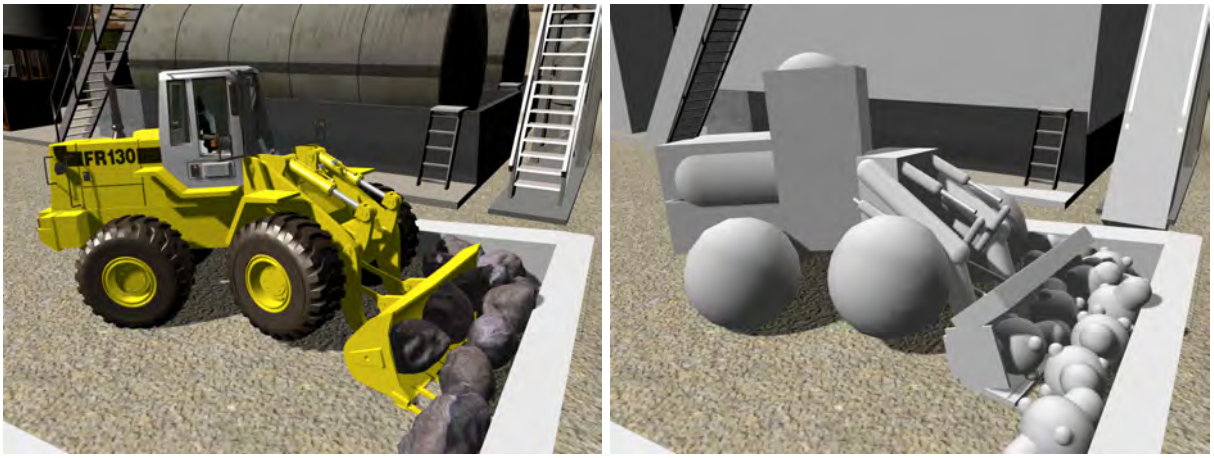


Abbildung 6.16.: Visualisierungsmodell und physikalisches Ersatzmodell der Radladersimulation mit Felsbrocken als Ladegut

sualisierungsmodell und die Visualisierung des physikalischen Ersatzmodells. Natürlich lässt sich das LKW-Modell durch einen zweiten Benutzer gleichzeitig mit dem Radlader interaktiv steuern. Neben der reinen Fahrfunktion kann der Kipplaster seine Ladefläche abkippen, um Ladegut abzuladen.

6.3.4. Erfahrungen und Ergebnisse

Die reine Starrkörperanwendung bestehend aus Radlader, Felsbrocken und Kipplaster ist eine Standardanwendung des realisierten Systems und kann ohne großen Aufwand modelliert werden. Sie ist ein gutes Beispiel für die Fähigkeit einer hochintegrierten Mehrkörperdynamiksimulation, schnell funktionale und effektvolle Prototypen zu entwickeln. Vor allem zu Präsentationszwecken sind solche Anwendungen gefragt und äußerst „wirksam“, Messeeinsätze u.a. auf der Hannovermesse 2010 und 2011 belegen das.

Das beschriebene Verfahren für die Integration der Schüttgutsimulation konnte effizient und vor allem robust realisiert werden. Die Realitätsnähe der Simulation ist jedoch begrenzt. Zwar lässt sich das Schüttgut mit dem Radlader bearbeiten und transportieren und es gibt „spürbare“ Rückwirkungen vom Sand auf den Radlader durch die eingebrachten Zwangskräfte. Die Definition von Regelsätzen für ein realitätsnahes Verhalten der Zellautomaten im Schüttgut hat sich jedoch als schwierige und langwierige Aufgabe erwiesen.

Eine weitere Schwäche dieses Vorgehens ist die Tatsache, dass der Austausch von

kinematischen und dynamischen Informationen zwischen beiden Verfahren immer erst im nächsten Zeitschritt möglich ist. Wird die Position der Schaufel zum Zeitpunkt $t = t_0$ an die Schüttgutsimulation übertragen, kann eine resultierende Widerstandskraft erst im Zeitpunkt $t = t_0 + h$ in der Mehrkörperdynamik berücksichtigt werden. Erst in $t = t_0 + 2h$ erhält die Schüttgutsimulation dann die aktualisierte Position der Schaufel unter Einwirkung der von ihr ausgeübten Kraft zum Zeitpunkt $t = t_0 + h$. Diese nur verzögert geschlossene Schleife im Zusammenspiel mit für den interaktiven Betrieb notwendigen Zeitschrittweiten von mindestens 1/100 s. sorgt für ein nur bedingt natürliches Aussehen der Animation.

Das im nächsten Anwendungsbeispiel eingesetzte Gluing-Verfahren, das eine konsistente Verbindung von unterschiedlichen Dynamik-Simulationsverfahren erlaubt, liefert hier bessere Ergebnisse.

6.4. Entwicklung eines Virtuellen Testbeds zur Evaluierung von Laufrobotern im Explorationseinsatz

Die Exploration fremder Planeten ist ein wichtiges Feld der Weltraumforschung der näheren Zukunft. Egal ob Mond, Mars oder andere Planeten unseres Sonnensystems, um Kosten und Risiken solcher Missionen zu begrenzen, sind unbemannte Missionen für Explorationsaufgaben die erste Wahl. Vor allem auf dem Mars wurden mit den im Jahre 2003 gestarteten, radgebundenen Systemen Opportunity und Spirit der amerikanischen Weltraumbehörde NASA große Erfolge erzielt. Größte Schwäche der radgebundenen Systeme ist besonders schwer zugängliches Terrain, das sie nur schwer oder gar nicht erreichen können. Vor allem auf dem Mond werden jedoch gerade auf dem Grund von Kratern wertvolle Rohstoffvorkommen vermutet, weshalb steile Kraterhänge in den Fokus der Entwicklungen gerückt sind.

Am Deutschen Forschungsinstitut für Künstliche Intelligenz (DFKI) am Standort Bremen [2] wurde ein Konzept einer unbemannten Explorationsmission entwickelt, in der unterschiedliche Robotersysteme miteinander kooperieren („LUNARES“, [37]). Ein radgebundener Rover überbrückt relativ schnell und energieeffizient größere Distanzen weniger schwierigen Geländes. Für den Zugang zu Kratertälern trägt er einen zusätzlichen Laufroboter, der bei Bedarf vom Rover abgesetzt werden, Krater hinabsteigen

und von dort Proben zurück zum Rover bringen kann. Untersucht wurden bereits acht- („Scorpion“ [125]) und vierbeinige („ARAMIES“ [124, 61]) Systeme, aktuell steht ein sechsbeiniges System („SpaceClimber“ [15]) im Fokus der Entwicklungen.

Ziel des Projektes „Virtual Crater“[2] [143] ist es, ein Virtuelles Testbed für Laufroboter in extraterrestrischem Gelände zu schaffen. Das Virtuelle Testbed erlaubt die ganzheitliche Simulation von Explorationsmissionen in Virtueller Umgebung und eröffnet dem Versuchsleiter damit den Blick auf kommende Problemstellungen, die erst durch die Interaktion unterschiedlicher Systeme und der Systeme und ihrer Umwelt auftreten. Die Visualisierung und Interaktion in und mit der Virtuellen Realität soll dem Versuchsleiter dabei ein Maximum an Intuition ermöglichen. Trotz des ganzheitlichen Simulationsansatzes soll das Testbed die Möglichkeit bieten, durch Verfeinerungen des Simulationsmodells mittels anderer Simulationsverfahren neben der Mehrkörperdynamiksimulation auch den Blick auf spezielle Aspekte der Anwendung richten zu können. In Virtual Crater sind das vor allem die Fuß-Boden-Interaktion sowie detaillierte Motordynamikmodelle.

6.4.1. Anwendungsspezifische Anforderungen

Aus dem Projekt Virtual Crater lassen sich folgende Anforderungen an die Mehrkörperdynamiksimulation ableiten:

- Echtzeitfähige Simulation von Laufrobotern in einer realitätsnahen Umgebung.
- Verbindung der Mehrkörperdynamiksimulation mit unterschiedlichen Methoden der Bodenmechaniksimulation.
- Möglichkeit der direkten Interaktion eines Anwenders z.B. durch einen Datenhandschuh mit der Simulation. Szenarien hierfür sind das Platzieren des Roboters oder eines Steines an beliebiger Stelle in der Szene.

[2]Dieses Projekt wird gefördert durch das Deutsche Zentrum für Luft- und Raumfahrt (DLR). Förderkennzeichen 50 RA 0913.

6.4.2. Anwendungsspezifische Verfahren

Grundlagen des Gluing zur Verbindung separater Simulationsmodelle

Die Interaktion zwischen Fuß und Boden ist eine der spannendsten Fragen bei der Simulation eines Laufroboters für extraterrestrische Explorationsaufgaben und stellt bei der Entwicklung eines Virtullen Testbed daher einen der zentralen Aspekte dar. Aus dem Blickwinkel der ganzheitlichen Simulation stellen sich zwei Fragen:

1. Wie lässt sich Bodenmechanik simulieren?

2. Wie wirkt sich die Bodenmechanik auf den Roboter aus?

Die erste Frage sollte aus Sicht der Mehrkörperdynamiksimulation nebensächlich sein. Im Rahmen des Projektes sollen unterschiedliche Verfahren auf ihre Tauglichkeit hin überprüft werden und dazu an die Mehrkörperdynamiksimulation angebunden werden. Gefragt ist daher ein Verfahren, um ein (fast) beliebiges Bodenmechanik-Simulationsverfahren möglichst generisch anzubinden.

Sogenannte Gluing-Strategien leisten genau dies. Schon vor einigen Jahren durch Tseng und Hulbert [134] und Wang et al. [136] zunächst nur zur Verbindung getrennter Mehrkörpersysteme vorgestellt, werden sie heute auch zur Verbindung unterschiedlicher Simulationsverfahren eingesetzt. So verbinden Ryu et al. [112] ein Mehrkörpersystem mit einer Finite-Elemente-Methode (FEM), um die Auswirkungen eines gelenkgekoppelten Systems auf nicht starre Bauteile und umgekehrt zu bestimmen. Ihr konkretes Beispiel ist die Bestimmung der wirkenden Kräfte innerhalb einer Karosserie (FEM) unter Einfluss des Fahrwerks (Mehrkörpersystem).

Tseng und Hulbert kategorisieren Gluing-Strategien in drei unterschiedliche Gruppen. Sie bezeichnen alle kinematischen Größen, also Positionen und Orientierungen, Geschwindigkeiten sowie Beschleunigungen mit „X-Typ“-Information und alle dynamischen Größen, also Kräfte und Drehmomente, mit „T-Typ“-Informationen. Hieraus leiten sie drei Klassen von Gluing-Strategien ab, eine davon ist das T-T-Gluing. Abbildung 6.17 illustriert den Vorgang abstrakt. Etwas konkreter wird das Konzept mit einem simplen Beispiel: Angenommen, zwei Massepunkte sind mit einem virtuellen Stab so miteinander verbunden, dass sie immer konstanten Abstand voneinander haben. Jeder der Massepunkt unterliegt dem individuellen Einfluss eines Kräftevektorfeldes.

Mithilfe von T-T-Gluing lässt sich dies wie folgt simulieren: Subsystem 1 simuliert den ersten Massepunkt schlicht durch Animation der Bewegungsgleichungen unter Einfluss

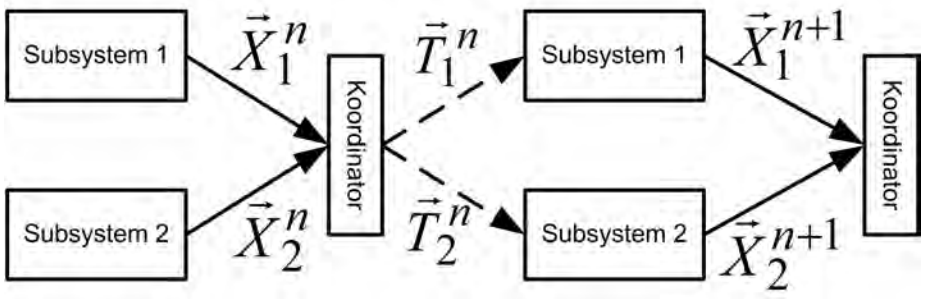


Abbildung 6.17.: Klassische T-T-Gluing-Strategie. Aus: [134]

der externen Kräfte, Subsystem 2 analog dazu den zweiten. Zunächst führen beide Subsysteme unabhängig voneinander einen Zeitschritt durch. Der Koordinator erhält die kinematischen Informationen der beiden Massepunkte - ihre Positionen - und bestimmt, wie groß der Fehler, d.h. die Differenz zwischen Ist- und dem Sollabstand ist, den der virtuelle Stab erzwingt. Aus dem Fehler wird eine Korrekturkraft geschätzt, die nötig gewesen wäre, um den Fehler im Laufe des letzten Zeitschritts nicht aufkommen zu lassen. Dann werden beide Subsysteme in ihren letzten Zustand zurückversetzt und führen den Zeitschritt erneut aus, berücksichtigen dieses Mal aber beide zusätzlich die Korrekturkraft, je in entgegengesetzter Richtung. Am Ende wird der Fehler anhand der kinematischen Informationen erneut bestimmt. Fällt er unter eine gegebene Schranke, kann mit dem nächsten Zeitschritt fortgefahren werden. Ist er noch zu groß, muss das Verfahren iterativ wiederholt und die Korrekturkraft weiter angenähert werden.

Der große Vorteil dieses Verfahrens ist seine Allgemeingültigkeit, denn es erlaubt, jede beliebige Art von Simulationsmodellen miteinander zu verbinden, die Kräfte und Momente „verarbeiten“ und kinematische Größen zurückliefern können. Sein Nachteil liegt auf der Hand: Die Aufgabe des Koordinators, die zwischen den Systemen wirkenden Kräfte zu schätzen, ist nicht trivial. Je nach Anforderung an die Genauigkeit der Lösung kann eine große Anzahl Iterationen notwendig sein und das Verfahren damit ein äußerst ungünstiges Laufzeitverhalten aufweisen. Schließlich müssen mit jeder Iteration beide Subsysteme vollständig gelöst werden. Realzeitfähige Kombinationen einer Mehrkörperdynamiksimulation mit anderen Simulationsmodellen sind daher nicht zu erwarten. Der Koordinator muss außerdem für jede neue Paarung von Subsystemen angepasst werden, da die Schätzung der ausgetauschten Kräfte detailliertes Wissen über die inneren Vorgänge in den gekoppelten Prozessen erfordert. Im Beispiel mit den zwei Massepunkten bedarf schon die Änderung der Masse einer der beiden Masse-

punkte eine entsprechende Anpassung im Koordinator.

Kooperatives Gluing zur Anbindung einer regelbasierten Bodenmechaniksimulation

Roßmann et al. [106] stellen ein Verfahren zur Simulation von Bodenmechanik auf Grundlage zellulärer Automaten vor, das zur Simulation des Untergrundes unter einem Laufroboter genutzt werden soll. Das Verfahren ähnelt dem Verfahren zur Schüttgutsimulation aus der Radlader-Anwendung. Für die Bodenmechaniksimulation kommt jedoch nur eine zweidimensionale Datenstruktur zum Einsatz: Ein Gitter unterteilt die Oberfläche in eine feste Anzahl Zellen. Jede Zelle enthält als wesentliche Eigenschaft nur ihre gegenwärtige Höhe. Sie agiert autonom mit ihren direkten Nachbarzellen und befolgt dabei eine Menge einfacher Regeln, die die Einhaltung einer maximalen Oberflächenkrümmung und eines maximalen Böschungswinkels sicherstellen. Das Verfahren selbst hat zunächst keinen direkten Bezug zu Kräften und Momenten. Pla-Castells et al. [99] beschreiben jedoch, wie Kraftgrößen in einem verwandten Verfahren durch „kraftäquivalente Höhen“ in den Zellen berücksichtigt werden können.

Damit lässt sich das Verfahren prinzipiell mithilfe einer T-T-Gluing-Strategie mit der Starrkörperdynamiksimulation verbinden. Eine interaktive oder gar echtzeitfähige Realisierung des Gluing-Mechanismus für beliebige Modelle ist aber kaum erreichbar. Am Beispiel eines Laufroboters ist das Problem leicht nachvollziehbar: Ein externer Koordinator zwischen Mehrkörperdynamik und Bodenmechanik kann nicht wissen, was der Fuß trägt, der auf die Oberfläche eines Bodenmechanikmodells tritt. Daher ist eine Schätzung allein der Größenordnung der auftretenden Kontaktkräfte ein Ratespiel. Hierdurch wird eine große Anzahl Iterationen nötig, bevor der kinematische Fehler eine gegebene Fehlerschranke unterschreitet.

Somit bietet es sich an, das Wissen der Mehrkörperdynamiksimulation über das, was der Fuß trägt, während der Koordination zu nutzen. Eine neu entwickelte Variante des T-T-Gluing geht dazu wie folgt vor:

1. Der Fehler in den kinematischen Bedingungen zwischen Bodenmechanik und Mehrkörperdynamiksimulation wird ausgewertet - ein Beispiel hierfür ist die Größe der Durchdringung zwischen Fuß und Oberfläche des Bodens.

2. Sind die kinematischen Zwangsbedingungen nicht verletzt bzw. liegt der Fehler unter einer gegebenen Schranke, müssen keine Kontaktkräfte bestimmt bzw. an-

gepasst werden und beide Subsysteme können unabhängig voneinander einen Zeitschritt durchführen. Die Iteration endet. Sind die kinematischen Bedingungen verletzt, d.h. berührt oder durchdringt ein Starrkörper die Bodenoberfläche, werden entsprechende Kontaktnormalenzwangsbedingungen in der Mehrkörpersimulation formuliert, so als würden die betreffenden Starrkörper auf einen starren Untergrund stoßen.

3. Das Mehrkörpersystem wird einmalig gelöst, d.h. die Beträge aller Zwangskräfte bzw.- impulse werden bestimmt. Der Zeitschritt wird jedoch noch nicht ausgeführt, die Zwangskräfte haben also noch keinen Einfluss auf die Simulation.

4. Der Koordinator nutzt die Kontaktnormalenzwangskräfte, die durch einen starren Kontakt hervorgerufen würden, als Grundlage für seine erste Schätzung der tatsächlichen Kontaktkräfte zwischen Bodenmechanik- und Mehrkörpermodell. Diese erste Schätzung fließt in beide Subsysteme ein, in die Mehrkörperdynamiksimualtion als externe Kraft (im Vektor $\vec{f}$, vgl. Gl. 3.21 auf Seite 86), in die Bodenmechaniksimulation in Form einer „kraftäquivalenten Höhe“.

5. Beide Subsysteme werden unter Berücksichtigung der ersten Schätzung der Kontaktkräfte erneut gelöst. Sind die kinematischen Bedingungen hiernach immernoch verletzt, beginnt die nächste Iteration bei Schritt 4. Andernfalls endet der Prozess.

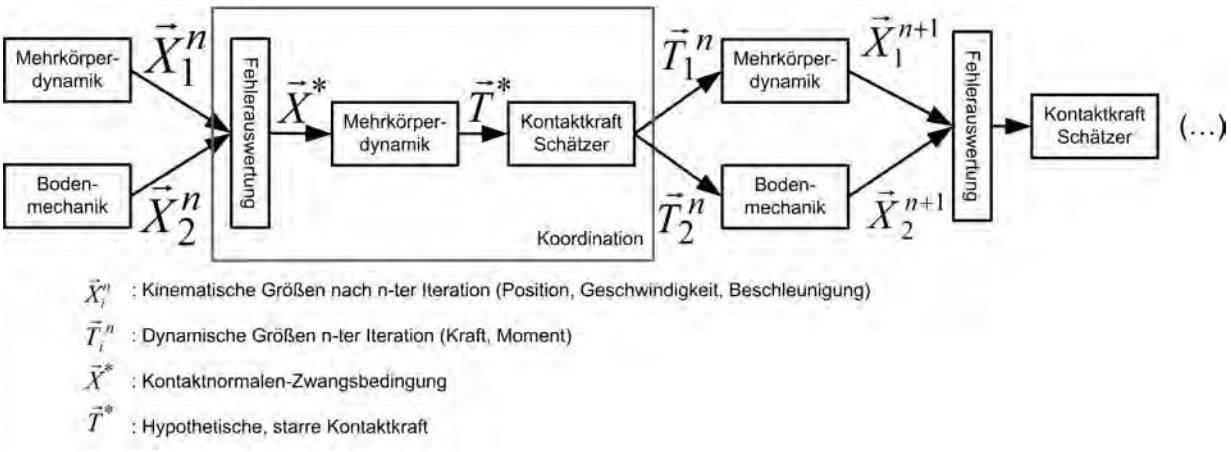


Abbildung 6.18.: Kooperatives T-T-Gluing zur Kopplung von Bodenmechanik- und Mehrkörperdynamiksimulation

Abbildung 6.18 verdeutlicht den Ablauf grafisch. Da bei der realisierten Implementierung die Komponenten „Fehlerauswertung“ und „Kontaktkraftschätzer“ Teil der Bibliothek der Bodenmechaniksimulation sind, entstand die Bezeichnung „kooperatives T-T-Gluing“. Zur einfachen Bestimmung der Kontaktgeometrien zwischen Füßen und Untergrund wurde als Fußform die Kugel gewählt.

