Interaktive Simulation biomechanischer Bewegungsabläufe

DISSERTATION

zur Erlangung des Grades eines Doktors

der Naturwissenschaften

der Fakultät für Mathematik und Physik
der Eberhard-Karls-Universität zu Tübingen

vorgelegt von

Torsten Hans
aus Wilhelmshaven

2004

Tag der mündlichen Prüfung:	07.05.2004
Dekan:	Prof. Dr. Herbert Müther
1. Berichterstatter:	Prof. Dr. Hanns Ruder
2. Berichterstatter:	Prof. Dr. Thomas Ertl

Zusammenfassung

Ziel dieser Arbeit war die Entwicklung eines allgemeinen biomechanischen Menschmodells, das unter dem Gesichtspunkt einer schnellen interaktiven Simulation in Wechselwirkung mit seiner Umwelt treten kann. Schnell bedeutet hierbei, dass spätestens innerhalb von wenigen Sekunden das gewünschte Simulationsergebnis vorliegt. Interaktiv heißt zum einen, dass während der Berechnung jederzeit eine grafische Darstellung der Simulationsszene erfolgt und zum anderen, dass der Anwender während der Simulation die Möglichkeit hat aktiv einzugreifen. Die Genauigkeit der Simulation soll dabei keineswegs vernachlässigt werden.

Erreicht wird dies vor allem indem bei der Kollisionserkennung neue Wege gegangen werden und bei der Aufstellung der Bewegungsgleichungen eine spezielle Struktur erstellt wird.

Dabei wird eine starke Verknüpfung zwischen der 3-dimensionalen Grafik und der Simulation hergestellt, so dass beide eng miteinander verbunden sind. Das heißt zum einen, dass die physikalischen Parameter wie zum Beispiel Masse und Trägheitstensor direkt aus der 3D-Flächendarstellung eines Körpers berechnet werden und zum anderen, dass die Simulation die 3D-Grafik verwendet um eine Wechselwirkung des Menschmodells mit der Umwelt und sich selbst herzustellen.

Für biomechanische Simulationen sind vor allem die Kenntnis bzw. Rekonstruktion der inneren Momente des Menschen bei bestimmten Bewegungsabläufen interessant. Am Beispiel einer aktiven Vorwärtssimulation eines Reckturners wird die Einsetzbarkeit des hier entwickelten Verfahrens zur Rekonstruktion von inneren Momenten demonstriert.

Weitere Ziele der biomechanischen Menschsimulation sind ausgiebige Parameter- und Sensitivitätsanalysen ohne die eine Simulation wenig Sinn macht. Dies soll am Beispiel von Fussgänger-PKW Unfallsimulationen demonstriert werden.

Inhaltsverzeichnis

Kapitel 1

Motivation und Einführung

Die Simulation menschlicher Bewegungsabläufe und die Wechselwirkung des Menschen mit der Umwelt sind schon seit vielen Jahren eine große Herausforderung im Bereich der biomechanischen Computersimulationen. Biomechanische Computersimulationen können sowohl rein passiver als auch aktiver Natur sein. Eine passive Simulation ist zum Beispiel die Mensch-PKW Unfallrekonstruktion, bei der die menschlichen Muskeln aufgrund der sehr kurzen Zeitspanne in der der Unfall abläuft, keine bewußt gesteuerten Aktivitäten durchführen. Solche Simulationen beinhalten zwar Muskelkräfte, jedoch werden diese als konstant während des gesamten Unfalls angesehen.

Aktive Simulationen beinhalten auch die Simulation von Muskeln um zum Beispiel die Bewegungsabläufe beim Gehen und Laufen zu rekonstruieren und um hieraus Kenntnis über die inneren Momente zu erlangen. Auch in den Sportwissenschaften interessieren sich seit jeher die Wissenschaftler für die Kräfte und Belastungen die im Menschen bei vielen Sportarten auftreten. Bis heute existieren nur wenige stark vereinfachende Modelle, um diese Kräfte zu bestimmen.

Im Grafikbereich ist die Rechenleistung aktueller Computer heute bereits an einem Punkt angelangt, an dem vom Computer generierte Animationen kaum noch von der Realität zu unterscheiden sind. Hingegen stößt man sehr schnell an die Grenzen der Leistungsfähigkeit heutiger Rechner, wenn es um die korrekte physikalische Simulation von selbst einfachsten mechanischen Systemen geht und dies vor allem unter dem Gesichtspunkt der Interaktivität.

1.1 Bisherige Ansätze

Bisherige Arbeiten, Verfahren und Simulationen im Bereich der biomechanischen Menschsimulation sind nicht interaktiv in der Ausführungsgeschwindigkeit, was ihren Nutzen im Hinblick auf Parameter- und Sensitivitätsanalysen erheblich einschränkt. Hierbei ist zu betonen, dass dies nur zu einem geringen Teil auf die aktuelle Computerleistung zurückzuführen ist, sondern vielmehr auf die zugrundeliegende Methode an sich. Gerade Parameteranalysen, zum Beispiel wieviel Einfluss das Gewicht oder die Größe des Menschen oder die Anfahrgeschwindigkeit des PKW auf den Simulationsausgang bei einem PKW-Unfall hat, verlangen nach interaktiven

Ausführungsgeschwindigkeiten, um einen breiten Parameterbereich abdecken zu können und den Einfluss der Modellierungsparameter zu bestimmen.

Bisherige Arbeiten und Programme zur Simulation von Menschmodellen werden oft unter Zuhilfenahme kommerzieller Programmpakete, wie zum Beispiel SimPack [1], Madymo [2] oder Adams [3] realisiert. In der Biomechanik Abteilung der TAT an der Universität Tübingen wurden bereits mehrere Dissertationen mit Hilfe dieser Programme realisiert. Oana Schüszler [4] und Valentin Keppler [5] verwendeten SimPack und Adams und erzielten hiermit sehr gute Ergebnisse. Leider erlaubten diese Simulationen keine interaktive Ausführungsgeschwindigkeit.

Mag auch ein erstes zufriedenstellendes Resultat mit diesen Programmen schnell realisierbar sein, so stößt man aber unweigerlich an die Grenzen dieser Programme: So lässt sich beim Programmpaket SimPack während der Simulation nicht einmal die Erhaltung des Impulses und Drehimpulses vernünftig überprüfen.

1.2 Zielsetzung

Ziel dieser Arbeit ist die Entwicklung eines von kommerziellen Produkten unabhängigen, völlig modular aufgebauten Verfahrens und Programms zur Simulation von Mehrkörpersystemen speziell zur Simulation von Menschmodellen und deren Interaktion mit der Umwelt. Jedoch soll dies unter dem Gesichtspunkt der interaktiven Ausführungsgeschwindigkeit erfolgen. Hierbei beschränkt sich die Arbeit keineswegs nur auf passive Bewegungsabläufe, sondern es wird am Beispiel eines Reckturners auch die Implementierung von aktiven Muskelkräften realisiert, wodurch es zum erstenmal möglich ist einen zeitlichen Verlauf aller auftretenden Kräfte und Belastungen während der Bewegung zu erhalten.

Bisherige Modelle teilen den Menschen in 15 bis 16 Einzelkörper ein und simulieren jedes einzelne Körperteil als starren Körper, teilweise mit einer einfachen Ankopplung von Schwabbelmassen, um auch den Einfluss der Weichteile in der Simulation zu berücksichtigen. Die Gelenke werden durch Zwangskräfte realisiert, d.h. die Differentialgleichungen werden unter Zwangsbedingungen integriert, die dafür sorgen, dass die Gelenke nur die Bewegungen durchführen für die sie auch zuständig sind. Die Zwangsbedingungen werden dabei als zusätzliche Gleichungen bzw. Ungleichungen dem Differentialgleichungssystem hinzugefügt. Allerdings verlangt die Lösung von Differentialgleichungen unter Zwangsbedingungen nach einer bestimmten Klasse von Lösungsverfahren, den sogenannten Differential-Algebraic-Equations Verfahren (kurz DAE).

Hierbei werden für die meisten Gelenke ideale Kugelgelenke (z.B. Hüft- oder Schultergelenk) oder Scharniergelenke (z.B. Ellenbogen- oder Kniegelenk) verwendet. Dieser Ansatz ist natürlich eine Idealisierung, da die menschlichen Gelenke alles andere als perfekte Kugel- oder Scharniergelenke sind. Dennoch resultiert diese Idealisierung bei einem weiten Bereich von Simulationen in sehr zufriedenstellenden Ergebnissen. Diese Art Menschmodelle können z.B. sehr gut die Kopfaufprallgeschwindigkeiten bei PKW-Unfallsimulationen rekonstruieren.

In dieser Arbeit wird nun ein neuer Weg zur Simulation der Gelenke verwendet, der für die biomechanische Menschsimulation wesentlich besser geeignet ist und eine größere Flexibilität in der Modellierung erlaubt. Die Gelenke werden nicht mehr durch Zwangsbedingungen in ihren Bahnen gehalten, sondern durch entsprechend starke Feder- Dämpferelemente. Auch die Gelenkanschläge und Gelenksteifigkeiten werden auf diese Weise implementiert. Für den Grenzfall hoher Kraftkonstanten entspricht dieser Ansatz einer Realisierung der Gelenke durch reine Zwangsbedingungen.

1.3 Was ist neu?

Es wird hier nun kurz zusammengefasst welche neuen Ansätze in dieser Arbeit zum Tragen kommen und was über den Stand der aktuellen Forschung hinausgeht.

Um die Geschwindigkeit von Mehrkörpersystemen für biomechanische Anwendungen in interaktive Bereiche zu bringen wurden folgende neue Ansätze und Verfahren entwickelt:

- Ein neuer Formalismus zur Aufstellung der Bewegungsgleichungen, um den Einsatz spezieller Integrationsverfahren zu ermöglichen.
- Neue Ansätze und Methoden zur Kollisionserkennung, um die Wechselwirkung des Menschen mit der Umwelt und sich selbst zu realisieren.
- Ein universelleres und erweiterbares Menschmodell mit physiologisch korrekten Gelenken.

Bei der Aufstellung der Bewegungsgleichungen wird Abstand genommen von den Standard Mehrkörpersystemen, die den Lagrangeformalismus zusammen mit Zwangsbedingungen oder generalisierten Koordinaten verwenden. Für biomechanische Simulationen ist dies in vielen Fällen zu einschränkend. Mittels eines speziellen Ansatzes mit Feder/Dämpfer Elementen und in Verbindung mit bestimmten Integrationsverfahren für Differentialgleichungen kann hierbei eine deutliche Geschwindigkeitssteigerung gegenüber Standardverfahren erreicht werden. Durch geeignete Aufstellung der Bewegungsgleichungen wird die Komplexität zur Lösung des Systems weiter reduziert. Dabei wird bei diesem Ansatz keineswegs die Genauigkeit geopfert, sondern es kann durch geeignete Parametereinstellungen das qualitativ gleiche Verhalten erreicht werden wie beim Lagrangeformalismus.

Bei der Kollisionserkennung für 3D-Modelle existiert eine riesige Anzahl an verschiedenen Verfahren, doch keines davon läßt sich direkt in eine physikalische Simulation integrieren ohne dass mit starken Geschwindigkeitseinbrüchen zu rechnen ist. Um interaktive Geschwindigkeiten zu erreichen wird ein abstandsbasierter und iterativer (und somit von der Polygonzahl nahezu unabhängiger) Kollisionsalgorithmus entwickelt, der neben dem Punktpaar mit der kürzesten Verbindung auch Nachbarpunkte berücksichtigt. Gerade dies ist ein Schlüsselpunkt zur Verbesserung der Geschwindigkeit (und Genauigkeit!) und wurde bisher bei Simulationen kaum berücksichtigt. Dies ist jedoch bei hohen bis moderaten Genauigkeitsanforderungen

unumgänglich, da sonst Fälle eintreten können in denen das Integrationsverfahren deutlich länger rechnet.

Ein realistisches Menschmodell zu implementieren ist eine weitere Herausforderung. Hier werden gerade im Bereich der Gelenke neue Methoden implementiert um die resultierenden Bewegungen realistischer aussehen zu lassen.

Besonders wird dabei auf die enge Verknüpfung der 3D-Grafik mit der physikalischen Simulation Wert gelegt.

1.4 Überblick zur vorliegenden Arbeit

In den ersten Kapiteln wird das Grundgerüst entwickelt, um überhaupt eine biomechanische Simulation in interaktiver Ausführungsgeschwindigkeit realisieren zu können. Besondere Aufmerksamkeit wird deshalb den Bewegungsgleichungen und deren Integration sowie der Kollisionserkennung gewidmet.

Aufgrund der Realisierung der Gelenke durch Feder/Dämpfer Elemente erhält man zwangsläufig steife Bewegungsgleichungen, die nur mit einer bestimmten Klasse von Integrationsmethoden vernünftig gelöst werden können. In Kapitel 2 wird deshalb ein Überblick über die wichtigsten Integrationsverfahren gegeben und näher auf ihre Vor- und Nachteile eingegangen.

In Kapitel 3 werden die Bewegungsgleichungen aufgestellt und Verfahren aufgeführt, die ihre Integration beschleunigen. Dies geschieht allerdings noch völlig allgemein und wird erst in späteren Kapiteln für spezielle biomechanische Anwendungsfälle konkretisiert.

In Kapitel 4 wird ein Algorithmus entwickelt, mit dem man direkt aus den 3D Dreiecksflächen die Masseeigenschaften, also Schwerpunkt und Trägheitstensor eines Körpers, gewinnen kann. Auf diese Weise lassen sich alle relevanten Problemparameter sehr schnell bestimmen ohne dass zunächst *per hand* die Parameter dem 3D Modell angepasst werden müssen.

Kapitel 5 entwickelt und beschreibt einen iterativen und von der Polygonzahl nahezu unabhängigen Algorithmus zur Abstandsbestimmung von 3D Körpern zueinander. Dabei wird besonderes auf die für physikalische Simulationen wichtigen Zusätze eingegangen um die Kollisionserkennung in einer interaktiven Simulation einsetzen zu können.

Da in dieser Arbeit die Betonung auf biomechanischer Menschsimulation liegt, wird in Kapitel 6 die Implementierung eines Menschmodells beschrieben, mit dem die Anwendungsfälle dieser Arbeit berechnet wurden. Dieses Menschmodell lehnt sich dabei eng an schon existierende Modelle an, damit ein direkter Vergleich möglich ist. Es werden aber auch wichtige Neuerungen im Bereich der Gelenkanschläge und der einfache Muskelsimulation vorgestellt.