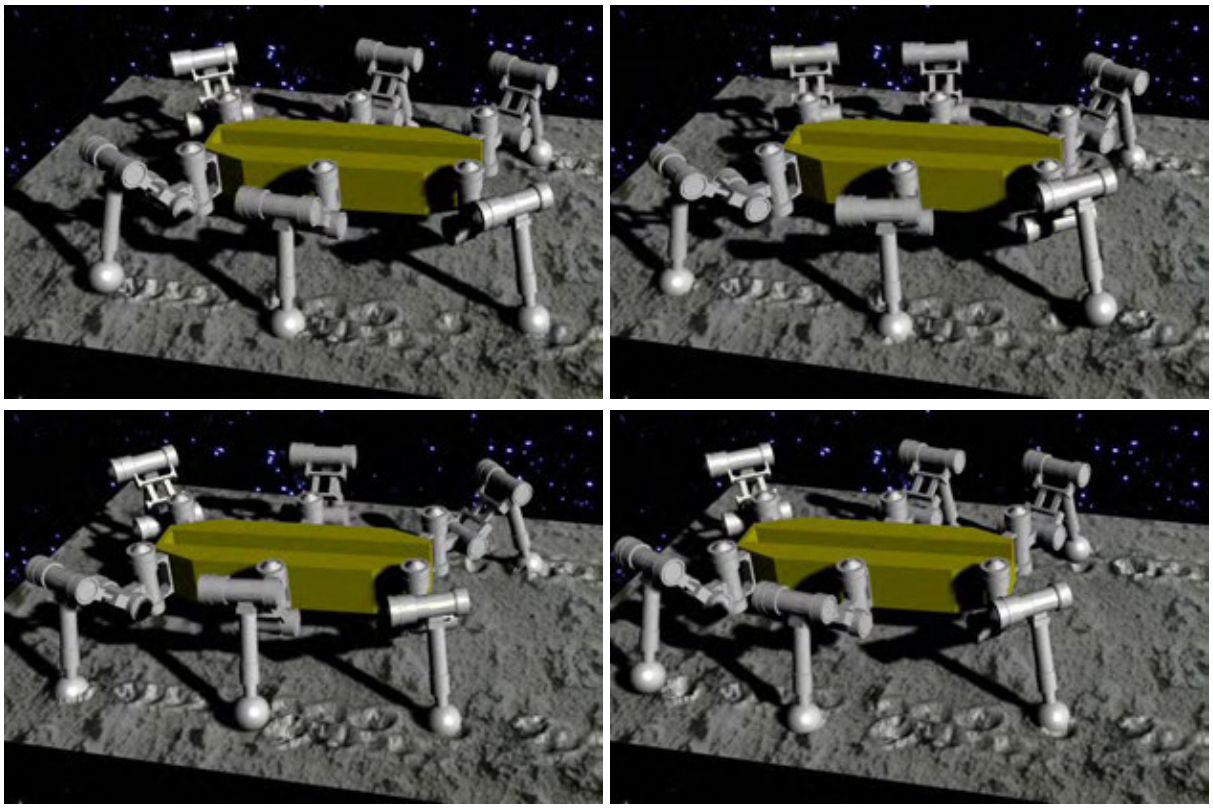


Abbildung 6.19.: Kopplung von Bodenmechanik- und Mehrkörperdynamiksimulation durch kooperatives T-T-Gluing

Abbildung 6.19 zeigt eine Sequenz der gemeinsamen Simulation von regelbasierter Bodenmechanik- und Mehrkörperdynamiksimulation. Deutlich zu sehen ist die „Verformung“ des Untergrundes und dass der Roboter leicht in die Oberfläche einsinkt, anstatt starr aufzuliegen.

Realisierung eines direkten Interaktionsmechanismus zwischen Anwender und Mehrkörperdynamiksimulation

Bei den bisher vorgestellten Anwendungen trat der Anwender nur indirekt mit der Mehrkörperdynamiksimulation in Interaktion, typischerweise durch Vorgabe von Sollge-

schwindigkeiten von Motoren oder Hydraulikzylindern in Maschinensimulationen. Möchte man dem Anwender ermöglichen, ganz direkt einzugreifen, z.B. um mit dem Datenhandschuh dem Laufroboter einen Stein in den Weg zu legen, wird ein Interaktionsmedium zur vollständigen, direkten Kontrolle von Starrkörpern notwendig. Diese Aufgabe lässt sich in zwei Teilaufgaben zerlegen:

1. Ein Starrkörper in der Simulation soll durch Vorgabe entweder von Geschwindigkeiten in allen sechs Raumrichtungen direkt bewegt oder durch Vorgabe von Position und Orientierung direkt platziert und orientiert werden können.

2. Zum Greifen von Gegenständen ist ein Mechanismus erforderlich, der es dem Anwender ermöglicht, einen beliebigen Körper der Mehrkörpersimulation fest an den durch ihn direkt kontrollierten Körper zu „heften“.

Direkte 6D-Kontrolle eines Starrkörpers Die direkte Kontrolle eines Starrkörper in allen sechs Raumrichtungen lässt sich durch entsprechende Zwangsbedingungen realisieren. Hierzu wurde die Klasse `RBDRigidBodyDirectControl` eingeführt, die die direkte Steuerung eines Starrkörpers auf Grundlage von Zwangsbedingungen im Raum ermöglicht. Da sie kein Gelenk im engeren Sinne darstellt, das zwei Körper in Relation setzt, sondern sich auf nur einen einzelnen Körper bezieht, ist sie direkt von der Klasse `RBDConstraintResource` abgeleitet. Bei der Modellierung stehen folgene Optionen zur Auswahl: Das Koordinatensystem zur Referenzierung der Zwangsbedingungen kann raumfest angenommen oder extern festgelegt werden. Letztere Variante bietet sich an, um dem Anwender zu ermöglichen, Bewegungen entsprechend der aktuellen Kameraperspektive ausführen zu können. Zur Vorgabe von Sollgeschwindigkeiten bestehen zwei Varianten: Die Sollgeschwindigkeiten können direkt vorgegeben werden, dies bietet sich bei der Nutzung einer 3D-Maus[3] als Interaktionsgerät an, da ihre Ausgaben optimal als Sollgeschwindigkeiten interpretiert werden können. Kommt ein Trackingsystem zum Einsatz, das direkt eine Position und Orientierung vorgibt, kann der Instanz von `RBDRigidBodyDirectControl` ein beliebiger Knoten des Modells vorgegeben werden, dessen Koordinatensystem sie als Sollwert interpretiert. Im Verlauf der Simulation versucht `RBDRigidBodyDirectControl` dann, Ankerpunkt und Orientierung des Körpers, auf den sie sich bezieht, entsprechend des Sollwert-Koordinatensystems auszurichten. Über einen PID-Regler bestimmt sie hierzu eine translatorische und eine rotatorische

[3] 6D-Eingabegerät, z.B. SpaceNavigator™ des Herstellers 3DConnexion

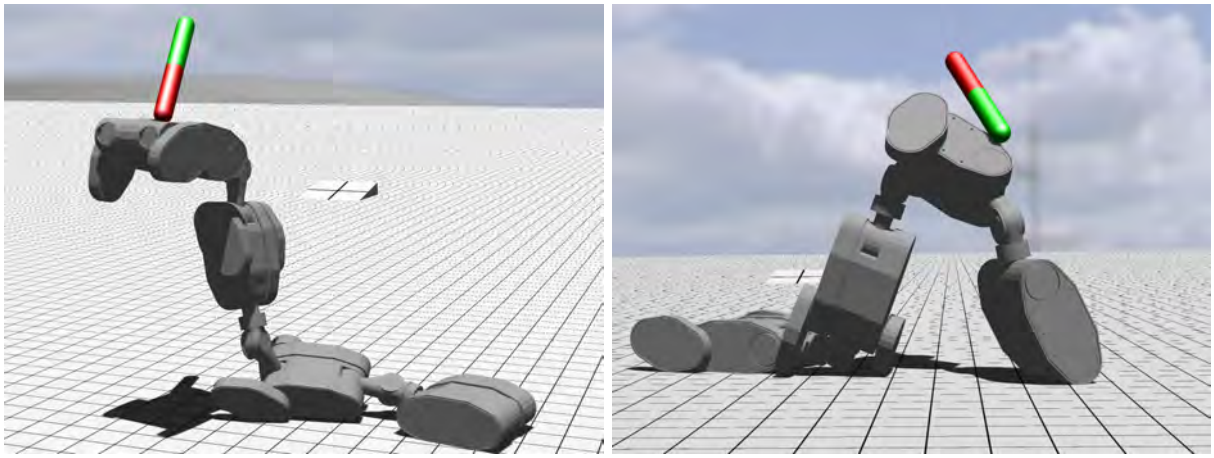


Abbildung 6.20.: Direkte Interaktion zwischen Anwender und Mehrkörpersimulation, 1-2: Hier mit der 3D-Maus-gesteuerten Magnet-Metapher

Geschwindigkeit, die die Fehler zwischen Ist- und Sollposition und -orientierung ausgleichen und als rechte Seiten der geschwindigkeitsbezogenen Zwangsbedingungen eingesetzt werden.

Seien in beiden Fällen $\vec{x}, \vec{y}, \vec{z}$ die drei Basisvektoren des Referenzkoordinatensystems. Sei außerdem $\vec{r}$ der Vektor vom Schwerpunkt des Körpers zum Ankerpunkt der Direktsteuerung. Die Einträge in den rotatorischen Teil-Jacobimatrizen entsprechen denen eines Lineargelenks (vgl. Kapitel 3.2.6) für einen Körper:

$$\mathbf{J}_{\mathrm{direct,rot},i} = \begin{pmatrix} 0 & 0 & 0 & x_1 & x_2 & x_3 \\ 0 & 0 & 0 & y_1 & y_2 & y_3 \\ 0 & 0 & 0 & z_1 & z_2 & z_3 \end{pmatrix} \tag{6.4}$$

Analog dazu entsprechen die Einträge in den translatorischen Teil-Jacobimatrizen z.B. denen in der Radaufhängung (vgl. Kapitel 3.4.4):

$$\mathbf{J}_{\mathrm{direct,tra},i} = \begin{pmatrix} x_1 & x_2 & x_3 & -(\vec{x} \times \vec{r})_1 & -(\vec{x} \times \vec{r})_2 & -(\vec{x} \times \vec{r})_3 \\ y_1 & y_2 & y_3 & -(\vec{y} \times \vec{r})_1 & -(\vec{y} \times \vec{r})_2 & -(\vec{y} \times \vec{r})_3 \\ z_1 & z_2 & z_3 & -(\vec{z} \times \vec{r})_1 & -(\vec{z} \times \vec{r})_2 & -(\vec{z} \times \vec{r})_3 \end{pmatrix} \tag{6.5}$$

Die Ausgestaltung der rechten Seiten der Zwangsbedingungen hängt von der Quelle der Geschwindigkeitsvorgaben ab. Werden die Geschwindigkeiten z.B. durch eine 3D-Maus vorgegeben, dienen deren Ausgabewerte $\vec{v}_{\mathrm{soll}}, \vec{\omega}_{\mathrm{soll}}$ direkt als Sollgeschwin-

digkeiten und damit als rechte Seite:

$$\begin{aligned} \vec{b}_{\text{trans}} &= \vec{v}_{\text{soll}} \\ \vec{b}_{\text{rot}} &= \vec{\omega}_{\text{soll}} \end{aligned} \tag{6.6}$$

Soll der Ankerpunkt des bezogenen Starrkörpers eine vorgegebene Position und Orientierung erreichen, müssen diese beiden Größen zunächst bestimmt werden. Gefragt sind translatorisch und rotatorisch jeweils eine Richtung und ein Betrag.

Mit einer Proportionalitätskonstante c ergeben sich Richtung und Betrag einer Korrekturgeschwindigkeit für die translatorischen Zwangsbedingungen direkt aus dem Fehler zwischen Ist- ($\vec{p}_{\text{ist}}$) und Sollposition ($\vec{p}_{\text{soll}}$) des Ankerpunkts:

$$\vec{v}_{\text{soll}} = c \cdot (\vec{p}_{\text{soll}} - \vec{p}_{\text{ist}}) \tag{6.7}$$

Für die rotatorische Zwangsbedingung wird eine Winkelgeschwindigkeit $\vec{\omega}$ gesucht, mit deren Hilfe der Starrkörper aus seiner Istorientierung $\vec{q}_{\text{ist}}$ in seine Sollorientierung $\vec{q}_{\text{soll}}$ gedreht wird. Hierzu leitet man zunächst ein Rotationsquaternion $\vec{\Phi} \in \mathbb{R}^4$ her, dass eben diese Rotation ausführt:

$$\begin{aligned} \vec{\Phi} \cdot \vec{q}_{\text{ist}} &= \vec{q}_{\text{soll}} \\ \vec{\Phi} &= \vec{q}_{\text{soll}} \cdot \vec{q}_{\text{ist}}^{-1} \end{aligned} \tag{6.8}$$

Bringt man diese Rotation $\vec{\Phi}$ in die Drehvektor-Drehwinkel-Darstellung $< \vec{\alpha}, a >$ (siehe Anhang A.5), so hat man mit dem Drehvektor $\vec{\alpha}$ eine Richtung und mit dem Drehwinkel a einen Betrag der gesuchten Winkelgeschwindigkeit $\vec{\omega}$ zur Verfügung, die die Soll- in die Istorientierung überführt. Setzt man einen simplen P-Regler mit Proportionalitätskonstante c an, ergeben sich die rechten Seiten der Zwangsbedingungen zu:

$$\begin{aligned} \vec{b}_{\text{trans}} &= c \cdot \vec{v} \\ &= c \cdot (\vec{p}_{\text{soll}} - \vec{p}_{\text{ist}}) \\ \vec{b}_{\text{rot}} &= c \cdot \vec{\omega} \\ &= c \cdot a \cdot \vec{\alpha} \end{aligned} \tag{6.9}$$

Mechanismus für das vereinfachte Greifen von Körpern der Simulation Nachdem mit dem Mechanismus für die direkte Kontrolle ein beliebiger Starrkörper z.B. mit einem

Abbildung 6.21.: Direkte Interaktion zwischen Anwender und Mehrkörpersimulation, 2-2: Hier mit einem positionsgetrackten Datenhandschuh CyberGlove®

getrackten Datenhandschuh direkt positions- und orientierungsbezogen im Raum ausgerichtet werden kann, soll natürlich auch ein Greifen von Gegenständen ermöglicht werden.

Hierzu dient die Erweiterung `ExtensionMagnetic` der VEROSIM®-Schnittstellenbibliothek VSPluginRBDynamX, mit der ein beliebiger Starrkörper der Simulation um die Funktion erweitert werden kann, bei Berührung eines anderen Körpers mit diesem eine starre Verbindung einzugehen. Die Realisierung ist dank Kontaktgraphenanalyse denkbar einfach und gleichzeitig effizient: Der Mechanismus reagiert auf Kollisionsereignisse, die von der Mehrkörperdynamiksimulation ausgesandt werden. Tritt eine Kollision mit dem Körper auf, der durch den ExtensionMagnetic-Mechanismus erweitert wurde, so wird ein starres Gelenk (`RBDJointRigid`) angelegt. Der nachfolgend ausgelöste Aufruf der Kontaktgraphenanalyse zur Bestimmung der Menge aller Starrkörper findet diese starre Verbindung und verschmilzt die betroffenen Körper zu einem logischen Starrkörper der Simulation.

Die Abbildungen 6.20 und 6.21 zeigen einige Eindrücke der direkten Interaktion eines 3D-Maus-gesteuerten „Magneten“ mit einem mobilen Roboter und der direkten Interaktion zwischen getracktem Datenhandschuh und verschiedenen Elementen des virtuellen Testbeds für Laufroboter in der VR-Umgebung des Instituts für Mensch-Maschine-Interaktion (MMI) der RWTH Aachen.

6.4.3. Anwendung generischer Verfahren und Modellierung

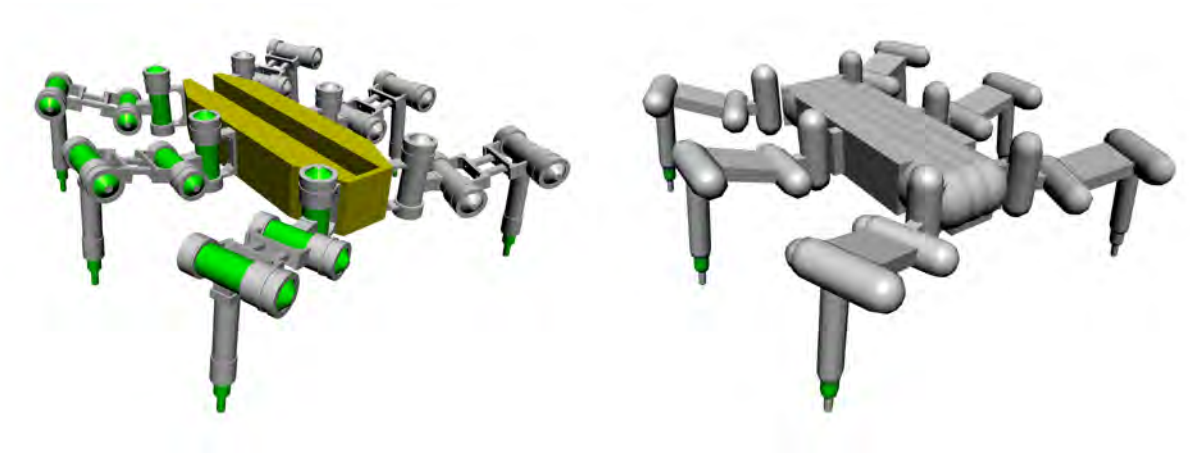


Abbildung 6.22.: Visualisierungs- und physikalisches Ersatzmodell des DFKI [2] Scarabaeus Laufroboters

Die Modellierung des reinen Starrkörpermodells beinhaltet keine wesentlichen Besonderheiten. Abbildung 6.22 zeigt Visualisierungsmodell (links) und physikalisches Ersatzmodell (rechts). Alle Gelenke des Scarabaeus Robotermodells sind mit Standard-Positionsstellern vom Typ `ExtensionServoMotor` ausgestattet. Hieran kann direkt eine existierende Steuerung angebunden werden. Durch diese Steuerung, auch Laufmustergenerator genannt, kann das Simulationsmodell komplexe Laufmuster abarbeiten.

6.4.4. Experimente im Virtuellen Testbed

Durch die direkte Zugriffsmöglichkeit auf die Parameter und den Zustand der Mehrkörperdynamiksimulation an der Benutzeroberfläche von VEROSIM® können interessante Fehlersimulationen durchgeführt werden. Abbildung 6.23 zeigt eine Sequenz der Simulation des Laufroboters mit einer Zeitspanne von fünf Sekunden. Zu Beginn wurde die Eigenschaft „constMaxTorque“, also das maximale Drehmoment der beiden horizontal ausgerichteten Motoren am linken vorderen Bein, von „3“ auf „0“ ([Nm]) gesetzt. Dies entspricht einem totalen Drehmomentverlust am Motor, z.B. durch eine Unterbrechung der Stromzufuhr. Das erwartete Symptom ist das Einknicken der betroffenen Gelenke. Die Sequenz zeigt die Reaktion des simulierten Roboters auf den Eingriff. Durch folgende Modellparameter kann zusätzlich eine verbliebene Motorreibung nachgebildet werden: Durch Entfernung des Reglers der Positionssteller erhält man geschwindig-

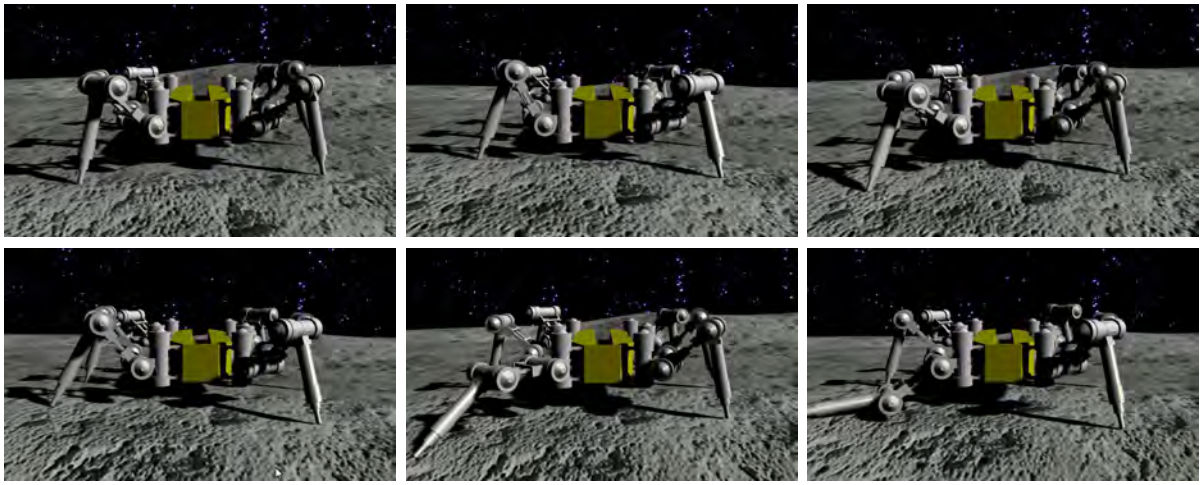


Abbildung 6.23.: Simulation des vollständigen Drehmomentverlusts in zwei Motoren

keitsgestellte Motoren. Eine Sollgeschwindigkeit von Null und ein kleines maximales Drehmoment von z.B. 0,1 [Nm] resultieren in einem geringen Widerstandsmoment der defekten Motoren.

Weil es sich bei den im Laufroboter eingesetzten Motoren um solche Typen handelt, die durch Zwangsbedingungen realisiert werden, also Instanzen von `ExtensionVelocityBasedMotor` oder `ExtensionServoMotor`, wird der Wert „constMaxTorque“ genutzt, um die Grenzwerte für die zugehörigen Lagrange-Multiplikatoren zu bestimmen.

Ein anderer Fehlertyp ist der Totalverlust eines Gelenks durch einen mechanischen Bruch. Entsprechend der Idee der Extensions in VEROSIM® (vgl. Kapitel 5.3) kann dies sehr einfach nachgestellt werden, indem die entsprechende Gelenk-Extension an der VEROSIM®-Oberfläche aus der Modelldatenbank gelöscht wird. Damit verschwindet die Eigenschaft, dass es sich bei einem Knoten um ein Gelenk handelt. Dies kann problemlos zur Laufzeit der Simulation – interaktiv – durchgeführt werden. Abbildung 6.24 zeigt die Reaktion der Simulation hierauf. Offensichtlich kann man beobachten, wie der Roboter mit dem Verlust eines Beines umgeht. Viel interessanter jedoch ist die Tatsache, dass das „verlorene Bein“ zu einem problematischen Hindernis für ein nachfolgendes Bein werden kann. Dies ist ein typisches Beispiel für den Mehrwert der Simulation im Virtuellen Testbed: Eine Detailsimulation, die von vornherein den Fokus auf das offensichtliche Problem des fehlenden Beines legt, offenbart nicht ein solches, sekundäres Problem.

Ein dritter möglicher Fehlerfall ist der Totalverlust eines Gelenks durch Blockade.

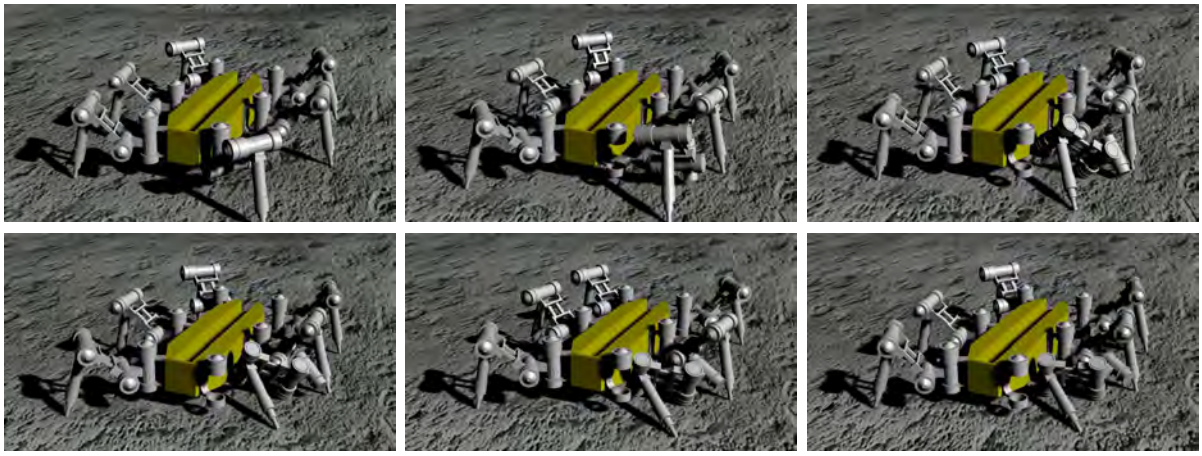


Abbildung 6.24.: Simulation eines Gelenkbruchs

Für diesen Fall wurde eine spezielle Eigenschaft „isBlocked“ (ist blockiert) an der Gelenk-Extension vorgesehen. Abhängig vom Wert dieser Eigenschaft werden die Zwangsbedingungen, die durch das Gelenk impliziert werden, eine komplett starre Verbindung zwischen den beteiligten Körpern herstellen. Das bestehende Drehgelenk (`RBDJointHinge`) wird praktisch durch ein starres Gelenk (`RBDJointRigid`) ersetzt. Alternativ kann auch eine nur teilweise Blockade simuliert werden, indem der Motor am Gelenk derart modifiziert wird, so dass er eine konstante Sollgeschwindigkeit von Null und wiederum ein begrenztes maximales Drehmoment besitzt. So könnte das Gelenk unter Einfluss äußerer Kräfte noch bewegt werden, hat selber jedoch lediglich verzögernden Einfluss auf die Bewegung.

Abbildung 6.25 zeigt die Simulation einer vollständigen Blockade des markierten Gelenks. Die Sequenz deckt eine Zeitspanne von 45 Sekunden ab. Die Kamera hat eine fixe Orientierung, steht aber immer senkrecht über dem Roboter. Schön zu sehen ist hier, wie der Geradeausgang des Systems durch diesen Fehler beeinträchtigt wird.

Vorgenannte Fehlermodelle erlauben die Simulation wesentlicher Aspekte realer Fehlerfälle. Die Beispiele zeigen, wie einfach die Simulationen solcher Fehlerfälle im Virtuellen Testbed ohne weitere Anpassungen am Simulationsmodell möglich sind. Gerade in frühen Phasen der Entwicklung z.B. von Laufstrategien sind solche Fehlersimulationen sehr gut und einfach einsetzbar, um ein intuitives Gefühl für die Fähigkeiten eines neuen Entwurfs zu vermitteln.

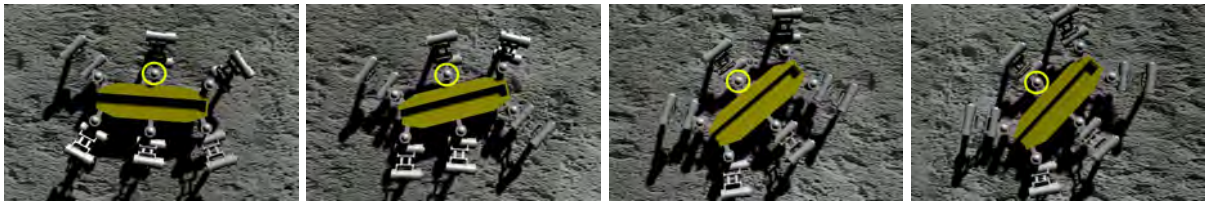


Abbildung 6.25.: Simulation eines blockierten Gelenks

6.4.5. Erfahrungen und Ergebnisse

Die Idee des Virtuellen Testbed ist für eine abschließende Bewertung noch zu jung. Aus dem Ansatz hat sich jedoch ein neues Konzept zur Erstellung von Simulationsmodellen entwickelt. Die Entwicklung von Simulationsmodellen technischer Systeme verläuft häufig entlang einer Bottom-Up Strategie. Die Entwicklung beginnt bei einem Detail des Gesamtsystems, z.B. dem Simulationsmodell für einen neu entwickelten Aktuator. Nach und nach werden Detailmodelle zu einem Gesamtsystem zusammengesetzt: Das Aktuatormodell fließt in ein Mehrkörpersystem ein, das Mehrkörpersystem wird Teil einer Virtuellen Welt. Tatsächlich jedoch werden auf diesem Wege nur selten zusammengesetzte Simulationsmodelle in einer vollständigen Virtuellen Welt erstellt. Tests am Gesamtsystem werden typischerweise erst in realen Versuchsumgebungen durchgeführt.

Im Rahmen des Projektes Virtual Crater wird ein anderes Konzept verfolgt: Entwicklung von Simulationsmodellen entsprechend einer Top-Down-Strategie [107, 105]. Um von Beginn an die Einflüsse einer komplexen Umgebung auf ein technisches System beurteilen zu können, beginnt die Entwicklung eines Simulationsmodells mit einem Gesamtmodell des Systems und seiner Umgebung, im konkreten Beispiel auf Grundlage der Mehrkörperdynamiksimulation. Um dennoch den Fokus auch auf spezielle Details richten zu können, kann das Simulationsmodell an Stellen von besonderem Interesse verfeinert werden. Eine solche Verfeinerung findet z.B. durch Anbindung von detaillierten Motordynamikmodellen mittels PowerCoupling oder einer Bodenmechaniksimulation mittels des hier vorgestellten Gluings statt.

6.5. Weitere realisierte Anwendungen

Die Vielseitigkeit der hier vorgestellten Mehrkörperdynamiksimulation zeigt sich auch daran, dass sie nicht nur in Anwendungen eingesetzt wird, in der sie selbst ein Hauptzweck ist, sondern vielmehr als eine Art Basisfunktionalität vielen Anwendungen zu Gu-

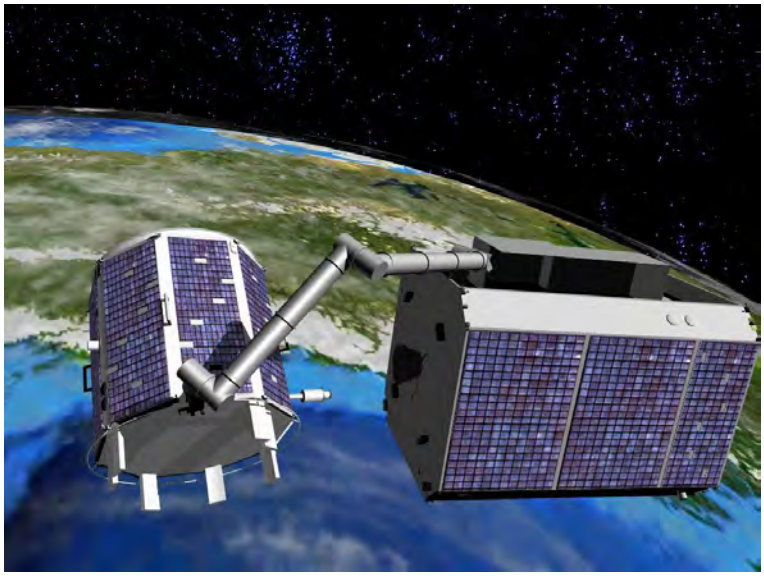


Abbildung 6.26.: Das TECSAS / DEOS Projekt des Deutschen Zentrums für Luft- und Raumfahrt

te kommt. Dieses Kapitel beschreibt daher zwei weitere Anwendungen, die von der einfachen Verfügbarkeit einer robusten Mehrkörperdynamiksimulation in VEROSIM® profitieren.

6.5.1. Steuerungsentwicklung in der Weltraumrobotik

Ein wichtiges Anwendungsgebiet der zukünftigen Weltraumrobotik ist die Wartung und Reparatur künstlicher Satelliten in den Erdumlaufbahnen. Bis in die jüngste Vergangenheit konnten solche Operationen ausschließlich durch den amerikanischen Spaceshuttle durchgeführt werden. Mit seinem Roboterarm kann es Satelliten einfangen und damit einem Astronauten als fixes Arbeitsobjekt - fix in Relation zum Shuttle - zur Verfügung stellen. Der Einfangprozess eines frei im Raum schwebenden Satelliten ist jedoch kompliziert. Um starke Beschleunigungen zu vermeiden, muss die Bewegung des Roboterarms an die Bewegung eines Greifpunktes am zu fangenden Satelliten angepasst werden. Außerdem müssen die Massenverhältnisse und die Rückwirkungen des Roboterarms und der Trägheit des zu fangenden Satelliten auf die Dynamik des Shuttles berücksichtigt werden.

Die Problematik verschärft sich weiter, wenn man an den Einsatz unbemannter, im Vergleich mit dem relativ großen und schweren Shuttle deutlich kleinerer Servicerobo-

tersatelliten denkt. Die Massenrelationen zwischen den Teilen des Roboterarms und seiner frei schwebenden Plattform verändert sich mehr in Richtung des Arms, ebenso die Relationen zwischen dem fangenden (engl.: *Chaser*) und dem zu fangenden Satelliten (engl.: *Target*). Die dynamischen Rückwirkungen auf den *Chaser* haben wachsenden Einfluss auf dessen Bewegungsverhalten. Rein kinematisch basierte Steuerungen können den Greifer nicht erfolgreich platzieren, weil sich die Ausrichtung der Basis des Roboters als Reaktion auf seine Bewegung verändert und klassische, kinematische Strategien diese Veränderung der Basisausrichtung nicht berücksichtigen.

Das Forschungsprojekt TECSAS/DEOS [41] des Deutschen Zentrums für Luft- und Raumfahrt (DLR) behandelt dieses Problem. Ziel ist es, anhand zweier Versuchssatelliten Greifstrategien zwischen zwei unbemannten Satelliten zu erproben, vgl. Abbildung 6.26. Das Szenario wurde im Rahmen einer Forschungsarbeit durch Kaigom [71] am Institut für Mensch-Maschine-Interaktion auf Grundlage der hier vorgestellten Dynamiksimulation in VEROSIM® nachgebildet, um unterschiedliche Optimierungsalgorithmen zur Findung optimaler Trajektorien zur Platzierung des Greifers zu erproben. Die Algorithmen minimieren den Fehler zwischen einer Soll- und einer erreichten Istausrichtung des Greifers an der Spitze des Armes nach Abfahren der zu bewertenden Trajektorie.

Zwei Varianten wurden untersucht: Die erste Variante bewertet lediglich die Abweichung des Greifers zu seiner Sollausrichtung und ignoriert die Auslenkung der Basis des Roboters aus seiner Ausgangsorientierung. Ergebnis ist eine Trajektorie, bei der der Greifer zwar seine Sollausrichtung erreicht, bei der die Basis des Systems jedoch ebenfalls ihre Ausgangsposition und -orientierung verlässt. Abbildung 6.27 zeigt eine Sequenz einer solchen Trajektorie.

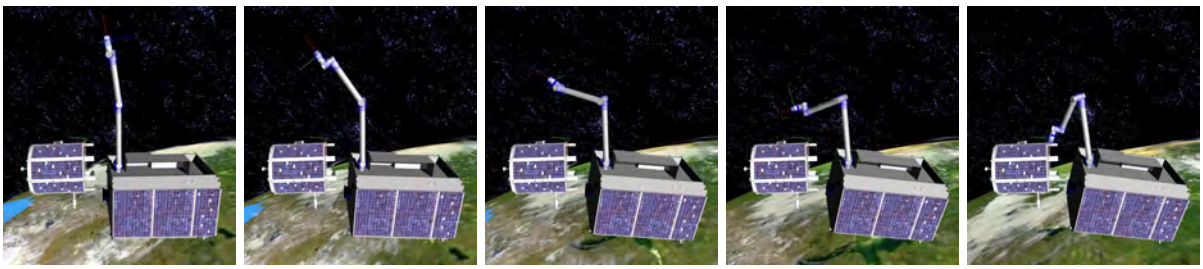


Abbildung 6.27.: Eine optimale Trajektorie des DEOS Roboterarms ohne Berücksichtigung der Roboterbasis

Die zweite Variante versucht zusätzlich, die Orientierung der Basis innerhalb vorgegebener Grenzen zu halten, also zum Beispiel nicht mehr als maximal +/-3° Ab-

weichung um jede Achse zuzulassen. Diese Anforderung ergibt sich zum Beispiel aus einer optimalen Ausrichtung von Solarzellen in Richtung der Sonne. Die zu optimierende Fehlerfunktion ist offensichtlich komplizierter, die Optimierung, d.h. das Finden einer geeigneten Trajektorie dauerte auf einem Standard-PC mitunter mehrere Tage. Die Resultate zeigten jedoch häufig sehr eindrucksvolle, weil unerwatet komplexe Ergebnisse. Abbildung 6.28 zeigt die Sequenz einer entsprechenden, optimalen Trajektorie.

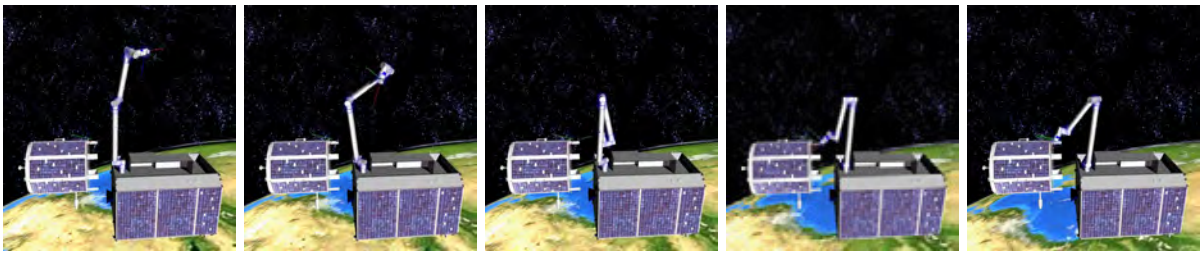


Abbildung 6.28.: Eine optimale Trajektorie des DEOS Roboterarms unter Berücksichtigung einer maximalen Abweichung der Basis um je drei Grad um jede Raumachse

Der Einsatz der Mehrkörperdynamiksimulation bietet eine praktische und problemlose Grundlage zur Erprobung der Optimierungsalgorithmen. Die Anwendung nutzt ausschließlich Standardfunktionalitäten und bedurfte daher keinerlei Erweiterungen an der Mehrkörperdynamiksimulation. Aufgrund der aus ihrer Sicht relativen Einfachheit des Modells mit seinen sieben Standard-Drehgelenken und (meist) keinen Kontakten, kann das Modell schneller als in Echtzeit simuliert werden, auch indem z.B. nur jeder tausendste Zeitschritt der Dynamiksimulation visualisiert wird. Hierdurch konnten im konkreten Fall viele zehntausend Trajektorien mit der Dynamiksimulation ausgewertet werden.

6.5.2. Fahrzeuganimation in einem 3D-Geoinformationssystem (3D-GIS)

Ein neues Anwendungsfeld für VEROSIM® sind GIS-Anwendungen. GIS steht für **G**eo-**I**nformations-**S**ystem. Hierzu gehört u.a. die Städte- und Landschaftsvisualisierung. Die zu visualisierenden Modelle zeichnen sich vor allem durch ihren großen Umfang und die hohe Detailtreue aus. Mehrkörperdynamik zur Simulation ansprechender, interaktiv steuerbarer Fahrzeuge innerhalb solcher Modelle eignet sich außerordentlich gut, um

den Betrachtern ein realitätsnahes Erleben des Geo-Modells zu ermöglichen: Die Perspektive wie auch das Bewegungsverhalten sind vertraut, anders als bei nur in Virtueller Realität möglichen, gänzlich uneingeschränkten „3D-Maus-Rundflügen“.