In den letzten beiden Kapiteln 7 und 8 schließlich werden konkrete Anwendungsfälle simuliert um das Einsatzspektrum des Verfahrens zu demonstrieren. Zunächst wird die Vorwärtssimulation eines Reckturners durchgeführt um die für diese Bewegungen typischen Momente

in den Gelenken zu rekonstruieren. Hierbei wurden jeweils komplette Sequenzen vom Aufschwingen bis hin zur Riesenfelge simuliert und die Ergebnisse mit realen Bewegungsabläufen verglichen. Die Simulationsgeschwindigkeit war hierbei voll interaktiv, so dass der Anwender aktiv und online die Simulation beeinflußen konnte. Auf sämtliche Daten wie Gesamtimpuls/Gesamtdrehimpuls, Schwerpunktpositionen und Geschwindigkeiten sowie Belastungen und Kräfte in den Gelenken hat man dabei Zugang. Diese Daten können nach oder während der Simulation zur Analyse herangezogen werden. Die folgenden drei Bilder zeigen Auszüge aus einer Simulation des Reckturners.

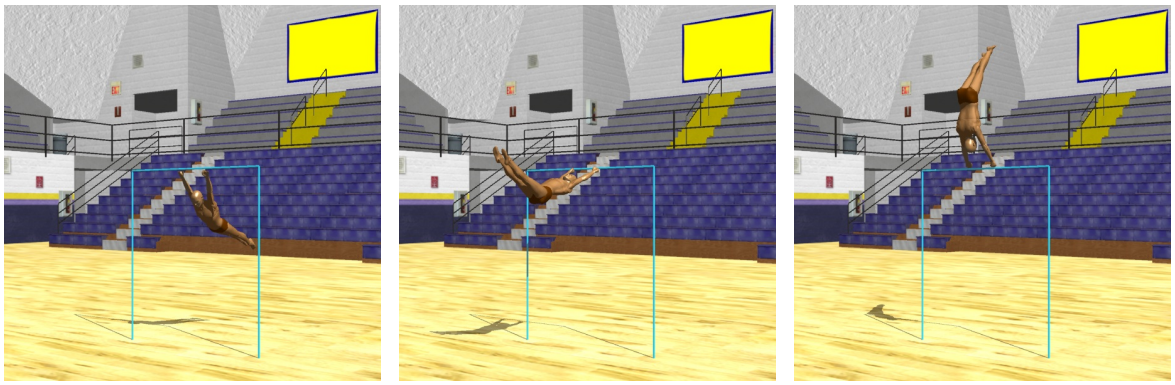


Abbildung 1.1: Vorwärtssimulation eines Reckturners

Der zweite Anwendungsfall simuliert einen Fußgänger-PKW Unfall. Konkret wurde hier untersucht wie die Kopfaufprallgeschwindigkeit (die Haupttodesursache bei Fußgänger-PKW Unfällen!) von diversen Parametern wie Anfahrgeschwindigkeit und Anfahrrichtung sowie der Anfangsstellung der Person abhängt. Als PKW wurden ein VW-Golf IV sowie eine Mercedes A-Klasse verwendet, um auch den Einfluß der unterschiedlichen PKW-Geometrien auf den Simulationsausgang zu untersuchen.

Abbildung 1.2: Fußgänger-PKW Unfallsimulation

Als Programmiersprache kam C++ zum Einsatz, die zunehmend auch in naturwissenschaftlichen Simulationen ihre Anwendung findet. Aufgrund der objektorientierten Auslegung von C++ lassen sich wissenschaftliche Anwendungen sehr schön realisieren, ohne dass das Programm unleserlich wird. Durch geeignet programmierte Matrix- und Vektorklassen liest sich der Quelltext nahezu so einfach wie ein Buch.

Die in dieser Arbeit verwendeten Szenen und Modelle wurden alle mit dem Programm 3D Studio Max von Kinetix erstellt. 3D Studio MAX bietet leistungsfähige Funktionen an, um sehr komplexe 3D Modelle zu generieren. Alle Objekte werden als Dreiecksmodelle erstellt, die mit einem selbst geschriebenen Plugin (einem Zusatzprogramm zu 3D Studio Max) in ein einfaches Format exportiert wurden. Durch virtuelle Objekte (also 3D Flächen, die später in der Simulation nicht dargestellt werden) können dann auch schon in 3D Studio Max die Positionen der Gelenke des Menschmodells festgelegt werden. In Abbildung 1.3 sieht man das 3D-Modell des Menschen im Gittermodus.

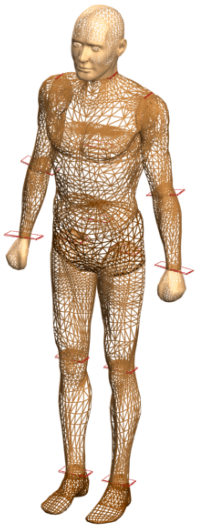


Abbildung 1.3: Festlegung der Gelenkdrehpunkte im Programm 3D-Studio-Max

Die flachen Boxen an den geweiligen Gelenken werden beim Start des Simulationsprogramms eingelesen und aus der Position und Orientierung dieser Boxen werden automatisch die Gelenkpositionen und die Gelenkachsen berechnet. Dieser Ansatz erspart die oft mühselige Arbeit eine Simulationsszene durch verwirrende Winkel- und Positionsangaben in Textdateien in ihre gewünschte Ausgangssituation zu bringen. Die Anzahl der verwendeten Dreiecke spielt hierbei keine entscheidende Rolle. Es können also sehr hochaufgelöste Modelle verwendet werden, ohne die Ausführungsgeschwindigkeit merklich zu reduzieren. Die in dieser Arbeit verwendeten Modelle haben typischerweise 20000 bis 100000 Dreiecke. Es können somit in der Simulation eines Mensch-PKW Unfalls sehr hoch aufgelöste Automodelle verwendet werden, die von den realen Modellen optisch so gut wie nicht mehr zu unterscheiden sind. Die Verwendung von 3D Studio MAX ist jedoch nicht zwingend notwendig, sondern wurde in dieser Arbeit nur verwendet weil der Autor schon über entsprechende Erfahrungen mit diesem Programm verfügte.

Kapitel 2

Integrationsverfahren

Das Gebiet der numerischen Lösung von Differentialgleichungen ist sehr komplex und es existieren eine Fülle von unterschiedlichen Verfahren und Algorithmen. Die Wahl eines geeigneten Verfahrens für ein bestimmtes Problem verlangt oftmals einiges an Aufwand und viel *Probieren*, denn jedes Verfahren hat seine Vor- und Nachteile, die sehr problemspezifisch sind.

In den letzten 20 Jahren ist auf dem Gebiet der numerischen Integration von Differentialgleichungen einiges an theoretischer Arbeit geleistet worden, und die Qualität der unterschiedlichen Verfahren ist und wird stetig weiter verbessert. So ist vor ein paar Jahren noch der Einsatz impliziter Integrationsverfahren für interaktive Simulationen völlig undenkbar gewesen. Erst mit dem Erscheinen von Programmen wie zum Beispiel RADAU5 [9] und VODE[10]/CVODE[11] stieg die Praktikabilität der impliziten Verfahren enorm an. Im Kapitel 2.2.4 wird genauer auf diese beiden Verfahren eingegangen.

Es wird nun ein kurzer Überblick über die wichtigsten Integrationsverfahren gegeben und beschrieben welche Klassen von Integrationsverfahren für die hier vorliegende Simulation verwendet werden und wie sich diese Verfahren beschleunigen lassen. Ein System von Differentialgleichungen sei allgemein gegeben durch

$$\dot{\boldsymbol{Y}}(t) = \boldsymbol{F}(t, \boldsymbol{Y}(t)), \qquad \boldsymbol{Y}, \dot{\boldsymbol{Y}}, \boldsymbol{F} \in \mathbb{R}^N \tag{2.1}$$

Hierbei ist N die Dimension des Systems. Differentialgleichungen 2. Ordnung, wie sie zum Beispiel bei mechanischen Systemen auftreten, lassen sich natürlich auch in der Form (2.1) schreiben.

Zur Simulation eines einzelnen punktförmigen Körpers in 3 Dimensionen ohne Einschränkung der Freiheitsgrade wäre $N = 6$, denn die 3 Freiheitsgrade für die Translation führen auf insgesamt 6 Gleichungen: 3 für die Änderung der Geschwindigkeit ($\dot{\boldsymbol{v}} = \boldsymbol{a} = \boldsymbol{F}/M$) und 3 für die Änderung der Position ($\dot{\boldsymbol{r}} = \boldsymbol{v}$).

Eine geschlossene Darstellung der Funktion $\boldsymbol{F}(t, \boldsymbol{Y}(\boldsymbol{t}))$ ist für die biomechanische Simulation leider nicht möglich. Man sieht dies zum Beispiel an der Kollision, da erst ab einem gewissen

Abstand zweier Körper zueinander eine Wechselwirkung stattfindet und auch erst dann diese Wechselwirkung in der Funktion $\boldsymbol{F}(t, \boldsymbol{Y}(t))$ erscheint.

Integrationsverfahren lassen sich zwei großen Klassen zuordnen, den expliziten sowie den impliziten Verfahren. Es wird an einem einfachen Beispiel gezeigt wann diese Verfahren zum Einsatz kommen.

Aufgrund der Realisierung der Körpergelenke durch Feder/Dämpfer Elemente erhält man zwangsläufig steife Bewegungsgleichungen. Ein einfaches mechanisches System, das auf steife Differentialgleichungen führt, ist zum Beispiel das Stabpendel der Masse M mit konstanter Länge L, das mit einer harten, masselosen Feder der Kraftkonstanten K an einer Decke befestigt ist.

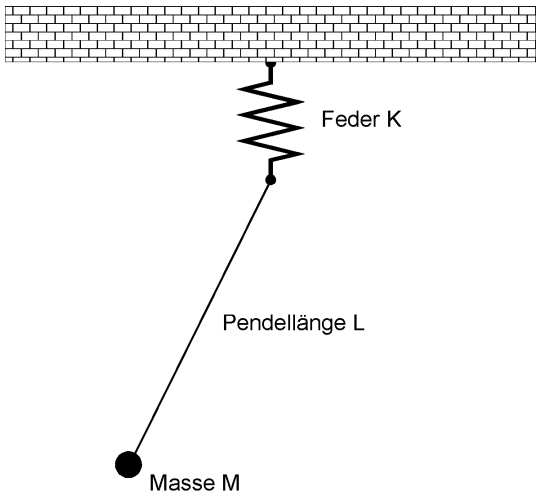


Abbildung 2.1: Einfaches Beispiel für ein steifes mechanisches System

Die Schwingungsdauern des Pendels und der Feder betragen näherungsweise [1]

$$\begin{aligned} T_{\text{Pendel}} &= 2\pi \left(1 + \tfrac{1}{4}\sin^2 \tfrac{\alpha_0}{2}\right) \sqrt{\tfrac{L}{g}} \\ T_{\text{Feder}} &= 2\pi \sqrt{\tfrac{M}{K}} \end{aligned}$$

Hierbei ist α_0 der Startwinkel des Pendels aus der vertikalen Richtung. Im Grenzfall einer sehr hohen Federkonstanten simuliert dieses System ein ideales mathematisches Pendel. Ein steifes System erhält man, wenn die Federkonstante derart gewählt wird, dass gilt $T_{\text{Pendel}} \gg T_{\text{Feder}}$, das heißt das System besteht aus 2 verschieden schnell oszillierenden Teilen, die also auch erheblich unterschiedliche Eigenwerte aufweisen.

Mit der Wahl für $L = 1m$, $M = 1.0kg$ und $K = 10000N/m$ ist das Verhältnis der beiden Schwingungsdauern $T_{\text{Pendel}} : T_{\text{Feder}} \approx 30$.

[1] Die exakte Schwingungsdauer des Gesamtsystems ist natürlich komplizierter, spielt aber zur Verdeutlichung des Sachverhalts keine Rolle.

Wird dieses System nun mit expliziten Integrationsverfahren gelöst, so muß aufgrund der beschränkten Stabilität (siehe Kapitel 2.2.4) eine Schrittweite verwendet werden, die in der Größenordnung der Schwingungsdauer der Feder liegt. Bei einer größeren Schrittweite würde das explizite Verfahren also schlichtweg falsche Ergebnisse liefern und wäre instabil. Je härter die Federkonstante, desto länger benötigen diese Verfahren. Für sehr hohe Federkonstanten K sind diese Verfahren daher völlig unpraktikabel, da die Schrittweite Größenordnungen verlangt, die von der Fließkommagenauigkeit begrenzt werden.

Implizite Verfahren unterliegen dieser Beschränkung aufgrund des größeren Stabilitätsgebiets nicht, allerdings benötigen sie einen erheblich größeren Rechenaufwand, denn zu jedem Zeitschritt muß (meistens) ein nichtlineares Gleichungssystem gelöst werden. Aber schon am Beispiel des einfachen Stabpendels, mit der *moderat* gewählten Federkonstanten $K = 10000N/m$, sind die expliziten Verfahren erheblich langsamer. Implizite Verfahren besitzen die Eigenschaft, diese hochfrequenten Schwingungen zu dämpfen.

Mit Hilfe von MATLAB [6] wurde ein Programm geschrieben, das das Stabpendel simuliert und mit dem gezeigt werden soll wie unterschiedlich die Laufzeiten der expliziten und impliziten Methoden in Abhängigkeit von der Federkonstanten K bei diesem Problem sind. Ein Listing des Programms ist im Anhang A zu sehen. Integriert wurde das System über einen Zeitraum von 4 Sekunden. Zur Integration wurden die MATLAB eigenen Routinen ODE45 (explizite Runge-Kutta Verfahren) und ODE15s (implizite BDF-Verfahren variabler Ordnung) sowie eine Matlab Version von RADAU5 verwendet. Eine genaue Beschreibung der Routinen ODE45 und ODE15s findet sich in der Veröffentlichung von Lawrence F. Shampine und Mark W. Reichelt [8]. Auf das RADAU5 Verfahren wird in diesem Kapitel noch genauer eingegangen. In der folgenden Tabelle ist die Anzahl der Funktionsaufrufe in Abhängigkeit von der Kraftkonstanten K aufgelistet. In Klammern ist hinter der Anzahl der Funktionsaufrufe noch die gesamte Laufzeit angegeben.