Abbildung 6.29.: Screenshots von Fahrzeugsimulationen in unterschiedlichen GIS-Anwendungen

Wesentliche Grundlage für die Visualisierung beliebig großer Landschafts- und Stadtmodelle in VEROSIM® ist ein Mechanismus für das dynamische Laden und Entladen der relevanten Modellteile aus einer externen Datenbank [109]. Entsprechend dem geladenen Modellausschnitt wird auch in der Dynamiksimulation die Geometrie des Starrkörpers, der die Umgebung repräsentiert, stetig aktualisiert.

Abbildung 6.29 zeigt einige Screenshots aus GIS-Anwendungen mit Fahrzeugsimulationen. In der oberen Reihe von links nach rechts: Interaktive Visualisierung und Simulation eines Modells der Stadt Düsseldorf, ein Geländefahrzeug im Virtuellen Wald [108] und die dynamikbasierte Simulation eines Karnevalszugs bestehend aus Traktoren und Anhängern in einem Modell der Stadt Köln.

Unten links und unten Mitte zwei Screenshots einer interaktiven, dynamikbasierten Simulation eines Modells der Stadt Stuttgart. Die Abbildungen zeigen, dass mehrere Fahrzeuge in einem Modell problemlos möglich sind. Mehrere, unabhängig voneinander steuerbare Ansichtsfenster in VEROSIM® (u.r.) erlauben es auch mehreren Bedienern, gleichzeitig in einer Szene miteinander zu interagieren, so wie es in vielen Autorenn-

spielen üblich ist.

Die zahlreichen hier vorgestellten Anwendungen des Mehrkörperdynamiksimulationssystems, das aus den Entwicklungen dieser Arbeit hervorgegangen ist, vermittelt einen Eindruck des gesamten Anwendungsspektrums des Systems. Seine Flexibilität und Leistungsfähigkeit sowie die unterschiedlichen Erweiterungsmöglichkeiten um anwendungsspezifische Heuristiken, um die Fähigkeit zur Interaktion mit anderen Simulationsverfahren und zum Umgang mit sich verändernden Virtuellen Welten, machen es zu einer robusten, leistungsfähigen, vielseitigen und gut erweiterbaren Grundlage für die Anwendungsentwicklung.

Kapitel 7.

Zusammenfassung der Ergebnisse und Ausblick

7.1. Zusammenfassung der Ergebnisse

In dieser Arbeit werden Grundlagen, Realisierung, systematische Integration und Optimierung einer Mehrkörperdynamiksimulation mit modernden Methoden und in modernen Strukturen der Virtuellen Realität vorgestellt. Aus den Entwicklungen geht ein System hervor, dessen Leistungsfähigkeit an unterschiedlichsten Anwendungen wie Arbeitsmaschinensimulatoren für die Aus- und Weiterbildung, Virtuellen Testbeds im Bereich der Weltraumrobotik und sogar an GIS-Anwendungen belegt wird.

Im Hinblick auf das Verfahren zur Bestimmung der inversen Dynamik wird detailliert die Formulierung von Zwangsbedingungen aller Art beschrieben. Hierzu zählen sowohl klassische Gelenkverbindungen als auch erweiterte Zusammenhänge wie Differenziale, das Schraubengelenk, eine vollständige Vorderradaufhängung eines Fahrzeugs und Feder-Dämpfer-Systeme im Allgemeinen. Motiviert durch konkrete Anwendungen wird weiterhin detailliert erläutert, wie sich auf Grundlage von Zwangsbedingungen die direkte Kontrolle eines Starrkörpers der Simulation in allen sechs Freiheitsgraden z.B. durch einen Datenhandschuh realisieren lässt. Bei den Lösungsverfahren für die Komplementaritätsformulierung werden die relativ einfache, angepasste Gauß-Seidel Routine sowie die Methode nach Dantzig umfassend besprochen und verglichen.

Für die universelle Mehrkörperdynamiksimulation hat sich die geschwindigkeitsbasierte Formulierung mit Lagrange-Multiplikatoren als flexibel und leistungsfähig erwiesen. Die Formulierung geschwindigkeitsbezogener Zwangsbedingungen mit Grenzwerten für die Zwangsimpulse erlaubt die Realisierung einer Vielzahl unterschiedlicher

dynamischer Zusammenhänge, die sich allesamt in einer einheitlichen Form fassen lassen. Auf Seiten der Lösungsroutinen für die geschwindigkeitsbasierte Formulierung stellt sich heraus, dass die angepasste Gauß-Seidel Routine zwar sehr einfach in der Realisierung und robust im Stabilitätsverhalten, für Modelle mit komplexen kinematischen Ketten jedoch nicht ausreichend genau ist. Hier bietet die angepasste Dantzig Routine einen deutlich besseren Kompromiss aus Laufzeitverhalten und Genauigkeit, wenngleich ihre Realisierung deutlich aufwendiger und fehleranfälliger ist. Lediglich für sehr „kontaktlastige“ Simulationen mit kleinen oder gar keinen gelenkgekoppelten Strukturen ist das iterative Gauß-Seidel Verfahren die bessere Wahl. Hier kann die Dantzig Routine häufig nicht terminieren. Außerdem bietet Gauß-Seidel die Möglichkeit, durch die Beschränkung der Iterationszahl abzuwägen zwischen gewünschter Genauigkeit der Lösung und der benötigten Rechenzeit.

Erst durch die geschickte Einbindung des Simulationsverfahrens in das VR-System VEROSIM® entsteht ein Dynamiksimulationssystem, mit dem schnell Prototypen und Demonstratoren modelliert und mit geringem programmiertechnischem Aufwand Anwendungen für den Produktiveinsatz realisiert werden können. Die Modellierung aller Elemente der Dynamiksimulation wie Körper, Gelenke, Motoren, Differenziale usw. folgt einem neuartigen, sehr intuitiven Konzept, das jedem Anwender schnell einen Zugang zur Modellierung von Simulationsmodellen eröffnet.

Der Kontaktgraph kann als Datengrundlage einer Reihe von Werkzeugen zur Optimierung Mehrkörperdynamiksimulation dienen. Der Einsatz von Stoßfortpflanzung ist zwar äußerst effektiv im Hinblick auf die Simulation großer Stapel, für die im Rahmen dieser Arbeit untersuchten Anwendungen spielt sie jedoch eine untergeordnete Rolle, da extreme Stapel nur geringe Anwendungsrelevanz besitzen. Sind Stapel Teil einer Anwendung, in denen gestapelte Körper z.B. gegriffen werden müssen, tritt zudem die Schwierigkeit auf, dass sich die Höhe im Stapel als Eigenschaft jedes Körpers nicht mehr sinnvoll bestimmen lässt. Aus diesem Grund wurde in den hier vorgestellten Anwendungen im Zweifel auf dieses Konzept verzichtet. Die anderen vorgestellten Anwendungen des Kontaktgraphen, die Bestimmung von Simulationsclustern, das Finden von Kollisionsgruppen und die dynamische Rekonfiguration kommen jedoch in allen Anwendungen zum Einsatz und haben sich als effektiv und praktisch erwiesen.

Das entwickelten Mehrkörperdynamiksimulationssystems als Grundlage der VR-Anwendungsentwicklung hat sich bewährt, dies belegt sein Einsatz in einem breiten Anwendungsfeld. Für schnelle Prototypen ohne weitere Ergänzungen an der Softwa-

re verspricht die vorgestellte Mehrkörperdynamiksimulation in kürzester Zeit realitätsnahe und beeindruckende Simulationsmodelle. Für Anwendungen im Produktiveinsatz sind auch auf Grundlage der Mehrkörperdynamiksimulation in aller Regel Ergänzungen in Form zusätzlicher Softwarekomponenten notwendig, die einfache Heuristiken umsetzen. Diese Ergänzungen fallen jedoch deutlich kleiner aus als jeweils individuelle Komplettlösungen auf Basis kinematischer, phänomenologischer, zustands- und steuerungsbasierter Simulationen.

Für hochexakte Simulationen zum Zwecke einer möglichst genauen Vorhersage realen Verhaltens ist das hier vorgestellte Verfahren - wie auch jedes andere Starrkörpermodell - allein nur bedingt geeignet. Die Annahmen des Starrkörpermodells sind dazu für viele Simulationsanwendungen zu weit entfernt von der Realität. Die Idee der Verfeinerung des Dynamiksimulationsmodells mit anwendungsspezifischen Simulationsmodellen je nach Anforderung scheint jedoch ein vielversprechender Ansatz zu sein, diese Schwäche zu überwinden. Mit Gluing und der in dieser Arbeit neu eingeführten Weiterentwicklung „Kooperatives T-T-Gluing“ steht ein Konzept zur Verknüpfung unterschiedlicher Dynamiksimulationsmodelle zur Verfügung, das die Integration unterschiedlicher Simulationsmodelle in die Mehrkörpersimulation erlaubt. Der kooperative Ansatz konnte zur Anbindung einer automatenbasierten Bodenmechaniksimulation bereits erfolgreich eingesetzt werden.

7.2. Ausblick

Mit Blick auf das eingesetzte Verfahren für die Starrkörperdynamiksimulation selbst sind anwendungsnahe Vergleiche mit den anderen, heute gebräuchlichen und universell einsetzbaren Verfahren von großem Interesse. Hier gilt es, als Vergleichskriterien nicht nur einzelne Aspekte wie die Rechenzeit oder die Genauigkeit zu beurteilen, sondern den Nutzwert der anderen Verfahren für ein möglichst breites Spektrum an Anwendungen zu untersuchen.

Um in Zukunft vermehrt als Grundlage für virtuelle Versuchsstände, sogenannte Virtuelle Testbeds, dienen zu können, erscheint vor allem die Integration mit anderen Simulationsmodellen von großer Bedeutung. Die Idee der Simulationsmodellverfeinerung sollte daher weiterentwickelt werden, um Starrkörpermodelle an den Stellen, an denen seine Annahmen in nicht hinnehmbaren Maße von der Realität abweichen, durch andere, realitätsnähere Teilmodelle zu ergänzen. Beispiele hierfür sind Modelle für die

Interaktion von Füßen oder Rädern mit Bodenmechanik, z.B. in Form eines Sandbodens, Modelle für Motordynamik und hydraulische Systeme und Simulationsmodelle für nicht starre Körper. Auch die Verbindung mit spezialisierten Starrkörpersimulationsmodellen in reduzierten Koordinaten für die hochgenaue Simulation einzelner dynamischer Strukturen, z.B. unter Ausnutzung von Integrationsschemata höherer Ordnung, ist eine mögliche Verfeinerung.

Neben der Verbindung unterschiedlicher dynamikbasierter Modelle ist auch die Frage interessant, wie sich hybride Simulationsmodelle aus kinematischen und dynamikbasierten Simulationsmodellen zusammensetzen lassen. Je nach Fokus einer Anwendung können Teile eines umfangreichen Modells durch rein kinematische Komponenten ersetzt werden, ohne dass die Realitätsnähe der Simulation darunter leidet. Hierdurch kann Rechenzeit eingespart und auch der Modellieraufwand reduziert werden, weil dynamische Eigenschaften wie Massen, Trägheiten, Kräfte und Momente nicht modelliert werden müssen. Beispiele hierfür sind z.B. Roboter in großen Fertigungsanlagen, deren dynamisches Verhalten nicht von Interesse ist oder das „Wackeln“ vieler kleiner Äste an einem Baum, den ein simulierter Harvester gerade aufarbeitet. Ein erster Ansatzpunkt hierzu kann das in Kapitel 6.4.2 vorgestellte Verfahren zur direkten Kontrolle eines Starrkörpers in der Dynamiksimulation z.B. durch den Anwender sein: Anstelle eines Anwenders kann eine beliebige kinematische Steuerung treten, um einen Starrkörper direkt zu kontrollieren.

Eine der größten Herausforderungen für die Simulation im Allgemeinen ist die Behandlung stets umfangreicher werdender Modelle. Die Mehrkörperdynamik und die Konzepte zu ihrer Verbindung mit anderen Simulationsverfahren können hier einen wertvollen Beitrag leisten. Sie schaffen die notwendigen Strukturen auf dem Weg zu einer „Weltsimulation“, die nicht auf einem einzelnen, die Realität perfekt nachbildenden Simulationsverfahren beruht, sondern auf der Verknüpfung unterschiedlicher, problem- und anwendungsorientierter Verfahren der Simulation.

Anhang A.

Mathematische und physikalische Grundlagen

A.1. Eigenschaften des Trägheitstensors

Die Massenverteilung im Starrkörper ist durch seinen Trägheitstensor definiert. Gute Herleitungen finden sich in vielen Lehrbüchern der Mechanik, zum Beispiel in [56] und [116]. An dieser Stelle sollen nur einige wichtige Eigenschaften des Trägheitstensors für die Anwendung in der Dynamiksimulation aufgezeigt werden.

Der Trägheitstensor hat die Form

$$\mathbf{\Theta}_A = \begin{pmatrix} \theta_x & \theta_{xy} & \theta_{xz} \\ \theta_{yx} & \theta_y & \theta_{yz} \\ \theta_{zx} & \theta_{zy} & \theta_z \end{pmatrix} \tag{A.1}$$

Die Elemente θ_x, θ_y und θ_z der Hauptdiagonalen sind die *Massenträgheitsmomente* bzgl. der x- y- und z-Achse (im Weltkoordinatensystem). Die Elemente auf den Nebendiagonalen sind die Deviations- oder Zentrifugalmomente. Sie hängen sowohl vom gewählten Bezugspunkt als auch von den gewählten Achsen des Körperkoordinatensystems ab. Wegen $\theta_{xy} = \theta_{yx}$ ist der Trägheitstensor symmetrisch zur Hauptdiagonalen. Die Eigenwerte des Trägheitstensors sind die Hauptträgheitsmomente des Starrkörpers und damit alle positiv.

Die Elemente des Trägheitstensors lassen sich mittels folgendem Integral über die

infinitesimalen Massenpunkte des Starrkörpers bestimmen [58]:

$$\mathbf{\Theta}_A = \int \begin{pmatrix} r_y^2 + r_z^2 & -r_x r_y & -r_x r_z \\ -r_y r_x & r_x^2 + r_z^2 & -r_y r_z \\ -r_z r_x & -r_z r_y & r_x^2 + r_y^2 \end{pmatrix} dm \tag{A.2}$$

Hierin sind r_x, r_y, r_z die Elemente des Vektors $\vec{r}$ vom Bezugspunkt A zum jeweiligen Massenpunkt. Zur Bestimmung der Elemente wird das Integral über die infinitesimalen Massenpunkte dm umgeformt in ein Volumenintegral. Für die durchschnittliche Dichte ρ des Körpers gilt:

$$\rho = \frac{dm}{dV} \tag{A.3}$$

In kartesischen Koordinaten ist $dV = dx\, dy\, dz$. Unter Annahme einer konstanten Dichte $\rho(x, y, z) = const.$ des Starrkörpers gilt damit für die Elemente des Trägheitstensors:

$$\mathbf{\Theta}_A = \rho \iiint_V \begin{pmatrix} r_y^2 + r_z^2 & -r_x r_y & -r_x r_z \\ -r_y r_x & r_x^2 + r_z^2 & -r_y r_z \\ -r_z r_x & -r_z r_y & r_x^2 + r_y^2 \end{pmatrix} dx\, dy\, dz \tag{A.4}$$

Für jeden Bezugspunkt A existiert ein Achsensystem mit drei zueinander orthogonalen Achsen x, y und z, für das alle Deviationsmomente Null sind. Man spricht von den *Symmetrieachsen* des Körpers. Für dieses *Hauptachsensystem* nimmt der Trägheitstensor die besonders einfache Form

$$\mathbf{\Theta}_A = \begin{pmatrix} \theta_x & 0 & 0 \\ 0 & \theta_y & 0 \\ 0 & 0 & \theta_z \end{pmatrix} \tag{A.5}$$

an. In dieser Form sind die Einträge der Hauptdiagonalen die *Hauptträgheitsmomente* des Körpers.

A.1.1. Koordinatentransformation des Trägheitstensors

Der Trägheitstensor ist abhängig vom Koordinatensystem, in dem er angegeben wird. Besitze ein Körper bezogen auf seinen Schwerpunkt und im körperfesten Koordinatensystem den zeitinvarianten Trägheitstensor ${}_K\Theta_S$. Die Rotation des Körpers relativ zum

raumfesten System sei $\mathbf{R}(t) \in \mathbb{R}^{3\times3}$. Der zeitveränderliche Trägheitstensor ${}_W\mathbf{\Theta}_S(t)$ im Weltkoordinatensystem ergibt sich aus der Vorschrift [58]:

$$ {}_W\mathbf{\Theta}_S(t) = \mathbf{R}(t) \cdot {}_K\mathbf{\Theta}_S \cdot \mathbf{R}^T(t) \tag{A.6} $$

Ebenso gilt für den inversen Trägheitstensor im Weltkoordinatensystem:

$$ \mathbf{\Theta}_S^{-1}(t) = (\mathbf{R}(t) \cdot {}_K\mathbf{\Theta}_S \cdot \mathbf{R}^T(t))^{-1} \tag{A.7} $$

$$ = \mathbf{R}^T(t)^{-1} \cdot (\mathbf{R}(\mathbf{t}) \cdot {}_K\mathbf{\Theta}_S)^{-1} \tag{A.8} $$

$$ = \mathbf{R}(t) \cdot {}_K\mathbf{\Theta}_S^{-1} \cdot \mathbf{R}^T(t) \tag{A.9} $$

A.1.2. Akkumulation von Trägheitstensoren

Wird ein Starrkörper aus mehreren einzelnen Starrkörpern zusammengesetzt, so können die Trägheitstensoren elementweise addiert werden. Dazu müssen sie jedoch im selben Koordinatensystem referenziert sein und somit ggf. entsprechend transformiert werden.

Die Rotation geschieht entsprechend Gleichung A.6. Für eine Verschiebung oder Translation wird die Steiner-Huygens-Relation [58] genutzt: Um den Trägheitstensor $\mathbf{\Theta}^A$ eines Körpers der Masse m gegeben relativ zum Referenzpunkt $\vec{A}$ relativ zum Referenzpunkt $\vec{B}$ anzugeben, ist die folgende Transformation anzuwenden:

$$ \mathbf{\Theta}^B = \underbrace{\begin{pmatrix} \theta_x & \theta_{xy} & \theta_{xz} \\ \theta_{yx} & \theta_y & \theta_{yz} \\ \theta_{zx} & \theta_{zy} & \theta_z \end{pmatrix}}_{\mathbf{\Theta}^A} + m \cdot \begin{pmatrix} t_y^2 + t_z^2 & -t_x t_y & -t_x t_z \\ -t_y t_x & t_x^2 + t_z^2 & -t_y t_z \\ -t_z t_x & -t_z t_y & t_x^2 + t_y^2 \end{pmatrix} \tag{A.10} $$

mit

$$ \vec{t} = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} = \vec{B} - \vec{A} \tag{A.11} $$

A.2. Das Coulombsche Reibungsmodell

Zur Bestimmung von Reibungskräften im Bereich der Starrkörperdynamik wird meist das Reibungsmodell von Coulomb angewandt. Demnach werden Kontaktsituationen in solche mit statischer und solche mit kinetischer Reibung unterteilt. Haben die jeweiligen Kontaktpunkte der in Kontakt stehenden Körper eine nicht verschwindende relative Geschwindigkeit entlang der Kontaktebene, so wirkt kinetische Reibung oder Gleitreibung $\vec{f}_{f_k}$. Für sie gilt, dass ihr Betrag proportional zur im Kontakt wirkenden Normalenkraft $\vec{f}_n$ ist und ihre Wirkrichtung genau der relativen Geschwindigkeit im Kontaktpunkt $\vec{v}_{rel}$ entgegengesetzt ist:

$$\vec{f}_{f_k} = \mu_k \cdot \left|\vec{f}_n\right| \cdot \frac{\vec{v}_{rel}}{|\vec{v}_{rel}|} \tag{A.12}$$

Der Faktor μ_k ist der Reibungskoeffizient der kinetischen Reibung. Es handelt sich um eine empirische Größe, die experimentell bestimmt werden kann und typischerweise zwischen Null und Eins liegt, jedoch auch über Eins liegen kann.

Verschwindet die relative Geschwindigkeit, tritt statische Reibung oder Haftreibung auf. Solange ihr Betrag unter einem Maximalwert liegt, der wiederum proportional zur Normalenkraft ist, ist sie genau so groß, dass die relative Geschwindigkeit auf Null gehalten wird. Mathematisch formuliert bedeutet dies für die statische Reibung:

$$\left|\vec{f}_{f_s}\right| \leq \mu_s \cdot \left|\vec{f}_n\right| \tag{A.13}$$

Analog zur kinetischen Reibung ist μ_s der Reibungskoeffizient für die statische Reibung, der typischerweise größer als jener der kinetischen Reibung ist. Der proportionale Zusammenhang zwischen Reibungs- und Normalenkraft definiert den sogenannten Reibungskonus, dargestellt in Abbildung A.1. Im Falle kinetischer Reibung verläuft der Reibungskraftvektor $\vec{f}_{f_k}$ genau vom Zentrum des Konus bis zu seinem Mantel. Im statischen Fall liegt der Reibungskraftvektor $\vec{f}_{f_s}$ innerhalb dieses Konus.

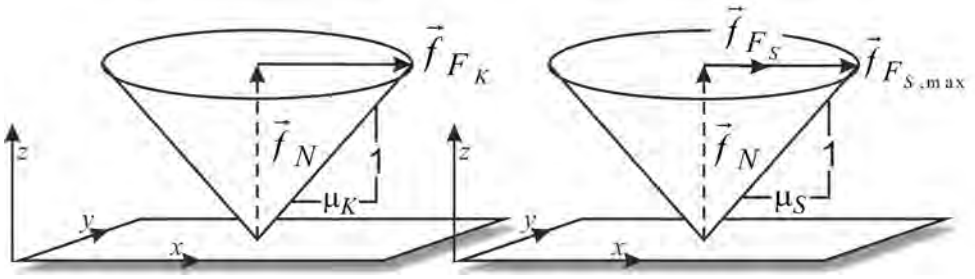


Abbildung A.1.: Das Coulombsche Reibungsmodell. Links: Kinetische Reibung, im Betrag genau proportional zum Betrag der Normalenkraft und der relativen Kontaktgeschwindigkeit entgegengerichtet. Rechts: Statische Reibung, sie liegt innerhalb des Reibungskonus und verhindert die relative Bewegung in der Kontaktebene vollständig.

A.3. Kreuzproduktmatrizen

Das Kreuzprodukt aus den Vektoren $\vec{a} \in \mathbb{R}^3, \vec{b} \in \mathbb{R}^3$:

$$\vec{a} \times \vec{b} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \times \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{pmatrix}$$

lässt sich auch durch folgenden Isomorphismus realisieren:

$$\vec{a} \times \vec{b} = \mathbf{a}^* \cdot \vec{b}$$

$$\text{mit } \mathbf{a}^* = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix} \in \mathbb{R}^{3\times 3}$$

A.4. LDLT-Zerlegung positiv semi-definiter Matrizen

Als $\mathbf{LDL}^T$-Zerlegung (Cholesky-Zerlegung) einer positiv definiten, symmetrischen Matrix bezeichnet man das Produkt:

$$\mathbf{A} = \mathbf{LDL}^T$$

Hierin ist $\mathbf{L}$ eine linke untere Dreiecksmatrix mit allen Diagonalelementen gleich Eins. $\mathbf{D}$ ist eine Diagonalmatrix.

Die Elemente der Matrizen $\mathbf{L}$ und $\mathbf{D}$ lassen sich mit folgender Routine [40] bestimmen:

Algorithmus 9: Algorithmus zur Bestimmung der $\mathbf{LDL}^T$ Zerlegung von $\mathbf{A}$.

```
for integer z = 0 to iAnzahlZeilen-1 do
    for integer s = 0 to z-1 do
        rSumme = 0 ;
        for integer l = 0 to s-1 do
            rSumme = rSumme + L[z][l] · D[l][l] · L[s][l] ;
        end
        L[z][s] = (A[z][s] − rSumme)/D[s][s] ;
    end
    L[z][z] = 1.0 ;
    D[z][z] = A[z][z] ;
    for integer l = 0 to z-1 do
        D[z][z] = D[z][z] − L[z][l] · L[z][l] · D[l][l];
    end
end
```

A.5. Das Quaternion als Orientierungsdarstellung

Sei eine Rotation gegeben durch den Drehvektor $\vec{r} \in \mathbb{R}^3, |\vec{r}| = 1$ und den Drehwinkel Φ. Hieraus lässt sich wie folgt das entsprechende Rotationsquaternion konstruieren:

$$\mathbf{q} = \left[\cos\frac{\Phi}{2}, \sin\frac{\Phi}{2} \cdot \vec{r}\right]^T = [a, \vec{v}]^T$$

Das Rotationsquaternion ist ein *Einheitsquaternion*, daher gilt:

$$|\mathbf{q}| = |a| + |\vec{v}| = 1$$

A.5.1. Quaternionenmultiplikation

Die Quaternionenmultiplikation von $\mathbf{p} = [a, \vec{v}]$ und $\mathbf{q} = [b, \vec{w}]$ ist definiert als:

$$\mathbf{p} \star \mathbf{q} = [ab - \vec{v} \cdot \vec{w}, a\vec{w} + b\vec{v} + \vec{v} \times \vec{w}]$$

A.5.2. Konjugiertes Quaternion

Sei $\mathbf{q} = [a, \vec{v}]$ ein Rotationsquaternion. Dann ist das konjugierte Quaternion $\overline{\mathbf{q}}$ definiert als:

$$\overline{\mathbf{q}} = [a, -\vec{v}]$$

A.5.3. Rotation eines Vektors

Um einen beliebigen Vektor $\vec{x} \in \mathbb{R}^3$ mittels einer Rotation gegeben durch das Quaternion $\mathbf{q} = [b, \vec{w}]$ zu rotieren, wird er als Quaternion $\mathbf{x} = [0, \vec{x}]^T$ interpretiert. Der rotierte Vektor $\vec{x}'$ ergibt sich dann aus folgender Quaternionenmultiplikation:

$$[0, \vec{x}'] = \mathbf{x}' = \mathbf{q} \star \mathbf{x} \star \overline{\mathbf{q}}$$

A.5.4. Inverse Rotation

Da Rotationsquaternione auch Einheitsquaternione sind, gilt für die inverse Rotation $\mathbf{q}^{-1}$ zu einer gegebenen Rotation $\mathbf{q}$:

$$\mathbf{q}^{-1} = \overline{\mathbf{q}}$$

A.5.5. Zusammenhang Quaternion und Winkelgeschwindigkeit

Sei $\mathbf{q}(t)$ die Orientierung eines Körpers und $\vec{\omega}$ seine Winkelgeschwindigkeit. Dann gilt für die Ableitung der Orientierung:

$$\dot{\mathbf{q}} = \frac{1}{2}(\omega \star \mathbf{q})$$

mit $\omega = [0, \vec{\omega}]^T$.

Ein expliziter Integrationsschritt mit Euler-Schema ergibt sich daraus zu:

$$\mathbf{q}(t+h) = \mathbf{q}(t) + \frac{1}{2}\left([0, h \cdot \vec{\omega}]^T \star \mathbf{q}\right)$$

Anhang B.

Ein analytisches Beispiel: Der SCARA-Roboter

Als ein etwas komplexeres, analytisches Beispiel für die Mehrkörperdynamiksimulation mit Lagrange-Multiplikatoren werden im Folgenden die Bewegungsgleichungen eines vereinfachten, 2D-SCARA-Roboters hergeleitet. Die Grundlagen des Ansatzes mit Lagrange-Multiplikatoren behandelt Kapitel 2.5.5 ab Seite 51. Die algebraischen Umformungen und Lösungen linearer Gleichungssysteme in diesem Kapitel wurden mit Hilfe der Software *Wolfram Mathematica*®[1] durchgeführt.

Abbildung B.1 zeigt das Modellbeispiel. Der erste Körper des Systems ist mit einem Drehgelenk an der festen Basis montiert, der zweite Körper über das zweite Drehgelenk am ersten. Dem Maximalkoordinatenansatz entsprechend werden beide Körper zunächst relativ zum Weltkoordinatensystem (WKS) betrachtet, jeweils in Form ihrer Schwerpunktposition $\vec{s}_i = (s_{i,x}\ s_{i,y})^T$ und ihrer Gesamtrotationen $\mathrm{Rot}(\varphi_1)$ und $\mathrm{Rot}(\varphi_3)$.

Als erstes werden die Bewegungs- bzw. Impulsbilanzgleichungen nach Newton und Euler (vgl. Kapitel 3.1.1 ab Seite 76) in Matrix-Vektor-Notation ohne Berücksichtigung

[1] Wolfram Mathematica ist eine kommerzielle Software für die Formelmanipulation. WWW: http://www.wolfram.com

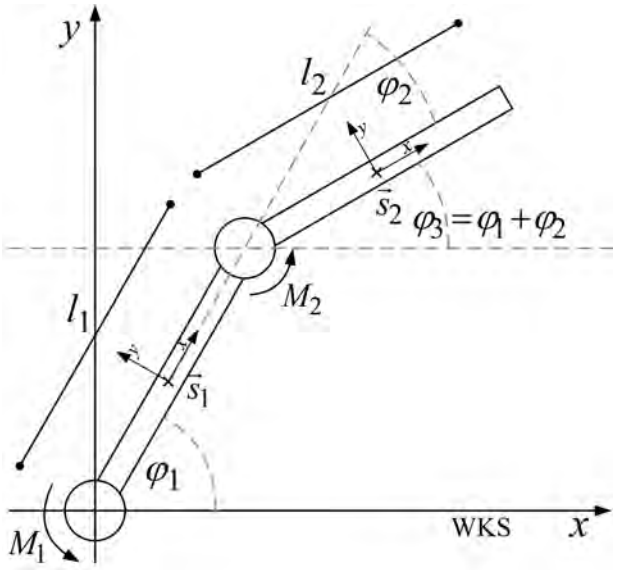


Abbildung B.1.: 2D-Modell eines Scara-Roboters

der Gelenke formuliert:

$$\underbrace{\begin{pmatrix} m_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & m_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \theta_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & m_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \theta_2 \end{pmatrix}}_{\mathbf{M}} \underbrace{\begin{pmatrix} \ddot{s}_{1,x} \\ \ddot{s}_{1,y} \\ \ddot{\varphi}_1 \\ \ddot{s}_{2,x} \\ \ddot{s}_{2,y} \\ \ddot{\varphi}_3 \end{pmatrix}}_{\ddot{\vec{x}}} = \underbrace{\begin{pmatrix} f_{1,x} \\ f_{1,y} \\ M_1 - M_2 \\ f_{2,x} \\ f_{2,y} \\ M_2 \end{pmatrix}}_{\vec{f}_{\text{ext}}} \tag{B.1}$$

Hierin sind $f_{i,x}, f_{i,y}$ die im Schwerpunkt von Körper i angreifenden Kräfte in x- bzw. y-Richtung, M_i die angreifenden Motormomente, m_i die Masse von Körper i und θ_i sein Trägheitsmoment bei Rotation um den Schwerpunkt. Die beiden Körper werden im Folgenden als idealisierte Stäbe angenommen. Hieraus ergibt sich für das Trägheitsmoment θ_i von Körper i mit der Länge l_i:

$$\theta_i = \frac{1}{2} l_i^2 m_i \tag{B.2}$$

Die Gelenke werden in Form von Zwangsbedingungen berücksichtigt, die ihrerseits Zwangskräfte der Form $\mathbf{J}^T \vec{\lambda}$ (vgl. Kapitel 2.5.1 ab Seite 39) implizieren. Das System impliziert zwei Paare von Zwangsbedingungen $\vec{c}_1(\vec{x,t}) = \vec{0}, \vec{c}_2(\vec{x,t}) = \vec{0}$, weil jedes Gelenk

je zwei translatorische Freiheitsgrade einschränkt. Das erste Gelenk erzwingt, dass das untere Ankerpunkt von Körper 1 im Ursprung des Weltkoordinatensystems verbleibt: Die Gelenke werden in Form von Zwangsbedingungen berücksichtigt, die ihrerseits Zwangskräfte der Form $\mathbf{J}^T\vec{\lambda}$ (vgl. Kapitel 2.5.1 ab Seite 39) implizieren. Das System impliziert zwei Paare von Zwangsbedingungen $\vec{c}_1(\vec{x,t}) = \vec{0}, \vec{c}_2(\vec{x,t}) = \vec{0}$, weil jedes Gelenk je zwei translatorische Freiheitsgrade einschränkt. Das erste Gelenk erzwingt, dass das untere Ankerpunkt von Körper 1 im Ursprung des Weltkoordinatensystems verbleibt:

$$\begin{aligned}
\vec{c}_1(\vec{x},t) =& \vec{s}_1(t) + \mathrm{Rot}\left(\varphi_1(t)\right)\begin{pmatrix}\frac{-l_1}{2}\\0\end{pmatrix} &&= \begin{pmatrix}0\\0\end{pmatrix}\\
&\vec{s}_1 + \begin{pmatrix}\cos(\varphi_1) & -\sin(\varphi_1)\\ \sin(\varphi_1) & \cos(\varphi_1)\end{pmatrix}\begin{pmatrix}\frac{-l_1}{2}\\0\end{pmatrix} &&= \begin{pmatrix}0\\0\end{pmatrix}\\
&\begin{pmatrix} s_{1,x} - \frac{1}{2}\cos\left(\varphi_1\right) l_1 \\ s_{1,y} - \frac{1}{2}\sin\left(\varphi_1\right) l_1 \end{pmatrix} &&= \begin{pmatrix}0\\0\end{pmatrix}
\end{aligned} \tag{B.3}$$

Zur Gewinnung der Form $\mathbf{J}\ddot{\vec{x}}$ wird $\vec{c}_1$ zweifach nach der Zeit abgeleitet:

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{c}_1 = \begin{pmatrix}\dot{s}_{1,x} + \frac{1}{2}l_1\dot{\varphi}_1\sin\left(\varphi_1\right)\\ \dot{s}_{1,y} - \frac{1}{2}l_1\dot{\varphi}_1\cos\left(\varphi_1\right)\end{pmatrix} = \begin{pmatrix}0\\0\end{pmatrix} \tag{B.4}$$

$$\begin{aligned}
\frac{\mathrm{d}^2}{\mathrm{d}t^2}\vec{c}_1 = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\mathrm{d}}{\mathrm{d}t}\vec{c}_1\right) &= \begin{pmatrix}0\\0\end{pmatrix}\\
\begin{pmatrix} \frac{1}{2}\cos\left(\varphi_1\right) l_1 {\dot{\varphi}_1}^2 + \ddot{s}_{1,x} + \frac{1}{2}\sin\left(\varphi_1\right) l_1\ddot{\varphi}_1 \\ \frac{1}{2}\sin\left(\varphi_1\right) l_1 {\dot{\varphi}_1}^2 + \ddot{s}_{1,y} - \frac{1}{2}\cos\left(\varphi_1\right) l_1\ddot{\varphi}_1 \end{pmatrix} &= \begin{pmatrix}0\\0\end{pmatrix}
\end{aligned} \tag{B.5}$$

Das zweite Gelenk erzwingt, dass der obere Ankerpunkt von Körper 1 gleich dem unteren Ankerpunkt von Körper 2 ist. Formal ergibt sich daher für $\vec{c}_2$:

$$\begin{aligned}
\vec{c}_2(\vec{x},t) = \vec{s}_2(t) + \mathrm{Rot}\left(\varphi_3(t)\right)\begin{pmatrix}-\frac{l_2}{2}\\0\end{pmatrix} - \vec{s}_1(t) - \mathrm{Rot}\left(\varphi_1(t)\right)\begin{pmatrix}\frac{l_1}{2}\\0\end{pmatrix} &= \begin{pmatrix}0\\0\end{pmatrix}\\
\begin{pmatrix} -s_{1,x} + s_{2,x} - \frac{1}{2}\cos\left(\varphi_1\right) l_1 - \frac{1}{2}\cos\left(\varphi_3\right) l_2 \\ -s_{1,y} + s_{2,y} - \frac{1}{2}\sin\left(\varphi_1\right) l_1 - \frac{1}{2}\sin\left(\varphi_3\right) l_2 \end{pmatrix} &= \begin{pmatrix}0\\0\end{pmatrix}
\end{aligned} \tag{B.6}$$

Zweifaches Ableiten nach t führt auf:

$$\begin{aligned}
\frac{\mathrm{d}^2}{\mathrm{d}t^2}\vec{c}_2 = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\mathrm{d}}{\mathrm{d}t}\vec{c}_2\right) &= \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\
\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} -\dot{s}_{1,x}(t) + \dot{s}_{2,x}(t) + \frac{1}{2}\sin(\varphi_1(t))\, l_1\dot{\varphi_1}(t) + \frac{1}{2}\sin(\varphi_3(t))\, l_2\dot{\varphi_3}(t) \\ -\dot{s}_{1,y}(t) + \dot{s}_{2,y}(t) - \frac{1}{2}\cos(\varphi_1(t))\, l_1\dot{\varphi_1}(t) - \frac{1}{2}\cos(\varphi_3(t))\, l_2\dot{\varphi_3}(t) \end{pmatrix} &= \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\
\begin{pmatrix} \frac{1}{2}\cos(\varphi_1)\, l_1{\dot{\varphi_1}}^2 + \frac{1}{2}\cos(\varphi_3)\, l_2{\dot{\varphi_3}}^2 - \ddot{s}_{1,x} + \ddot{s}_{2,x} + \frac{1}{2}\sin(\varphi_1)\, l_1\ddot{\varphi_1} + \frac{1}{2}\sin(\varphi_3)\, l_2\ddot{\varphi_3} \\ \frac{1}{2}\sin(\varphi_1)\, l_1{\dot{\varphi_1}}^2 + \frac{1}{2}\sin(\varphi_3)\, l_2{\dot{\varphi_3}}^2 - \ddot{s}_{1,y} + \ddot{s}_{2,y} - \frac{1}{2}\cos(\varphi_1)\, l_1\ddot{\varphi_1} - \frac{1}{2}\cos(\varphi_3)\, l_2\ddot{\varphi_3} \end{pmatrix} &= \begin{pmatrix} 0 \\ 0 \end{pmatrix}
\end{aligned} \tag{B.7}$$

Jetzt kann die Jacobimatrix der Zwangsbedingungen $\mathbf{J}$ aufgestellt werden. Mit

$$\ddot{\vec{x}} = \begin{pmatrix} \ddot{s}_{1,x} & \ddot{s}_{1,y} & \ddot{\varphi_1} & \ddot{s}_{2,x} & \ddot{s}_{2,y} & \ddot{\varphi_3} \end{pmatrix}^T$$

und der Zielform $\mathbf{J}\ddot{\vec{x}} = \vec{b}$ ergeben sich die Einträge der Jacobimatrix aus den Koeffizienten der zweifach abgeleiteten Zwangsbedingungen des SCARA-Roboter-Beispiels zu:

$$\mathbf{J}_{\mathrm{SCARA}} = \begin{pmatrix} 1 & 0 & \frac{1}{2}\sin(\varphi_1)\, l_1 & 0 & 0 & 0 \\ 0 & 1 & -\frac{1}{2}\cos(\varphi_1)\, l_1 & 0 & 0 & 0 \\ -1 & 0 & \frac{1}{2}\sin(\varphi_1)\, l_1 & 1 & 0 & \frac{1}{2}\sin(\varphi_3)\, l_2 \\ 0 & -1 & -\frac{1}{2}\cos(\varphi_1)\, l_1 & 0 & 1 & -\frac{1}{2}\cos(\varphi_3)\, l_2 \end{pmatrix} \tag{B.8}$$

Für die rechte Seite der Zwangsbedingungen, den Vektor $\vec{b}$, bleibt entsprechend (Vorzeichenwechsel beachten!):

$$\vec{b}_{\mathrm{SCARA}} = \begin{pmatrix} -\frac{1}{2}\cos(\varphi_1)\, l_1{\dot{\varphi_1}}^2 \\ -\frac{1}{2}\sin(\varphi_1)\, l_1{\dot{\varphi_1}}^2 \\ -\frac{1}{2}\cos(\varphi_1)\, l_1{\dot{\varphi_1}}^2 - \frac{1}{2}\cos(\varphi_3)\, l_2{\dot{\varphi_3}}^2 \\ -\frac{1}{2}\sin(\varphi_1)\, l_1{\dot{\varphi_1}}^2 - \frac{1}{2}\sin(\varphi_3)\, l_2{\dot{\varphi_3}}^2 \end{pmatrix} \tag{B.9}$$

Nun müssen Zwangsbedingungen und Bewegungsgleichungen zusammengeführt werden. Entsprechend dem „großen“ Lagrange-Multiplikatoren Ansatz (vgl. 2.5.5, Seite 52), wird das System wie folgt zusammengesetzt:

$$\begin{pmatrix} \mathbf{M} & -\mathbf{J}^T \\ \mathbf{J} & \mathbf{0} \end{pmatrix} \cdot \begin{pmatrix} \ddot{\vec{x}} \\ \vec{\lambda} \end{pmatrix} = \begin{pmatrix} \vec{f}_{\mathrm{ext}} \\ \vec{b} \end{pmatrix} \tag{B.10}$$

Das resultierende System ist in Gleichung B.11 abgebildet.