Kraftkonstante $K[N/m]$	RADAU5	ODE15s	ODE45
10	35 (0.11s)	106 (0.06s)	970 (1.02s)
100	79 (0.19s)	169 (0.08s)	2635 (3.22s)
1000	226 (0.39s)	435 (0.19s)	9437 (20.95s)
10000	532 (0.98s)	1085 (0.53s)	35201 (211.03s)
100000	992 (1.77s)	3304 (1.66s)	-

Während sich bei den impliziten Verfahren (RADAU5, ODE15s) die Laufdauer bei einer Verzehnfachung von K nur verdoppelt, steigt der Zeitaufwand bei den expliziten Verfahren (ODE45) direkt proportional zu K. Bei Kraftkonstanten von $K = 10000N/m$ benötigt ODE45 nahezu 200 mal länger zur Integration als RADAU5. Interessant ist ferner anzumerken, dass RADAU5 mit steigendem K immer effektiver wird. Der Grund hierfür wird in Kapitel 2.2.4 näher erläutert.

Dennoch soll hier ein Überblick über die zwei wichtigsten Klassen von expliziten Verfahren gegeben werden, denn moderne Integrationsverfahren (zum Beispiel VODE/CVODE) können durch eine automatische Steifigkeitserkennung zwischen expliziten und impliziten Verfahren hin- und herschalten und hierdurch die Integration beschleunigen.

2.1 Explizite Verfahren

Explizite Integrationsverfahren werden unterteilt in Einschritt- und Mehrschrittverfahren. Explizite Verfahren haben den Vorteil, dass sie verhältnismäßig einfach zu implementieren sind und $O(N)$ Zeitaufwand benötigen, wobei N die Anzahl der Gleichungen des Differentialgleichungssystems ist.

2.1.1 Einschrittverfahren

Die bekanntesten Vertreter der expliziten Einschrittverfahren sind die Runge-Kutta-Verfahren. Sie haben die Gestalt

$$\begin{aligned} \boldsymbol{Y}(t+h) &= \boldsymbol{Y}(t) + h\sum_{i=1}^{s} b_i \boldsymbol{F}(t, \boldsymbol{Y}^{(i)}(t)) \\ \boldsymbol{Y}^{(i)}(t) &= \boldsymbol{Y}(t) + h\sum_{j=1}^{i-1} a_{ij} \boldsymbol{F}(t, \boldsymbol{Y}^{(j)}(t)) \end{aligned}$$

Der neue Funktionswert $\boldsymbol{Y}(t+h)$ wird also aus Linearkombinationen der Funktion $\boldsymbol{F}(t, \boldsymbol{Y}(t))$ eines einzigen vorherigen Zeitschritts bestimmt. Die Koeffizienten werden dabei so gewählt, dass das Verfahren eine möglichst hohe Fehlerordnung besitzt (siehe hierzu [12]). Die einfachste Form der expliziten Runge-Kutta-Verfahren ist das Euler-Verfahren

$$\boldsymbol{Y}(t+h) = \boldsymbol{Y}(t) + h\dot{\boldsymbol{Y}}(t) = \boldsymbol{Y}(t) + h\boldsymbol{F}(t, \boldsymbol{Y}(t))$$

Der neue Funktionswert $\boldsymbol{Y}(t+h)$ an der Stelle $t+h$ wird einzig durch den Funktionswert an der Stelle t und der ersten Ableitung $\dot{\boldsymbol{Y}}(t)$ bestimmt.

2.1.2 Mehrschrittverfahren

Mehrschrittverfahren verwenden auch die Ergebnisse mehrerer vorheriger Funktionswerte um eine Lösung zu bestimmen. Ein allgemeines Mehrschrittverfahren ist gegeben durch

$$\sum_{l=0}^{s} \alpha_l \boldsymbol{Y}(t + \frac{l}{s}h) = h\sum_{l=0}^{s-1} \beta_l \boldsymbol{F}(t + \frac{l}{s}h, \boldsymbol{Y}(t + \frac{l}{s}h)) \tag{2.2}$$

mit

$$\alpha_l, \beta_l \in \mathbb{R} \quad \text{und} \quad \alpha_s \neq 0, \quad |\alpha_0| + |\beta_0| \neq 0$$

Hierbei wird die Funktion $\boldsymbol{Y}(t+h)$ an der Stelle $t+h$ angenähert durch die s Funktionswerte von vorherigen Zeitpunkten. Um ein Mehrschrittverfahren zu starten, müssen natürlich erst einmal die s zurückliegenden Funktionswerte bekannt sein. Dies geschieht meist durch ein Runge-Kutta-Einschrittverfahren gleicher Ordnung.

2.2 Implizite Verfahren

Auch implizite Verfahren werden in Ein- und Mehrschrittverfahren unterteilt. Sie unterscheiden sich von expliziten Verfahren dadurch, dass sie sich nicht explizit nach den zu berechnenden Werten $\boldsymbol{Y}(t+h)$ auflösen lassen und im allgemeinen nichtlineare Gleichungssysteme gelöst werden müssen, also einen erheblichen rechnerischen Mehraufwand benötigen.

Die Herleitung und Implementierung von impliziten Verfahren ist sehr aufwändig. Um die Grundidee zur Implementierung von impliziten Verfahren aufzuzeigen, wird das einfachste implizite Verfahren, das implizite Euler Verfahren, beschrieben. Die komplizierteren Verfahren höherer Ordnung basieren alle auf einem ähnlichen Schema.

2.2.1 Runge-Kutta-Verfahren

Die impliziten Runge-Kutta-Verfahren sind wie die expliziten Runge-Kutta-Verfahren, Einschrittverfahren. Zur Bestimmung des neuen Funktionswerts $\boldsymbol{Y}(t+h)$ reicht also die Kenntnis eines einzigen vorherigen Funktionswerts aus. Ein allgemeines Verfahren mit s Stufen hat die Struktur

$$\begin{aligned} \boldsymbol{Y}(t+h) &= \boldsymbol{Y}(t) + h\sum_{i=1}^{s} b_i \boldsymbol{F}(t+c_i h, \boldsymbol{Y}^{(i)}) \\ \boldsymbol{Y}^{(i)} &= \boldsymbol{Y}(t) + h\sum_{j=1}^{s} a_{ij} \boldsymbol{F}(t+c_j h, \boldsymbol{Y}^{(j)}) \end{aligned} \tag{2.3}$$

Es werden nun auch die unbekannten Funktionswerte $\boldsymbol{F}(t+c_j h, \boldsymbol{Y}^{(j)})$ zur Bestimmung der Lösung verwendet.

2.2.2 Das BDF-Verfahren

Im Gegensatz zu den expliziten Mehrschrittverfahren (2.2) verwenden die impliziten Mehrschrittverfahren auch den (noch) unbekannten Funktionswert $\boldsymbol{F}(t+h, \boldsymbol{Y}(t+h))$ zum Zeitpunkt $t+h$. Der bekannteste Vertreter dieser Verfahren ist das BDF-Verfahren (Backward-Differentiation-Formula)

$$\sum_{l=0}^{s} \alpha_l \boldsymbol{Y}(t+\frac{l}{s}h) = h\beta_s \boldsymbol{F}(t+h, \boldsymbol{Y}(t+h)) \tag{2.4}$$

Der Name Rückwärtsdifferentiationsmethode erklärt sich dadurch, dass die linke Seite von (2.4) das h-fache einer Formel zur numerischen Differentiation der ersten Ableitung von $\boldsymbol{F}$ darstellt.