Lösung des linearen Gleichungssystems B.11 liefert die gesuchten Berechnungsvorschriften für alle Beschleunigungen und Winkelbeschleunigungen formuliert im Weltkoordinatensystem sowie die Lagrange-Multiplikatoren und damit die vorzeichenbehafteten Beträge der Zwangskräfte im System. Die externen Kräfte $\vec{f_1}, \vec{f_2}$ wurden als $\vec{0}$ angenommen. Gleichung B.12 zeigt die Lösungen für die Körperbeschleunigungen, auf die Darstellung der Lösung für die Lagrange-Multiplikatoren wurde aus Platzgründen verzichtet.

Das lineare Gleichungssystem aus Bewegungsgleichungen und Zwangsbedingungen des SCARA-Modellroboters in der „großen“ Lagrange-Multiplikatoren-Formulierung. Aus Platzgründen wurden in den Gleichungen dieser Seite $\sin(x)$ durch $s(x)$ und $\cos(x)$ durch $c(x)$ ersetzt.

$$\begin{pmatrix}
m_1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\
0 & m_1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \\
0 & 0 & \frac{1}{12} l_1^2 m_1 & 0 & 0 & 0 & -\frac{1}{2} s\left(\varphi_1\right) l_1 & \frac{1}{2} c\left(\varphi_1\right) l_1 & -\frac{1}{2} s\left(\varphi_1\right) l_1 & \frac{1}{2} c\left(\varphi_1\right) l_1 \\
0 & 0 & 0 & m_2 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & m_2 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{12} l_2^2 m_2 & 0 & 0 & -\frac{1}{2} s\left(\varphi_3\right) l_2 & \frac{1}{2} c\left(\varphi_3\right) l_2 \\
1 & 0 & \frac{1}{2} s\left(\varphi_1\right) l_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -\frac{1}{2} c\left(\varphi_1\right) l_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & \frac{1}{2} s\left(\varphi_1\right) l_1 & 1 & 0 & \frac{1}{2} s\left(\varphi_3\right) l_2 & 0 & 0 & 0 & 0 \\
0 & -1 & -\frac{1}{2} c\left(\varphi_1\right) l_1 & 0 & 1 & -\frac{1}{2} c\left(\varphi_3\right) l_2 & 0 & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
\ddot{s}_{1,x} \\ \ddot{s}_{1,y} \\ \ddot{\varphi}_1 \\ \ddot{s}_{2,x} \\ \ddot{s}_{2,y} \\ \ddot{\varphi}_3 \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4
\end{pmatrix}
=
\begin{pmatrix}
f_{1,x} \\ f_{1,y} \\ M_1 - M_2 \\ f_{2,x} \\ f_{2,y} \\ M_2 \\ -\frac{1}{2} c\left(\varphi_1\right) l_1 \dot{\varphi_1}^2 \\ -\frac{1}{2} s\left(\varphi_1\right) l_1 \dot{\varphi_1}^2 \\ -\frac{1}{2} c\left(\varphi_1\right) l_1 \dot{\varphi_1}^2 - \frac{1}{2} c\left(\varphi_3\right) l_2 \dot{\varphi_3}^2 \\ -\frac{1}{2} s\left(\varphi_1\right) l_1 \dot{\varphi_1}^2 - \frac{1}{2} s\left(\varphi_3\right) l_2 \dot{\varphi_3}^2
\end{pmatrix} \tag{B.11}$$

Mit $f_{1,x} = f_{1,y} = f_{2,x} = f_{2,y} = 0$ ergibt sich für die Beschleunigungsterme:

$$\begin{pmatrix} \ddot{s}_{1,x} \\ \ddot{s}_{1,y} \\ \ddot{\varphi}_1 \\ \ddot{s}_{2,x} \\ \ddot{s}_{2,y} \\ \ddot{\varphi}_3 \end{pmatrix} = \begin{pmatrix}
\frac{-l_1^2 l_2 (8c(\varphi_1) m_1 + 3(5c(\varphi_1) - 3c(\varphi_1 - 2\varphi_3)) m_2) \dot{\varphi_1}^2 + 24(M_2 - M_1) s(\varphi_1) l_2 + 12 s(\varphi_1) l_1 \left(s(\varphi_1 - \varphi_3) l_2^2 m_2 \dot{\varphi_3}^2 + 3 M_2 c(\varphi_1 - \varphi_3)\right)}{2 l_1 l_2 (8 m_1 + 3(5 - 3c(2(\varphi_1 - \varphi_3))) m_2)} \\
-\frac{l_1^2 l_2 (8 s(\varphi_1) m_1 + 3(5 s(\varphi_1) + 3 s(\varphi_1 - 2\varphi_3)) m_2) \dot{\varphi_1}^2 + 24(M_2 - M_1) c(\varphi_1) l_2 + 12 c(\varphi_1) l_1 \left(s(\varphi_1 - \varphi_3) l_2^2 m_2 \dot{\varphi_3}^2 + 3 M_2 c(\varphi_1 - \varphi_3)\right)}{2 l_1 l_2 (8 m_1 + 3(5 - 3c(2(\varphi_1 - \varphi_3))) m_2)} \\
-\frac{3\left(3 s(2(\varphi_1 - \varphi_3)) l_1^2 l_2 m_2 \dot{\varphi_1}^2 + 8(M_2 - M_1) l_2 + 4 l_1 \left(s(\varphi_1 - \varphi_3) l_2^2 m_2 \dot{\varphi_3}^2 + 3 M_2 c(\varphi_1 - \varphi_3)\right)\right)}{l_1^2 l_2 (8 m_1 + 3(5 - 3c(2(\varphi_1 - \varphi_3))) m_2)} \\
\frac{l_1^2 l_2 m_2 ((5c(\varphi_1) + 3c(\varphi_1 - 2\varphi_3)) m_1 + 6c(\varphi_1) m_2) \dot{\varphi_1}^2 + 3(M_1 - M_2)(5 s(\varphi_1) + 3 s(\varphi_1 - 2\varphi_3)) l_2 m_2 + l_1 \left(3c(\varphi_1) m_2 \left(c(\varphi_1 - \varphi_3) l_2^2 m_2 \dot{\varphi_3}^2 - 12 M_2 s(\varphi_1 - \varphi_3)\right) + 4 m_1 \left(c(\varphi_3) l_2^2 m_2 \dot{\varphi_3}^2 + 3 M_2 s(\varphi_3)\right)\right)}{l_1 l_2 m_2 (3(3c(2(\varphi_1 - \varphi_3)) - 5) m_2 - 8 m_1)} \\
\frac{l_1^2 l_2 m_2 ((5 s(\varphi_1) - 3 s(\varphi_1 - 2\varphi_3)) m_1 + 6 s(\varphi_1) m_2) \dot{\varphi_1}^2 + 3(M_1 - M_2)(3c(\varphi_1 - 2\varphi_3) - 5c(\varphi_1)) l_2 m_2 + l_1 \left(3 s(\varphi_1) m_2 \left(c(\varphi_1 - \varphi_3) l_2^2 m_2 \dot{\varphi_3}^2 - 12 M_2 s(\varphi_1 - \varphi_3)\right) + m_1 \left(4 s(\varphi_3) l_2^2 m_2 \dot{\varphi_3}^2 - 12 M_2 c(\varphi_3)\right)\right)}{l_1 l_2 m_2 (3(3c(2(\varphi_1 - \varphi_3)) - 5) m_2 - 8 m_1)} \\
-\frac{3\left(4 s(\varphi_1 - \varphi_3) l_1^2 l_2 m_2 (m_1 + 3 m_2) \dot{\varphi_1}^2 + 12(M_2 - M_1) c(\varphi_1 - \varphi_3) l_2 m_2 + l_1 \left(8 M_2 m_1 + 3 m_2 \left(s(2(\varphi_1 - \varphi_3)) l_2^2 m_2 \dot{\varphi_3'}^2 + 8 M_2\right)\right)\right)}{l_1 l_2^2 m_2 (3(3c(2(\varphi_1 - \varphi_3)) - 5) m_2 - 8 m_1)}
\end{pmatrix} \tag{B.12}$$

Auf die Darstellung der Terme für die Lagrange-Multiplikatoren wurde aus Platzgründen verzichtet.

In der Robotik werden Bewegungsgleichungen in folgender Form dargestellt:

$$\vec{M} = \theta\ddot{\vec{\varphi}} + COR\vec{Z}EN(\vec{\varphi}, \dot{\vec{\varphi}}) + GR\vec{A}V(\vec{\varphi}) \tag{B.13}$$

Hierin sind $\vec{M}$ der Vektor der eingeprägten Kräfte und Momente in generalisierten bzw. Minimalkoordinaten, $\vec{\varphi}$ die generalisierten Koordinaten des Systems, $COR\vec{Z}EN(\vec{\varphi}, \dot{\vec{\varphi}})$ der Vektor der Coriolis- und Zentripetalkräfte und $GR\vec{A}V(\vec{\varphi})$ der Vektor der Gravitationskräfte.

Die eingeprägten Momente am SCARA-Beispiel-Roboter sind die Gelenkmomente M_1, M_2. Die zugehörigen, generalisierten Roboterkoordinaten sind die Gelenkwinkel φ_1 und φ_2. Aufgrund des Maximalkoordinatenansatzes wurde der zweite Körper bis hierher mit seiner Gesamtrotation relativ zum Weltkoordinatensystem φ_3 betrachtet. In diesem einfachen 2D-Modell gelten die Zusammenhänge:

$$\begin{aligned} \varphi_3 &= \varphi_1 + \varphi_2 \\ \dot{\varphi}_3 &= \dot{\varphi}_1 + \dot{\varphi}_2 \\ \ddot{\varphi}_3 &= \ddot{\varphi}_1 + \ddot{\varphi}_2 \end{aligned} \tag{B.14}$$

Unter der Annahme, dass seine Drehachsen parallel zur Gravitationsrichtung montiert sind, sind die Beträge der externen Kräfte in Gleichung B.11 gleich Null. Trennt man aus der Gesamtlösung B.12 die Vorschriften für $\ddot{\varphi}_1$ und $\ddot{\varphi}_3$ heraus, ersetzt φ_3 und alle Ableitungen hiervon wie angegeben und löst das System nach den eingeprägten Momenten M_1, M_2, so erhält man die Gleichungen:

$$\begin{aligned} M_1 &= \frac{1}{6}\left(3l_2l_1m_2\left(\left(2\ddot{\varphi}_1 + \ddot{\varphi}_2\right)\cos\left(\varphi_2\right) - \dot{\varphi}_2\left(2\dot{\varphi}_1 + \dot{\varphi}_2\right)\sin\left(\varphi_2\right)\right) + 2l_1^2\left(m_1 + 3m_2\right)\ddot{\varphi}_1 + 2l_2^2m_2\left(\ddot{\varphi}_1 + \ddot{\varphi}_2\right)\right) \\ M_2 &= \frac{1}{6}l_2m_2\left(3l_1\left({\dot{\varphi}_1}^2\sin\left(\varphi_2\right) + \ddot{\varphi}_1\cos\left(\varphi_2\right)\right) + 2l_2\left(\ddot{\varphi}_1 + \ddot{\varphi}_2\right)\right) \end{aligned} \tag{B.15}$$

Sortieren nach den jeweiligen Koeffizienten führt schließlich auf die gesuchte Form:

$$\begin{pmatrix} M_1 \\ M_2 \end{pmatrix} = \begin{pmatrix} \frac{1}{3}\left(3l_2l_1m_2\cos\left(\varphi_2\right) + l_1^2\left(m_1+3m_2\right) + l_2^2m_2\right) & \frac{1}{6}l_2m_2\left(3l_1\cos\left(\varphi_2\right)+2l_2\right) \\ \frac{1}{6}l_2m_2\left(3l_1\cos\left(\varphi_2\right)+2l_2\right) & \frac{1}{3}l_2^2m_2 \end{pmatrix} \begin{pmatrix} \ddot{\varphi}_1 \\ \ddot{\varphi}_2 \end{pmatrix} + \begin{pmatrix} -\dot{\varphi}_1\dot{\varphi}_2l_1l_2m_2\sin(\varphi_2) - \frac{1}{2}\dot{\varphi}_2^2l_1l_2m_2\sin(\varphi_2) \\ -\frac{1}{2}\dot{\varphi}_1^2l_1l_2m_2\sin(\varphi_2) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{B.16}$$

Anhang C.

Jacobimatrix der Zwangsbedingungen und in der Robotik

Im Kontext der Robotik wird der Begriff *Jacobimatrix* vor allem in folgendem Zusammenhang genutzt. Sei $\vec{q}^{\,\mathrm{R}} \in \mathbb{R}^m$ der Vektor der m unabhängigen (generalisierten) Koordinaten eines Manipulators. Am Beispiel eines Roboters mit sechs Drehgelenken entsprechen sie seinen sechs Gelenkwinkeln. Sei weiterhin $\vec{x}^{\,\mathrm{TCP}} \in \mathbb{R}^6$ die Lage (Position und Orientierung) des Werkzeugträgers (engl.: *Tool Center Point, TCP*) des Roboters. Dann gilt der Zusammenhang:

$$\dot{\vec{x}}^{\mathrm{TCP}} = \mathbf{J}^{\mathrm{TCP,R}} \cdot \dot{\vec{q}}^{\,\mathrm{R}} = \begin{pmatrix} \frac{\partial x_1^{\mathrm{TCP}}}{\partial q_1^{\mathrm{R}}} & \cdots & \frac{\partial x_1^{\mathrm{TCP}}}{\partial q_m^{\mathrm{R}}} \\ \vdots & & \vdots \\ \frac{\partial x_6^{\mathrm{TCP}}}{\partial q_1^{\mathrm{R}}} & \cdots & \frac{\partial x_6^{\mathrm{TCP}}}{\partial q_m^{\mathrm{R}}} \end{pmatrix} \cdot \begin{pmatrix} \dot{q}_1^{\mathrm{R}} \\ \vdots \\ \dot{q}_m^{\mathrm{R}} \end{pmatrix} \tag{C.1}$$

Hier ist $\mathbf{J}^{\mathrm{TCP,R}}$ die Jacobimatrix des Roboters. Sie bildet die Geschwindigkeiten der Gelenkachsen in ihren generalisierten (Roboter-) Koordinaten auf die kartesischen Geschwindigkeiten des TCP ab. Man erhält sie durch Ableitung der Koordinatentransformationen zwischen Roboter- $\vec{q}^{\,\mathrm{R}}$ und kartesischen Koordinaten des TCP $\vec{x}^{\,\mathrm{TCP}}$.

Um eine Verbindung zu der in dieser Arbeit genutzten Jacobimatrix der Zwangsbedingungen herzustellen, kann man folgendes Gedankenexperiment anstellen. Oben beschriebene Abbildung wird in zweierlei Hinsicht erweitert. Erstens müssen neben den kartesischen Koordinaten des TCP auch die kartesischen Koordinaten aller $n-1$ *inneren Glieder* des Roboters $\vec{x}^{G_1} \ldots \vec{x}^{G_{n-1}}, \vec{x}^{G_i} \in \mathbb{R}^6$ berücksichtigt werden. Die entsprechend erweiterte Jacobimatrix erhält man ebenfalls durch Ableiten der Koordinatentransformationen zwischen den Roboterkoordinaten und den kartesischen Koordinaten

aller Glieder.

$$\begin{pmatrix} \vec{x}^{\,\mathrm{TCP}} \\ \vec{x}^{\,G_1} \\ \vdots \\ \vec{x}^{\,G_{n-1}} \end{pmatrix} = \left(\begin{array}{ccc} \frac{\partial x_1^{\mathrm{TCP}}}{\partial q_1^{\mathrm{R}}} & \cdots & \frac{\partial x_1^{\mathrm{TCP}}}{\partial q_m^{\mathrm{R}}} \\ \vdots & & \vdots \\ \frac{\partial x_6^{\mathrm{TCP}}}{\partial q_1^{\mathrm{R}}} & \cdots & \frac{\partial x_6^{\mathrm{TCP}}}{\partial q_m^{\mathrm{R}}} \\ \hdashline \frac{\partial x_1^{G_1}}{\partial q_1^{\mathrm{R}}} & \cdots & \frac{\partial x_1^{G_1}}{\partial q_m^{\mathrm{R}}} \\ \vdots & & \vdots \\ \frac{\partial x_6^{G_{n-1}}}{\partial q_1^{\mathrm{R}}} & \cdots & \frac{\partial x_6^{G_{n-1}}}{\partial q_m^{\mathrm{R}}} \end{array}\right) \cdot \begin{pmatrix} \dot{q}_1^{\mathrm{R}} \\ \vdots \\ \dot{q}_m^{\mathrm{R}} \end{pmatrix} \tag{C.2}$$

Zweitens ergänzt man den Vektor der unabhängigen Koordinaten $\vec{q}$ um die die $6n-m$ entfernten Freiheitsgrade repräsentierenden Koordinaten $\vec{q}_i^{\,\mathrm{Z}}$. Am Beispiel eines Drehegelenks bedeutet das: Zu der Koordinate, die den Gelenkwinkel repräsentiert, kommen 5 weitere (2 rotatorische, 3 translatorische) Koordinaten und damit Freiheitsgrade hinzu, die *eigentlich* durch Zwangsbedingungen entfernt sind. Diese entfernten Freiheitsgrade ordnen wir im Vektor $\vec{q}$ hinter den Gelenkwinkeln des Roboters an:

$$\begin{pmatrix} \dot{\vec{x}}^{\,\mathrm{TCP}} \\ \dot{\vec{x}}^{\,G_1} \\ \vdots \\ \dot{\vec{x}}^{\,G_{n-1}} \end{pmatrix} = \underbrace{\left(\begin{array}{ccc:ccc} \frac{\partial x_1^{\mathrm{TCP}}}{\partial q_1^{\mathrm{R}}} & \cdots & \frac{\partial x_1^{\mathrm{TCP}}}{\partial q_m^{\mathrm{R}}} & \frac{\partial x_1^{\mathrm{TCP}}}{\partial q_1^{\mathrm{Z}}} & \cdots & \frac{\partial x_1^{\mathrm{TCP}}}{\partial q_{n-m}^{\mathrm{Z}}} \\ \vdots & & \vdots & \vdots & & \vdots \\ \frac{\partial x_6^{\mathrm{TCP}}}{\partial q_1^{\mathrm{R}}} & \cdots & \frac{\partial x_6^{\mathrm{TCP}}}{\partial q_m^{\mathrm{R}}} & \frac{\partial x_6^{\mathrm{TCP}}}{\partial q_1^{\mathrm{Z}}} & \cdots & \frac{\partial x_6^{\mathrm{TCP}}}{\partial q_{n-m}^{\mathrm{Z}}} \\ \hdashline \frac{\partial x_1^{G_1}}{\partial q_1^{\mathrm{R}}} & \cdots & \frac{\partial x_1^{G_1}}{\partial q_m^{\mathrm{R}}} & \frac{\partial x_1^{G_1}}{\partial q_1^{\mathrm{Z}}} & \cdots & \frac{\partial x_1^{G_1}}{\partial q_{6n-m}^{\mathrm{Z}}} \\ \vdots & & \vdots & \vdots & & \vdots \\ \frac{\partial x_6^{G_{n-1}}}{\partial q_1^{\mathrm{R}}} & \cdots & \frac{\partial x_6^{G_{n-1}}}{\partial q_m^{\mathrm{R}}} & \frac{\partial x_6^{G_{n-1}}}{\partial q_1^{\mathrm{Z}}} & \cdots & \frac{\partial x_6^{G_{n-1}}}{\partial q_{6n-m}^{\mathrm{Z}}} \end{array}\right)}_{\mathbf{J}_{\mathrm{Rob}}} \cdot \underbrace{\begin{pmatrix} \dot{q}_1^{\mathrm{R}} \\ \vdots \\ \dot{q}_m^{\mathrm{R}} \\ \hdashline \dot{q}_1^{\mathrm{Z}} \\ \vdots \\ \dot{q}_{6n-m}^{\mathrm{Z}} \end{pmatrix}}_{\dot{\vec{q}}} \tag{C.3}$$

Beim Roboter sind die Elemente $\dot{q}_1^{\mathrm{Z}} \ldots \dot{q}_{6n-m}^{\mathrm{Z}}$ alle gleich Null, weil dies gerade die entfernten Freiheitsgrade sind, entlang derer keine Bewegung stattfindet. Der gesamte rechte Teil der dargestellten Jacobimatrix $\mathbf{J}_{\mathrm{Rob}}$ hat demnach - entsprechend den Erwartungen - keinen Einfluss auf die Bewegungen der Roboterglieder und seines TCP.

Gruppieren wir nun diese Darstellung wie folgt:

$$\mathbf{J}_{\mathrm{Rob}} = \left(\begin{array}{c|c} \mathbf{J}^{\mathrm{TCP,R}} & \mathbf{J}^{\mathrm{TCP,Z}} \\ \hline \mathbf{J}^{\mathrm{G,R}} & \mathbf{J}^{\mathrm{G,Z}} \end{array}\right) = \left(\begin{array}{c|c} \mathbf{J}^{\mathrm{R}} & \mathbf{J}^{\mathrm{Z}} \end{array}\right) \tag{C.4}$$

Hierin repräsentiert $\mathbf{J}^{\mathrm{Z}}$ den Einfluss der entfernten Freiheitsgrade auf die Bewegungen der Glieder und den TCP des Roboters in kartesischen Koordinaten.

Nimmt man an, die Matrix $\mathbf{J}_{\mathrm{Rob}}$ ist invertierbar, so lässt sich Gleichung C.3 lösen nach den $\dot{\vec{q}}$:

$$\left(\begin{array}{c|c} \mathbf{J}^{\mathrm{TCP,R}} & \mathbf{J}^{\mathrm{TCP,Z}} \\ \hline \mathbf{J}^{\mathrm{G,R}} & \mathbf{J}^{\mathrm{G,Z}} \end{array}\right)^{-1} \cdot \begin{pmatrix} \dot{\vec{x}}^{\mathrm{TCP}} \\ \dot{\vec{x}}^{G_1} \\ \vdots \\ \dot{\vec{x}}^{G_{n-1}} \end{pmatrix} = \begin{pmatrix} \dot{q}_1^{\mathrm{R}} \\ \vdots \\ \dot{q}_m^{\mathrm{R}} \\ \hline \vec{0} \end{pmatrix} \tag{C.5}$$

Gleichung C.5 hat die bekannte Matrix-Vektor-Form der Zwangsbedingungen wie in einer geschwindigkeitsbasierten Formulierung mit Lagrange-Multiplikatoren in Kapitel 3.1.2, Seite 79. Die Matrix $\mathbf{J}_{\mathrm{Rob}}^{-1}$ entspricht also einer Jacobimatrix von Zwangsbedingungen.

Zerlegt man $\mathbf{J}_{\mathrm{Zwa}} = \mathbf{J}_{\mathrm{Rob}}^{-1}$ noch in die beiden Teile $\mathbf{J}_{\mathrm{Zwa,Motor}}$, bestehend aus den oberen m Zeilen und in $\mathbf{J}_{\mathrm{Zwa,Gelenk}}$, bestehend aus den unteren $6n - m$ Zeilen von $\mathbf{J}_{\mathrm{Rob}}^{-1}$, so erhält man sogar einzelne Jacobimatrizen für Motor- (vgl. Kapitel 3.2.8, Seite 103) und Gelenkzwangsbedingungen (vgl. Kapitel 3.2.3 - 3.2.6, Seite 92 ff.). Der Vektor $\dot{\vec{q}}$ entspricht hier den Sollgeschwindigkeiten der Zwangsbedingungen ($\vec{b}$):

$$\begin{pmatrix} \mathbf{J}^{\mathrm{TCP,R}} & \mathbf{J}^{\mathrm{TCP,Z}} \\ \mathbf{J}^{\mathrm{G,R}} & \mathbf{J}^{\mathrm{G,Z}} \end{pmatrix}^{-1} \cdot \begin{pmatrix} \dot{\vec{x}}^{\mathrm{TCP}} \\ \dot{\vec{x}}^{G_1} \\ \vdots \\ \dot{\vec{x}}^{G_{n-1}} \end{pmatrix} = \underbrace{\begin{pmatrix} \mathbf{J}_{\mathrm{Zwa,Motor}} \\ \hline \mathbf{J}_{\mathrm{Zwa,Gelenk}} \end{pmatrix}}_{\mathbf{J}_{\mathrm{Zwa}}} \cdot \begin{pmatrix} \dot{\vec{x}}^{\mathrm{TCP}} \\ \dot{\vec{x}}^{G_1} \\ \vdots \\ \dot{\vec{x}}^{G_{n-1}} \end{pmatrix} = \underbrace{\begin{pmatrix} \dot{q}_1^{\mathrm{R}} \\ \vdots \\ \dot{q}_m^{\mathrm{R}} \\ \hline \vec{0} \end{pmatrix}}_{\dot{\vec{q}}} \tag{C.6}$$

Ein direkter, element-, spalten- oder zeilenweiser Zusammenhang zwischen den Matrizen $\mathbf{J}_{\mathrm{Zwa,Motor}}, \mathbf{J}_{\mathrm{Zwa,Gelenk}}$ und $\mathbf{J}^{\mathrm{TCP,R}}, \mathbf{J}^{\mathrm{TCP,Z}}, \mathbf{J}^{\mathrm{G,R}}, \mathbf{J}^{\mathrm{G,Z}}$ ergibt sich hieraus allerdings nicht. Auch ist zu berücksichtigen, dass sich die Relation im Allgemeinen nicht umkehren lässt. Dies gilt, weil die Jacobimatrix der Zwangsbedingungen nicht $6n$ Zeilen haben muss, sondern auch weniger haben darf. Hieraus entstünde ein unterbestimmtes System.

Literaturverzeichnis

[1] *Bullet Physics Library*. http://sourceforge.net/projects/bullet/, Abruf: Aug. 2009

[2] *Deutsches Forschungsinstitut für Künstliche Intelligenz (DFKI) – Robotics Innovation Center*. http://robotik.dfki-bremen.de/

[3] *Initiative zur rechnerintegrierten Fertigung (RIF) e.V.* http://www.rif-ev.de

[4] *Institut für Mensch-Maschine-Interaktion (MMI) der RWTH Aachen*. http://www.mmi.rwth-aachen.de

[5] ANITESCU, Mihai ; CREMER, James F. ; POTRA, Florian A.: Formulating 3D Contact Dynamics Problems. In: *Mechanics of Structures and Machines* 24 (1996), Nov., Nr. 4, S. 405–437. http://dx.doi.org/10.1080/08905459608905271. – DOI 10.1080/08905459608905271

[6] ANITESCU, Mihai ; POTRA, Florian: Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. In: *Nonlinear Dynamics* 14 (1997), 231–247. http://dx.doi.org/10.1023/A:1008292328909. – DOI 10.1023/A:1008292328909

[7] ANITESCU, Mihai ; POTRA, Florian A.: A time-stepping method for stiff multibody dynamics with contact and friction. In: *International Journal for Numerical Methods in Engineering* 55 (2002), S. 753–784

[8] ANITESCU, Mihai ; TASORA, Alessandro: An iterative approach for cone complementarity problems for nonsmooth dynamics. In: *Computational Optimization and Applications* (2008), November. http://dx.doi.org/10.1007/s10589-008-9223-4. – DOI 10.1007/s10589–008–9223–4

[9] BARAFF, David: Curved Surfaces and Coherence for Non-penetrating Rigid Body Simulation. In: *Proceedings of the 1990 ACM SIGGRAPH conference* Bd. 24. New York, NY, USA : ACM, 1990, S. 19–28

[10] BARAFF, David: *Dynamic Simulation of Non-Penetrating Rigid Bodies*. Ithaca, NY, USA, Department of Computer Science, Cornell University, Diss., March 1992

[11] BARAFF, David: Issues in Computing Contact Forces for Non-Penetrating Rigid Bodies. In: *Algorithmica* 10 (1993), October, Nr. 2–4, S. 292 – 352

[12] BARAFF, David: Fast Contact Force Computation for Nonpenetrating Rigid Bodies. In: *SIGGRAPH '94: Proceedings of the 21rd annual conference on Computer graphics and interactive techniques* Bd. 28, 1994, 23–34

[13] BARAFF, David: Linear-Time Dynamics using Lagrange Multipliers. In: *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* Bd. 30 ACM, 1996, 137–146

[14] BARAFF, David ; WITKIN, Andrew: Large Steps in Cloth Simulation. In: *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM New York, NY, USA, 1998, S. 43–54

[15] BARTSCH, Sebastian ; BIRNSCHEIN, Timo ; CORDES, Florian ; KÜHN, Daniel ; KAMPMANN, Peter ; HILLJEGERDES, Jens ; PLANTHABER, Steffen ; RÖMMERMANN, Malte ; KIRCHNER, Frank: SpaceClimber: Developement of a Six-Legged Climbing Robot for Space Exploration. In: *Proceedings for the joint conference of ISR 2010 (41st Internationel Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics), (ISR)*, 2010

[16] BAUMGARTE, J.: Stabilization of constraints and integrals of motion in dynamical systems. In: *Computer Methods in Applied Mechanics and Engineering* 1 (1972), S. 1–16

[17] BENDER, Jan: *Impulsbasierte Dynamiksimulation von Mehrkörpersystemen in der virtuellen Realität*, Universität Karlsruhe, Deutschland, Diss., 2007

[18] BENDER, Jan: Impulse-based dynamic simulation in linear time. In: *Computer Animation and Virtual Worlds* 18 (2007), Nr. 4-5, S. 225–233. http://dx.doi.org/http://dx.doi.org/10.1002/cav.v18:4/5. – DOI http://dx.doi.org/10.1002/cav.v18:4/5

[19] BENDER, Jan ; BAAS, Matthias ; SCHMITT, Alfred: Ein neues Verfahren für die mechanische Simulation in VR-Systemen und in der Robotik. In: *17. Symposium Simulationstechnik, ASIM 2003*, 2003, S. 111–116

[20] BENDER, Jan ; FINKENZELLER, Dieter ; SCHMITT, Alfred: An impulse-based dynamic simulation system for (VR) applications. In: *Proceedings of Virtual Concept 2005*, 2005

[21] BENDER, Jan ; SCHMITT, Alfred: Constraint-based collision and contact handling using impulses. In: *Proceedings of the 19th international conference on computer animation and social agents, Geneva, Switzerland*, 2006

[22] BENDER, Jan ; SCHMITT, Alfred: Fast Dynamic Simulation of Multi-Body Systems Using Impulses. In: *Virtual Reality Interactions and Physical Simulations (VRIPhys)*, 2006

[23] BENDER, Michael ; BRILL, Manfred: *Computergrafik: Ein anwendungsorientiertes Lehrbuch*. 2. Auflage. Hanser Fachbuchverlag, 2005

[24] BENTLEY, Jon L.: Multidimensional binary search trees used for associative searching. In: *Communications of the ACM* 18 (1975), Nr. 9, S. 509–517

[25] BERGEN, Gino van d.: *SOLID Collision Detection Library*. `http://www.dtecta.com/`, Abruf: Okt. 2009

[26] BERGEN, Gino van d.: Efficient Collision Detection of Complex Deformable Models using AABB Trees. In: *Journal of Graphics, GPU, and Game Tools* 2 (1997), April, Nr. 4, S. 1–14

[27] BERGEN, Gino van d.: A Fast and Robust GJK Implementation for Collision Detection of Convex Objects. In: *Journal of Graphics Tools* 4 (1999), Nr. 2, S. 7–25

[28] BERGEN, Gino van d.: Proximity Queries and Penetration Depth Computation on 3D Game Objects. In: *GDC (Game Developers Conference) Proceedings 2001*, 2001

[29] BRADSHAW, Gareth ; O'SULLIVAN, Carol: Sphere-Tree Construction using Dynamic Medial Axis Approximation. In: *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*. New York, NY, USA : ACM, 2002, S. 33–40

[30] BRADSHAW, Gareth ; O'SULLIVAN, Carol: Adaptive medial-axis approximation for sphere-tree construction. In: *ACM Transactions on Graphics (TOG)* 23 (2004), January, Nr. 1, S. 1–26

[31] BRONSTEIN, Ilja N. ; SEMENDJAJEW, K. A. ; MUSIOL, Gerhard ; MUEHLIG, Heiner: *Taschenbuch der Mathematik*. 4., überarbeitete und erweiterte Auflage der Neubearbeitung. Verlag Harri Deutsch, 1999

[32] CATTO, Erin: *Fast and Simple Physics using Sequential Impulses*. Präsentation gehalten auf der Game Developers Conference (GDC) 2006. `http://www.gphysics.com/downloads`. Version: 2006, Abruf: Okt. 2009

[33] CHRYSANTHOU, Y. ; SLATER, M.: Computing Dynamic Changes to BSP trees. In: *Computer Graphics Forum (EUROGRAPHICS '92 Proceedings)* Bd. 11, 1992, S. 321–332

[34] CLINE, Michael B. ; PAI, Dinesh K.: Post-stabilization for rigid body simulation with contact and constraints. In: *Proceedings of the 2003 IEEE International Conference on Robotics and Automation* Bd. 3, 2003, S. 3744–3751

[35] CLINE, Michael B.: *Rigid Body Simulation with Contact and Constraints*, The University of Texas at Austin, Diplomarbeit, 2002

[36] COHEN, Jonathan D. ; LIN, Ming C. ; MANOCHA, Dinesh ; PONAMGI, Madhav: I-COLLIDE: an interactive and exact collision detection system for large-scale environments. In: *SI3D '95: Proceedings of the 1995 symposium on Interactive 3D graphics*. New York, NY, USA : ACM Press, 1995, S. 189–ff.

[37] CORDES, Florian ; AHRNS, Ingo ; BARTSCH, Sebastian ; BIRNSCHEIN, Timo ; DETTMANN, Alexander ; ESTABLE, Stéphane ; HAASE, Stefan ; HILLJEGERDES, Jens ; KOEBEL, David ; PLANTHABER, Steffen ; RÖHR, Thomas ; SCHEPER, Marc ; KIRCHNER, Frank: LUNARES: Lunar Crater Exploration with Heterogeneous Multi Robot Systems. In: *Journal on Intelligent Service Robotics* o.A. (2010), S. o.A.

[38] COTTLE, Richard W. ; DANTZIG, George B.: Complementary Pivot Theory of Mathematical Programming. In: *Linear Algebra and its Applications* 1 (1968), S. 103–125

[39] COTTLE, Richard W. ; DANTZIG, George B.: A Generalization of the Linear Complementarity Problem. In: *Journal of Combinatorial Theory* 8 (1970), S. 79–90

[40] DAHMEN, W. ; REUSKEN, A.: *Numerik für Ingenieure und Naturwissenschaftler*. Springer-Verlag GmbH, 2006

[41] DLR: *Ongoing Space Missions: TECSAS / DEOS*. Website. `http://www.dlr.de/rm/desktopdefault.aspx/tabid-3825/5963_read-8759/`, Abruf: Okt. 2010

[42] DRUMWRIGHT, Evan: A Fast and Stable Penalty Method for Rigid Body Simulation. In: *IEEE Transactions on Visualization and Computer Graphics* 14 (2008), Nr. 1, S. 231–240

[43] ERLEBEN, Kenny: *Stable, Robust, and Versatile Multibody Dynamics Animation*, The Department of Computer Science, University of Copenhagen, Denmark, Diss., 2005. `ftp://ftp.diku.dk/diku/image/publications/erleben.050401.pdf`

[44] ERLEBEN, Kenny: Velocity-Based Shock Propagation for Multibody Dynamics Animation. In: *ACM Transactions on Graphics* 26 (2007), June, Nr. 2. `http://dx.doi.org/10.1145/1243980.1243986`. – DOI 10.1145/1243980.1243986

[45] FEATHERSTONE, Roy: The calculation of robot dynamics using articulated-body inertias. In: *International Journal of Robotics Research* 2 (1983), Nr. 13, S. 13–30. `http://dx.doi.org/10.1177/027836498300200102`. – DOI 10.1177/027836498300200102

[46] FEATHERSTONE, Roy: *Rigid Body Dynamics Algorithms*. 2., überarbeitete Auflage. Springer Verlag, 2007

[47] FEATHERSTONE, Roy ; ORIN, David: Robot Dynamics: Equations and Algorithms. In: *Proceedings of the IEEE International Conference on Robotics and Automation ICRA 00* Bd. 1, IEEE Computer Society, 2000, S. 826–834

[48] FOLEY, James D. ; VANDAM, Andries ; FEINER, Steven K.: *Computer Graphics: Principles and Practice*. Addison-Wesley Longman, Amsterdam, 1995

[49] FREUND, Eckhard ; ROSSMANN, Jürgen: Cybernetic Background of the Development of a Virtual Reality Based Wood Harvester Simulator. In: *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI)*, 2002

[50] FREUND, Eckhard ; ROSSMANN, Jürgen ; KRÄMER, Michael: Robots in the Woods: The Challenges to Simulate a Wood Harvester. In: *VDI-Berichte* 1679 (2002), S. 633–639

[51] GAMMA, Erich ; HELM, Richard ; JOHNSON, Ralph ; VLISSIDES, John: *Design patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Longman Publishing Co., Inc., 1995

[52] GIANG, Thanh ; BRADSHAW, Gareth ; O'SULLIVAN, Carol: Complementarity based multiple point collision resolution. In: *Proceedings of the Fourth Irish Workshop on Computer Graphics*, 2003

[53] GILBERT, Elmer G. ; JOHNSON, Daniel W. ; KEERTHI, S. S.: A Fast Procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space. In: *IEEE Journal of Robotics and Automation* 4 (1988), April, Nr. 2, S. 193–203

[54] GLOCKER, Christopher ; PFEIFFER, Friedrich: Multiple Impacts with Friction in Rigid Multibody Dystems. In: *Nonlinear Dynamics* 7 (1995), S. 471–497

[55] GOTTSCHALK, Stefan ; LIN, Ming C. ; MANOCHA, D.: OBB-Tree: A Hierarchical Structure for Rapid Interference Detection. In: *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. New York, NY, USA : ACM, 1996, S. 171–180

[56] GROSS, Dietmar ; HAUGER, Werner ; SCHNELL, Walter ; SCHRÖDER, Jörg: *Technische Mechanik 3*. 8., erweiterte Auflage. Springer Verlag, 2004

[57] GUENDELMAN, Eran ; BRIDSON, Robert ; FEDKIW, Ronald: Nonconvex rigid bodies with stacking. In: *ACM Transactions on Graphics* 22 (2003), July, Nr. 3, S. 871–878. http://dx.doi.org/10.1145/882262.882358. – DOI 10.1145/882262.882358

[58] HAHN, Hubert: *Rigid Body Dynamics of Mechanisms*. Bd. 1. Theoretical basis. Springer Verlag, 2002

[59] HAHN, James K.: Realistic Animation of Rigid Bodies. In: *Computer Graphics* Bd. 22, 1988, S. 299–308

[60] HEYN, Toby ; TASORA, Alessandro ; ANITESCU, Mihai ; NEGRUT, Dan: A Parallel Algorithm for Solving Complex Multibody Problems with Stream Processors. In: *Proceedings of the ASME 2009 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE)*, 2009