2.2.3 Implementierung

Alle impliziten Verfahren, gleich ob die Runge-Kutta-Einschrittverfahren oder die BDF-Mehrschrittverfahren, werden nach einem ähnlichen Schema implementiert und müssen stets ein nichtlineares Gleichungssystem lösen [2]. Die Komplexität dieses Gleichungssystems und der Aufwand zur Lösung hängt entscheidend von einer geeigneten Aufstellung der Bewegungsgleichungen ab. Am Beispiel des impliziten Euler-Verfahrens wird der Grundgedanke zur Implementierung dieser Verfahren aufgezeigt. Das einfachste implizite Verfahren ist das implizite Euler-Verfahren, das für $s = 1$ aus Gleichung (2.3) folgt. Es ist somit das einfachste implizite Runge-Kutta-Verfahren.

$$\boldsymbol{Y}(t+h) = \boldsymbol{Y}(t) + h\boldsymbol{F}(t+h, \boldsymbol{Y}(t+h)) \tag{2.5}$$

Im Gegensatz zum expliziten Euler Verfahren enthält diese Gleichung nun den unbekannten Funktionswert $\boldsymbol{F}(t+h, \boldsymbol{Y}(t+h))$. In den wenigsten Fällen kann man die Gleichung (2.5) direkt nach $\boldsymbol{Y}(t+h)$ auflösen. Es muß also ein nichtlineares Gleichungssystem gelöst werden, was üblicherweise durch die Newton-Iteration realisiert wird. Für das Newtonverfahren bringt man Gleichung (2.5) in die Form

$$\boldsymbol{Y}(t+h) - \boldsymbol{Y}(t) - h\boldsymbol{F}(t+h, \boldsymbol{Y}(t+h)) = 0$$

Die gesamte linke Seite fasst man zusammen zu

$$\boldsymbol{G}(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{Y}(t) - h\boldsymbol{F}(t+h, \boldsymbol{x}), \quad \text{mit} \quad \boldsymbol{x} = \boldsymbol{Y}(t+h)$$

$\boldsymbol{x} = \boldsymbol{Y}(t+h)$ ist der gesuchte Lösungsvektor. Um $\boldsymbol{x}$ zu finden, wird $\boldsymbol{G}(\boldsymbol{x})$ in eine Taylorreihe entwickelt und nach dem zweiten Glied abgebrochen, d.h. Terme der Ordnung $O(\boldsymbol{\Delta x^2})$ werden ignoriert

$$G_i(\boldsymbol{x} + \boldsymbol{\Delta x}) = G_i(\boldsymbol{x}) + \sum_{j=1}^{N} \frac{\partial G_i}{\partial x_j} \Delta x_j$$

und in Matrixschreibweise

$$\boldsymbol{G}(\boldsymbol{x} + \boldsymbol{\Delta x}) = \boldsymbol{G}(\boldsymbol{x}) + \boldsymbol{J}(\boldsymbol{x}) \cdot \Delta \boldsymbol{x}, \quad J_{ij} \equiv \frac{\partial G_i}{\partial x_j} \tag{2.6}$$

Hierbei ist $\boldsymbol{J}$ die Jacobi Matrix. Die Jacobi Matrix lässt sich, wenn $\boldsymbol{F}$ nicht geschlossen darstellbar ist, nicht analytisch berechnen. Sie wird dann mittels Differenzenquotienten approximiert.

$$J_{ij} = \frac{G_j(\boldsymbol{x} + \delta_i) - G_j(\boldsymbol{x})}{\delta_i} \tag{2.7}$$

[2] Eine Ausnahme bilden die Rosenbrock Methoden, wie zum Beispiel RODAS von Hairer und Wanner, allerdings ist dies mit einer erheblich geringeren Stabilität und Fehlerordnung verbunden.

Hierbei bedeutet $\boldsymbol{x}+\delta_i$, dass δ_i nur zur i-ten Komponente x_i addiert wird. Die Wahl eines geeigneten δ_i zur Bestimmung der Differenzenquotienten ist eine delikate Sache und eng mit der Rechnergenauigkeit, der relativen Fehlerschranke und dem absoluten Wert der Komponente von x_i verbunden. Die meisten Verfahren verwenden hierzu die Abschätzung

$$\delta_i = \sqrt{EPS \cdot MAX(1e^{-5}, |x_i|)} \tag{2.8}$$

Hierbei ist EPS die kleinste Zahl für die gilt $1 + EPS > 1$. Für DOUBLE Genauigkeit ist dies ungefähr $1e^{-16}$ (siehe Anmerkung [3]).

Ziel ist es nun einen Vektor $\boldsymbol{x} + \boldsymbol{\Delta x}$ zu finden, so dass die linke Seite von (2.6) verschwindet, also

$$\boldsymbol{G}(\boldsymbol{x} + \boldsymbol{\Delta x}) = 0 \quad \Rightarrow \quad \boldsymbol{J}(\boldsymbol{x}) \cdot \Delta \boldsymbol{x} = \boldsymbol{G}(\boldsymbol{x}) \tag{2.9}$$

Diese Gleichung kann je nach Struktur von $\boldsymbol{J}$ mit verschiedenen Verfahren gelöst werden. Wenn $\boldsymbol{J}$ keine spezielle Bandstruktur oder Dreiecksform hat wird häufig das Gauss-Verfahren verwendet. Die so erhaltenen Vektoren $\boldsymbol{\Delta x}$ führen zu einem neuen Lösungsvektor

$$\boldsymbol{x}_{\text{Neu}} = \boldsymbol{x}_{\text{Alt}} + \Delta \boldsymbol{x}$$

den man wiederum in Gleichung (2.9) einsetzt. Typischerweise werden zwischen 7 bis 10 Iterationen nach dem Newton Verfahren durchgeführt bis ein *brauchbarer* Lösungsvektor vorliegt.

Der Hauptaufwand von impliziten Verfahren gegenüber den expliziten Verfahren liegt also darin das nicht lineare Gleichungssystem (2.3) zu lösen.

Allgemeine implizite s-stufige Runge-Kutta-Verfahren müssen ein Gleichungssystem der Dimension $s \cdot N$ lösen, wodurch der Rechenaufwand enorm ansteigt. Eine spezielle Struktur von $\boldsymbol{J}$ würde also in erheblichen Geschwindigkeitssteigerungen resultieren, hierauf wird im Kapitel 3 näher eingegangen. Die BDF-Verfahren hingegen müssen nur ein Gleichungssystem der Dimension N lösen, sind also meist etwas schneller.

Speziell bei den impliziten s-stufigen Runge-Kutta-Verfahren können jedoch einige Transformationen durchgeführt werden, die den Aufwand zur Lösung des Gleichungssystems erheblich reduzieren. RADAU5, ein 3-stufiges Verfahren der Ordnung 5, reduziert auf diese Weise den Aufwand zur Lösung des Gleichungssystems um den Faktor 5. Obwohl die impliziten Runge-Kutta-Verfahren sehr aufwändig sind, sind sie die erste Wahl wenn es um eine hohe Fehlerordnung verbunden mit einer großen Stabilität - und dies auch für nichtlineare Differentialgleichungen -, geht.

[3] Der genaue Wert schwankt je nach verwendetem Prozessor und Compiler, da zum Beispiel i86 kompatible Prozessoren wahlweise mit 64 oder 80 Bit Genauigkeit rechnen können.