[61] HILLJEGERDES, Jens ; SPENNEBERG, Dirk ; KIRCHNER, Frank: The Construction of the Four-Legged Prototype Robot ARAMIES. In: *Proceedings of the Eighth International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines and Introductory Workshop on Mobile Robotics in London*, 2005

[62] HUBBARD, Philip M.: *Collision Detection for Interactive Graphics Applications*, Department of Computer Science, Brown University, Providence, Rhode Island, Diss., April 1995

[63] HUBBARD, Philip M.: Approximating polyhedra with spheres for time-critical collision detection. In: *ACM Transactions on Graphics (TOG)* 15 (1996), Juli, Nr. 3, S. 179–210

[64] IGLBERGER, Klaus ; RÜDE, Ulrich: Massively Parallel Rigid Multi-Body Dynamics / Institut für Informatik, Friedrich-Alexander-Universität Erlangen-Nürnberg. Version: 2009. `http://www10.informatik.uni-erlangen.de/Publications/TechnicalReports/TechRep09-8.pdf`. 2009. – Forschungsbericht

[65] IGLBERGER, Klaus ; RÜDE, Ulrich: A Parallel Rigid Body Dynamics Algorithm. In: *Lecture Notes in Computer Science* 5704/2009 (2009), S. 760–771

[66] IGLBERGER, Klaus ; RÜDE, Ulrich: Massively Parallel Granular Flow Simulations with Non-Spherical Particles. In: *Computer Science - Research and Development* 25 (2010), Nr. 1-2, S. 105–113

[67] JALÓN, Javier G. ; BAYO, Eduardo ; LING, Frederick F. (Hrsg.): *Kinematic and Dynamic Simulation of Multibody Systems*. Springer Ver, 1994 (Mechanical Engineering Series)

[68] JEFFERSON, David R.: Virtual Time. In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 7 (1985), July, Nr. 3, S. 404–425. `http://dx.doi.org/10.1145/3916.3988`. – DOI 10.1145/3916.3988

[69] JONES, Mark W. ; BÆRENTZEN, J. A. ; ŠRÁMEK, Miloš: 3D Distance Fields: A Survey of Techniques and Applications. In: *IEEE Transactions on Visualization and Computer Graphics* 12 (2006), Nr. 4, S. 581–599. `http://dx.doi.org/10.1109/TVCG.2006.56`. – DOI 10.1109/TVCG.2006.56

[70] JUNG, Thomas J. ; ROSSMANN, Jürgen: Realisierung von Simulatoren für Forstmaschinen für den Einsatz in der Maschinenführerausbildung. In: *10. IFF Wissenschaftstage: Virtual Reality und Augmented Reality zum Planen, Testen und Betreiben technischer Systeme*, Fraunhofer Institut für Fabrikbetrieb und Automatisierung, 2007

[71] KAIGOM, Eric ; ROSSMANN, Juergen ; JUNG, Thomas J.: Constrained PSO based Optimal Motion Planning of a Space Robot with Base Disturbance Minimization. In: *Proceedings of the 11th Symposium on Advanced Space Technologies in Robotics and Automation*, 2011

[72] KAUFMAN, Danny M. ; EDMUNDS, Timothy ; PAI, Dinesh K.: Fast frictional dynamics for rigid bodies. In: *Proceedings of ACM SIGGRAPH 2005* Bd. 24, 2005, S. 946–956

[73] KAVAN, Ladislav: Rigid Body Collision Response. In: *Proceedings of the 7th Central European Seminar on Computer Graphics*, 2003

[74] KIM, Young J. ; OTADUY, Miguel A. ; LIN, Ming C. ; MANOCHA, Dinesh: Fast penetration depth computation for physically-based animation. In: *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2002, S. 23–31

[75] KLOSOWSKI, James T. ; HELD, Martin ; MITCHELL, Joseph S. ; SOWIZRAL, Henry: Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs. In: *IEEE Transactions on Visualization and Computer Graphics* 4 (1998), Nr. 1, S. 21–36

[76] KUYPERS, Friedhelm: *Klassische Mechanik*. 8. erweiterte Auflage. WILEY-VCH Verlag GmbH & Co. KGaA, 2008

[77] LEMKE, C. E.: On complementary pivot theory. In: *Mathematics in the Decision Sciences* Bd. 1. G. B. Dantzig and A. F. Veinott, 1968, S. 95–114

[78] LIN, Ming C.: *Effcient Collision Detection for Animation and Robotics*. Berkeley, California, USA., Department of Electrical Engineering and Computer Science, University of California, Diss., 1993

[79] LIN, Ming C. ; CANNY, John F.: A Fast Algorithm for Incremental Distance Calculation. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, 1991, S. 1008–1014

[80] LIN, Ming C. ; GOTTSCHALK, Stefan: Collision Detection Between Geometric Models: A Survey. In: *Proc. of IMA Conference on Mathematics of Surfaces*, 1998, S. 37–56

[81] LÖTSTEDT, Per: Mechanical systems of rigid bodies subject to unilateral constraints. In: *SIAM Journal of Applied Mathematics* 42 (1982), April, 281–296. http://www.jstor.org/stable/2101212, Abruf: Nov. 2009

[82] LUH, J. Y. S. ; WALKER, M. W. ; PAUL, R. P. C.: On-line computational scheme for mechanical manipulators. In: *Journal of Dynamic Systems, Measurement, and Control* 102 (1980), Nr. 2, S. 69–77

[83] LUQUE, Rodrigo G. ; COMBA, João L. D. ; FREITAS, Carla M. D. S.: Broad-phase collision detection using semi-adjusting BSP-trees. In: *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, 2005, S. 179–186

[84] MAZHAR, Hammad: GPU Collision Detection Using Spatial Subdivision With Applications in Contact Dynamics. In: *Proceedings of the ASME 2009 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE 2009*, 2009

[85] MILLER, Andrew T. ; CHRISTENSEN, Henrik I.: Implementation of Multi-rigid-body Dynamics within a Robotic Grasping Simulator. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* Bd. 2, 2003, S. 2262–2268

[86] MIRTICH, Brian: Hybrid Simulation: Combining Constraints and Impulses. In: *Proceedings of First Workshop on Simulation and Interaction in Virtual Environments*, 1995

[87] MIRTICH, Brian: V-Clip: Fast and Robust Polyhedral Collision Detection / MERL—A Mitsubishi Electric Research Laboratory. 1998. – Forschungsbericht

[88] MIRTICH, Brian V.: *Impulse-based dynamic simulation of rigid body systems*, University of California at Berkely, Diss., 1996

[89] MIRTICH, Brian V.: Rigid Body Contact: Collision Detection to Force Computation / Mitsubishi Electric Research Laboratories. 1998. – Forschungsbericht

[90] MIRTICH, Brian V.: Timewarp Rigid Body Simulation / Mitsubishi Electric Research Laboratories. 2000. – Forschungsbericht

[91] MIRTICH, Brian V. ; CANNY, John: Impulse-based Simulation of Rigid Bodies. In: *SI3D '95: Proceedings of the 1995 symposium on Interactive 3D graphics*. New York, NY, USA : ACM, 1995, S. 181 ff.

[92] MÜLLER, Matthias: Hierarchical Position Based Dynamics. In: *VRIPHYS 08: Proceedings of the 5th Workshop on Virtual Reality Interaction and Physical Simulation*, 2008

[93] MÜLLER, Matthias ; HEIDELBERGER, Bruno ; HENNIX, Marcus ; RATCLIFF, John: Position Based Dynamics. In: MENDOZA, C. (Hrsg.) ; NAVAZO, I. (Hrsg.): *VRIPHYS '06: Proceedings of the 3rd Workshop in Virtual Reality Interactions and Physical Simulation*, 2006

[94] MÜLLER, Matthias ; HEIDELBERGER, Bruno ; HENNIX, Marcus ; RATCLIFF, John: Position based dynamics. In: *Journal of Visual Communication and Image Representation* 18 (2007), April, Nr. 2, S. 109–118

[95] MOORE, Matthew ; WILHELMS, Jane: Collision Detection and Response for Computer Animation. In: *Computer Graphics* Bd. 22, 1988, S. 289–298

[96] NAYLOR, Bruce F.: Interactive Solid Geometry Via Partitioning Trees. In: *Proceedings of Graphics Interface '92*, 1992

[97] ONOUE, Koichi ; NISHITA, Tomoyuki: Virtual Sandbox. In: *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, 2003, S. 252

[98] O'SULLIVAN ; DINGLIANA, J. C.: Real-Time Collision Detection and Response Using Sphere-Trees. In: *Proceedings of the Spring Conference in Computer Graphics*, 1999, S. 83–92

[99] PLA-CASTELLS, Marta ; GARCÍA-FERNÁNDEZ, Ignacio ; MARTÍNEZ, Rafael J.: Interactive Terrain Simulation and Force Distribution Models in Sand Piles. In: *Lecture Notes in Computer Science* 4173 (2006), S. 392–401. http://dx.doi.org/10.1007/11861201_46. – DOI 10.1007/11861201_46

[100] PONSSE: *Homepage*. http://www.ponsse.com. Version: 2010. – Finnland

[101] ROCHA, R. S. ; RODRIGUES, M. A. F.: Investigating Broad Phase Collision Detection Methods for 3D Scenarios Using Force Feedback Devices. In: *Proceedings of the XXXII Conferência Latino-Americana de Informática - CLEI'06*, 2006

[102] ROSSMANN, Jürgen: *Echtzetfähige, kollisionsvermeidende Bahnplanung für Mehrrobotersysteme*, Universität Dortmund, Diss., 1993

[103] ROSSMANN, Jürgen ; ALVES, Gerrit: Parallel Timestep Simulation Scheduling (PTSS) with Variable Time Increments for Factory Simulation Applications. In: *Proceedings of the 13th IEEE International Conference on Emerging Technologies and Factory Automation*, 2008

[104] ROSSMANN, Jürgen ; JUNG, Thomas J.: Dynamiksimulation für Virtuelle Welten: Erfahrungen, Anwendungen, Methoden. In: *7. Paderborner Workshop Augmented & Virtual Reality in der Produktentstehung*, 2008, S. 61–76

[105] ROSSMANN, Jürgen ; JUNG, Thomas J. ; RAST, Malte: Developing Virtual Testbeds for Mobile Robotic Applications in the Woods and on the Moon. In: *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010

[106] ROSSMANN, Jürgen ; JUNG, Thomas J. ; RAST, Malte: Developing Virtual Testbeds for Tasks in Research and Engineering. In: *Proceedings of the ASME 2010 World Conference on Innovative Virtual Reality*, 2010

[107] ROSSMANN, Jürgen ; JUNG, Thomas J. ; RAST, Malte: Entwicklung Virtueller Testbeds mit Dynamik- und Bodenmechaniksimulation für Aufgaben in Forschung und Entwicklung. In: *9. Paderborner Workshop Augmented & Virtual Reality in der Produktentstehung*, 2010, S. 173–187. – Ausgezeichnet mit dem Best-Paper-Award.

[108] ROSSMANN, Jürgen ; SCHLUSE, Michael ; BÜCKEN, Arno ; JUNG, Thomas ; KRAHWINKLER, Petra: Der Virtuelle Wald in NRW. In: *AFZ Der Wald* 18 (2007), S. 966–971

[109] ROSSMANN, Jürgen ; SCHLUSE, Michael ; HOPPEN, Martin ; WASPE, Ralf: GIS-Based Virtual Testbeds and their Application to Forestry and City Simulation. In: *Proceedings of the ASME 2010 World Conference on Innovative Virtual Reality*, 2010

[110] ROSSMANN, Jürgen ; SCHLUSE, Michael ; JUNG, Thomas J. ; RAST, Malte: Close to Reality Simulation of Bulk Solids Using a Kind of 3D Cellular Automaton. In: *Proceedings of the ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, IDETC/CIE 2009, San Diego, CA, USA*, 2009

[111] ROSSMANN, Jürgen ; SCHLUSE, Michael ; JUNG, Thomas J. ; RAST, Malte: Interaktive integrierte Starrkörperdynamik- und Schüttgutsimulation. In: *8. Paderborner Workshop Augmented & Virtual Reality in der Produktentstehung*, 2009, S. 31–48

[112] RYU, Geunsoo ; MA, Zheng-Dong ; HULBERT, Gregory M.: Integration of Finite Element and Multibody Dynamics Models Using Gluing Algorithm. In: *the ASME 2009 Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE)*, 2009

[113] SAUER, Jörg ; SCHÖMER, Elmar: A Constraint-Based Approach to Rigid Body Dynamics for Virtual Reality Applications. In: *Virtual Reality Software and Technology, Proceedings of the ACM symposium on Virtual reality software and technology*, ACM New York, NY, USA, 1998, S. 153–161

[114] SAUER, Jörg ; SCHÖMER, Elmar: Dynamiksimulation starrer Körper für Virtual Reality Anwendungen. In: *12. Symposium Simulationstechnik, ASIM '98*, 1998, S. 355–362

[115] SAUER, Jörg ; SCHÖMER, Elmar ; LENNERZ, Christian: Real-Time Rigid Body Simulations of some 'Classical Mechanics Toys'. In: *10. European Simulation Symposium and Exhibition, ESS '98*, 1998, S. 93–98

[116] SCHIEHLEN, Werner ; EBERHARD, Peter: *Technische Dynamik Modelle für Regelung und Simulation*. 2., neubearbeitete und ergänzte Auflage. B. G. Teubner, 2004

[117] SCHLUSE, Michael: *Zustandsorientierte Modellierung in Virtueller Realität und Kollisionsvermeidung*, Universität Dortmund, Internal Report, 2002

[118] SCHMITT, Alfred: Dynamische Simulation von gelenkgekoppelten Starrkörpersystemen mit der Impulstechnik / Institut für Betriebs- und Dialogsysteme, Fakultät für Informatik, Universität Karlsruhe. Version: 2003. `http://digbib.ubka.uni-karlsruhe.de/volltexte/4722003`, Abruf: Okt. 2009. 2003. – Forschungsbericht

[119] SCHMITT, Alfred ; BENDER, Jan: Impulse-Based Dynamic Simulation of Multibody Systems: Numerical Comparison with Standard Methods. In: *Proc. Automation of Discrete Production Engineering*, 2005, 324–329

[120] SCHMITT, Alfred ; BENDER, Jan ; PRAUTZSCH, Hartmut: Impulse-Based Dynamic Simulation of Higher Order and Numerical Results / Institut für Betriebs- und Dialogsysteme, Fakultät für Informatik, Universität Karlsruhe. 2005 (21). – Internal Report

[121] SMITH, Russel: *Open Dynamics Engine*. `http://www.ode.org`, Abruf: Aug. 2009

[122] SNYDER, John ; LENGYEL, Jed: Visibility sorting and compositing without splitting for image layer decompositions. In: *Computer Graphics Proceedings, Annual Conference Series*, 1998, S. 219–230

[123] SON, Wookho ; TRINKLE, Jeffrey C. ; AMATO, Nancy M.: Hybrid Dynamic Simulation of Rigid-Body Contact with Coulomb Friction. In: *Proceedings of the 2001 IEEE Internbational Conference on Robotics & Automation (IROS)*, 2001, S. 1376–1381

[124] SPENNEBERG, Dirk ; ALBRECHT, M. ; BACKHAUS, Till ; HILLJEGERDES, Jens ; KIRCHNER, Frank ; STRACK, A. ; ZSCHENKER, H.: ARAMIES: A Four-Legged Climbing and Walking Robot. In: *Proceedings of 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS)*, 2005

[125] *Kapitel* 9. In: SPENNEBERG, Dirk ; KIRCHNER, Frank: *The Bio-Inspired SCORPION Robot: Design,Control & Lessons Learned*. I-Tech Education and Publishing, 2007, S. 197–218

[126] STEWART, David ; TRINKLE, Jeffery C.: An Implicit Time-Stepping Scheme for Rigid Body Dynamics with Inelastic Collisions and Coulomb Friction. In: *International Journal for Numerical Methods in Engineering* 39 (1996), S. 2673–2691

[127] STEWART, David ; TRINKLE, Jeffery C.: An Implicit Time-Stepping Scheme for Rigid Body Dynamics with Coulomb Friction. In: *IEEE International Conference on Robots and Automation*, 2000, S. 162 – 169

[128] *Kapitel* 9. In: STEWART, David E.: *Impact and Friction of Solids, Structures and Machines*. Birkhäuser, 2000

[129] STRONGE, W. J.: Unraveling Paradoxical Theories for Rigid Body Collisions. In: *Journal of Applied Mechanics* 58 (1991), Dezember, Nr. 4, S. 1049–1055

[130] TASORA, Alessandro ; ANITESCU, Mihai: A Convex Complementarity Approach for Simulating Large Granular Flow. In: *Journal of Computational Nonlinear Dynamics* 5 (2010), Nr. 3. http://dx.doi.org/10.1115/1.4001371. – DOI 10.1115/1.4001371

[131] TASORA, Alessandro ; NEGRUT, Dan ; ANITESCU, Mihai: Large-Scale Parallel Multibody Dynamics With Frictional Contact on the GPU. In: *Proceedings of the ASME Dynamic Systems and Control Conference* ASME, 2008

[132] TRACY, Daniel J. ; BUSS, Samuel R. ; WOODS, Bryan M.: Efficient Large-Scale Sweep and Prune Methods with AABB Insertion and Removal. In: *Proceedings of the 2009 IEEE Virtual Reality Conference*. Washington, DC, USA : IEEE Computer Society, 2009, S. 191–198

[133] TRINKLE, J. C.: Formulation of Multibody Dynamics as Complementary Problems. In: *Proceedings of DETC'03 ASME 2003 Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2003

[134] TSENG, Fang-Chung ; HULBERT, Gregory M.: A Gluing Algorithm for Network-Distributed Multibody Dynamics Simulation. In: *Multibody System Dynamics* 6 (2001), S. 159–188

[135] WAGNER, Friedrich: *Konzepte und Methoden zu allgemeinen, physikalisch basierten Animationssystemen auf der Grundlage der Lagrange-Faktoren Methode*, Universität Rostock, Diss., 2001

[136] WANG, Jinzhong ; MA, Zheng-Dong ; HULBERT, Gregory M.: A Gluing Algorithm for Distributed Simulation of Multibody Systems. In: *Nonlinear Dynamics* 34

(2003), Nr. 1-2, S. 159–188. http://dx.doi.org/10.1023/B:NODY.0000014558.70434.b0. – DOI 10.1023/B:NODY.0000014558.70434.b0

[137] WEINSTEIN, Rachel ; TERAN, Joseph ; FEDKIW, Ron: Pre-stabilization for Rigid Body Articulation with Contact and Collision. In: *International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, ACM, 2005

[138] WEINSTEIN, Rachel ; TERAN, Joseph ; FEDKIW, Ron: Dynamic Simulation of Articulated Rigid Bodies with Contact and Collision. In: *IEEE Transactions on Visualization and Computer Graphics* 12 (2006), Nr. 3, S. 365–374. http://dx.doi.org/10.1109/TVCG.2006.48, Abruf: Okt. 2009. – DOI 10.1109/TVCG.2006.48

[139] WEINSTEIN, Rachel L.: *Simulation and Control of Articulated Rigid Bodies*, Stanford University, Departement of Computer Science, Diss., 2007

[140] WELLER, Rene ; ZACHMANN, Gabriel: Inner Sphere Trees / Technische Universität Clausthal, Institut für Informatik. 2008-2009. – Forschungsbericht

[141] WELLER, Rene ; ZACHMANN, Gabriel: Inner sphere trees for proximity and penetration queries. In: *Proceedings of Robotics: Science and Systems*, 2009

[142] YAMANE, Katsu ; NAKAMURA, Yoshihiko: Stable Penalty-Based Model of Frictional Contacts. In: *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, 2006

[143] YOO, Yong-Ho ; JUNG, Thomas ; RÖMMERMANN, Malte ; RAST, Malte ; KIRCHNER, Frank ; ROSSMANN, Jürgen: Developing a Virtual Environment for Extraterrestrial Legged Robot with Focus on Lunar Crater Exploration. In: *Proceedings of the 10th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2010

[144] ZACHMANN, Gabriel: Real-time and exact collision detection for interactive virtual prototyping. In: *Proceedings of DETC'97 1997 ASME Design Engineering Technical Conferences*, 1997

[145] ZHU, Yongning ; BRIDSON, Robert: Animating sand as a fluid. In: *International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2005