2.2.4 RADAU5 und VODE/CVODE

Die in den vorherigen Kapiteln vorgestellten impliziten Verfahren zu implementieren ist sehr aufwändig. Glücklicherweise existieren mittlerweile eine Vielzahl an frei erhältlichen und sogar im Quellcode vorliegenden Verfahren, die bereits seit vielen Jahren für verschiedene Anwendungen im Einsatz sind und sich dort auch bewährt haben.

Obwohl es eine Fülle an Programmen gibt um steife Differentialgleichungen zu lösen, kamen für diese Arbeit nur die Programme RADAU5 [9] und VODE [10]/CVODE [11] zum Einsatz. Zwar wurden auch andere Verfahren getestet, jedoch lieferten diese nicht annäherend die Leistung und Genauigkeit die RADAU5 und VODE/CVODE erreichten. Die simultane Verwendung dieser beiden Programme, die auf unterschiedlichen Verfahren basieren, bietet zudem auch eine gute Kontrolle hinsichtlich eventueller Fehler in diesen Programmen. Wie die Simulationsläufe gezeigt haben liefern jedoch beide Verfahren die gleichen Ergebnisse.

RADAU5 ist von Ernst Hairer und Gerhard Wanner an der Universität Genf entwickelt worden und liegt als gut dokumentierter Quellcode in Fortran vor. Ihr Buch, Solving Ordinary Differential Equations II [9], beschreibt recht ausführlich die Implementierung von RADAU5 und beschreibt auch Verfahren, mit denen das resultierende Gleichungssystem wesentlich schneller gelöst werden kann. RADAU5 ist das zur Zeit wahrscheinlich genaueste und stabilste Integrationsverfahren zur Simulation steifer Differentialgleichungen und ist für Anwendungen bei denen es auf hohe Rechengenauigkeit ankommt die erste Wahl. Von den gleichen Autoren existiert auch das RADAU Verfahren welches die impliziten Runge-Kutta Verfahren mit variabler Ordnung implementiert. Hierbei ist die maximale Ordnung 13. Für niedrige Fehlertoleranzen ($AbsTol > 1e-5$) ist dies kaum von Vorteil und resultiert vor allem in einer deutlich höheren Integrationszeit. Bei Anwendungen mit extrem hohen Genauigkeitsanforderungen ($AbsTol < 1e-6$) ist diese hohe Ordnung jedoch wesentlich effektiver obwohl das zugrunde liegende Gleichungssystem erheblich größer ist.

VODE und der Nachfolger CVODE wurde von mehreren Personen am Lawrence Livermore National Laboratory entwickelt. Die Hauptautoren sind George D. Byrne, P. N. Brown, Alan C. Hindmarsh und Scott D.Cohen. Die Fortran-Version VODE ist mittlerweile durch die C-Version CVODE ersetzt worden. CVODE wird stetig weiterentwickelt und bietet mittlerweile auch eine integrierte Sensitivitätsanalyse. Weiterhin existiert auch eine Variante PVODE, die speziell für Parallelrechner entwickelt wurde. Alle Versionen liegen als Quellcode vor und enthalten eine ausführliche Dokumentation und Beispielprogramme.

Der grundlegende Unterschied zwischen RADAU5 und CVODE liegt in der verwendeten Art des impliziten Verfahrens. RADAU5 basiert auf dem 3-stufigen impliziten Runge-Kutta-Verfahren der Ordnung $s = 5$, ist also ein Einschrittverfahren. CVODE hingegen verwendet das BDF-Verfahren mit variablen Ordnungen von 1 bis 5 und ist ein Mehrschrittverfahren.

Ein entscheidender Vorteil der impliziten Runge-Kutta Verfahren ist das grössere Stabilitätsgebiet im Vergleich zu den impliziten Mehrschrittverfahren gleicher Ordnung, also auch mit

vergleichbarem Diskretisierungsfehler. Zur Veranschaulichung dient hierzu die lineare Anfangswertaufgabe [4]

$$y'(t) = \lambda y(t), \qquad \lambda \in \mathbb{C} \tag{2.10}$$

mit der bekannten Lösung

$$y(t) = e^{\lambda t} \tag{2.11}$$

Für negative Realteile von λ, also $Re(\lambda) < 0$, handelt es sich um eine gedämpfte Schwingung, also gilt

$$\lim_{t\to\infty} |y(t)| = 0 \tag{2.12}$$

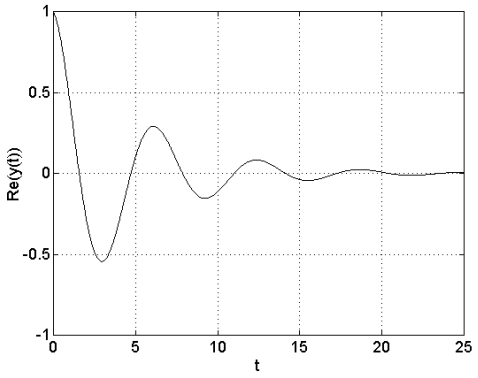


Abbildung 2.2: Gedämpfte Schwingung mit $\lambda = -0.2 + 1 * i$

Besitzt ein Verfahren diese Eigenschaft für alle λ mit $Re(\lambda) < 0$ nicht, so kann es instabil werden. Allgemein führt ein numerisches Verfahren, angewendet auf die Anfangswertaufgabe (2.10), auf

$$y(t + h) = R(h\lambda) \cdot y(t) \tag{2.13}$$

Hierbei ist $h \in \mathbb{R}$ die Schrittweite des Verfahrens und $R(h\lambda)$ ist die Verfahrensfunktion. Ein Verfahren nennt man nun genau dann A-Stabil, wenn gilt

$$|R(h\lambda)| \leq 1, \ \forall\, h\lambda \text{ mit } Re(h\lambda) \leq 0 \tag{2.14}$$

Es gibt noch schwächere Bedingungen als die A-Stabilität, zum Beispiel die A(α)-Stabilität, allerdings können diese Verfahren in gewissen Fällen instabil werden. Im Fall der linearen

[4] Auch für nichtlineare Differentialgleichungen lässt sich zeigen, dass die impliziten Runge-Kutta-Verfahren im Vergleich zu den Mehrschrittverfahren gleicher Ordnung absolut stabil sind (L- und B-Stabilität). Ein Beweis findet sich in [9]. Allerdings sind diese Beweise nicht so anschaulich wie bei der linearen Anfangswertaufgabe. Für Mehrschrittverfahren existiert bis heute keine vernünftige Variante, die bei hoher Ordnung absolut stabil ist.

Anfangswertaufgabe (2.10) lassen sich diese Gebiete für RADAU5 und CVODE analytisch berechnen und sind in den folgenen Grafiken dargestellt. In den Grafiken steht μ für $h\lambda$.

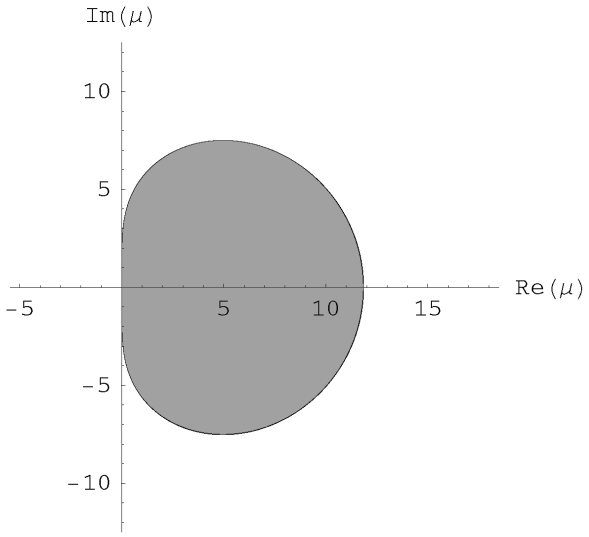


Abbildung 2.3: RADAU5

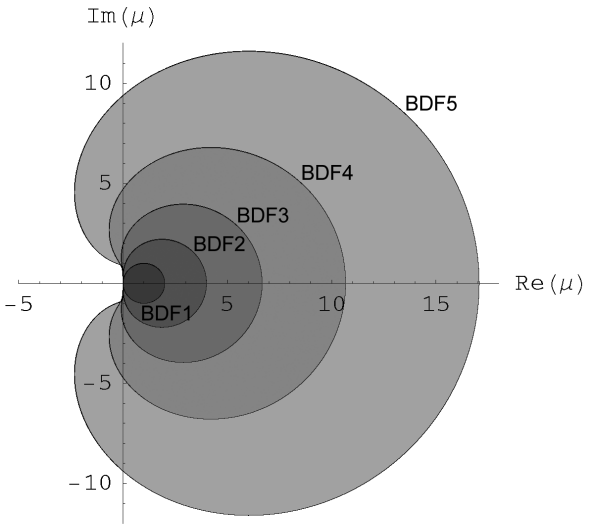


Abbildung 2.4: BDF Ordnung 1 bis 5

Die grau schattierten Flächen sind die Gebiete, in denen das jeweilige Verfahren nicht mehr stabil ist hinsichtlich der linearen Anfangswertaufgabe (2.10). Liegt also das Produkt Schrittweite h mal λ in diesen grau schattierten Flächen, so liefert das Verfahren keine sinnvollen Ergebnisse mehr.

Wie man sieht ist RADAU5 für beliebige komplexe $h\lambda$ mit negativem Realteil $Re(h\lambda) \leq 0$ absolut stabil, welche Schrittweite auch immer gewählt wird. Anders sieht es bei den BDF-Mehrschrittverfahren aus. Wie man der Abbildung entnehmen kann sind nur die Verfahren bis zur Ordnung $s = 2$ absolut stabil. Für $s = 3, 4$ und 5 reichen aber Teile des Instabilitätsgebiets in die linke Hälfte der komplexen Ebene, das heißt je nach Schrittweite h kann das Verfahren instabil werden. Dies ist eine starke Einschränkung im Hinblick auf ungedämpfte oder nur schwach gedämpfte Systeme, die sehr dicht an der imaginären Achse liegen, und kann in diesen Fällen zu unsinnigen Ergebnissen führen. Bei solchen Systemen können also nur die BDF-Verfahren bis zur Ordnung 2 verwendet werden, allerdings ist der Diskretisierungsfehler bei solch niedrigen Ordnungen für Rechnungen mit hoher Genauigkeitsanforderung völlig ungeeignet. BDF-Verfahren höherer Ordnung als 5 sind nicht einmal mehr nullstabil und haben deshalb keine praktische Verwendung.

RADAU5 bietet also eine uneingeschränkte Stabilität bei gleichzeitig hoher Fehlerordnung, also kleinem Diskretisierungsfehler und dies auch bei nichtlinearen Differentialgleichungen. Die BDF-Verfahren hingegen liefern nur bis zur Ordnung $s = 2$ absolute Stabilität, wodurch diese Verfahren für Simulationen mit hoher Anforderung an die Rechengenauigkeit nicht mehr

geeignet sind. Ein Vorteil der BDF-Verfahren hingegen ist ihre höhere Geschwindigkeit, da das zu lösende Gleichungssystem wesentlich kleiner ist als bei RADAU5. Dieser Geschwindigkeitsvorteil lässt sich allerdings nur *ausnutzen*, wenn das zu simulierende System sich relativ weit weg von den oben dargestellten Instabilitätsgebieten befindet. Schwach gedämpfte Systeme stellen also ein Problem für die BDF-Verfahren dar, denn sie machen den Vorteil des einfacheren Gleichungssystems wieder zunichte.

Für *gut* gedämpfte Systeme zeigt CVODE ungefähr die doppelte Leistungsfähigkeit im Vergleich zu RADAU5. Je schwächer die Dämpfung eingestellt wird, desto geringer wird der Unterschied. Bei den zwei in Kapitel 7 und 8 vorgestellten Simulationen lässt sich genau dieses Verhalten beobachten. Ändert man die Dämpfungskonstanten für alle Gelenke zu höheren Werten hin, so wird das System von CVODE schneller integriert. Bei schwächeren Dämpfungskonstanten überwiegt hingegen RADAU5.

Um den Nachteil von expliziten Verfahren bei steifen Differentialgleichungen zu verdeutlichen, ist zum Abschluß noch das Stabilitätsgebiet des expliziten Runge-Kutta-Verfahrens 4. Ordnung dargestellt, ein Verfahren also, das in vielen Bereichen der Physik eine Anwendung findet, aber eben bei steifen Differentialgleichungen kläglich versagt.

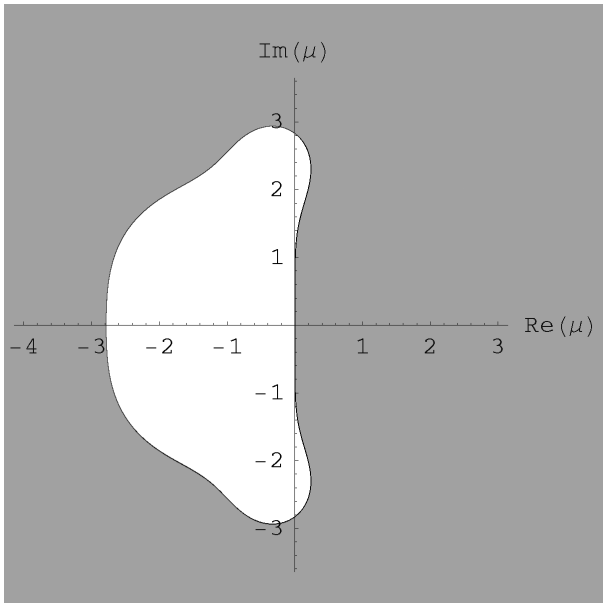


Abbildung 2.5: Runge-Kutta-Verfahren 4. Ordnung

Man erkennt sehr schnell wieso diese Verfahren für steife Differentialgleichungen völlig ungeeignet sind und wieso die Schrittweite sehr kleine Werte annehmen muß um überhaupt sinnvolle Ergebnisse zu liefern. Das weiße Stabilitätsgebiet umfasst nun lediglich einen sehr kleinen Bereich in der linken komplexen Ebene. Je höher die Kraftkonstanten nun gewählt werden, also je größer λ, desto kleiner muß die Schrittweite h gewählt werden um $|R(h\lambda)| < 1$ zu garantieren, um also im Stabilitätsgebiet zu bleiben.

Dieses *Problem* weisen alle expliziten Verfahren auf, gleich welcher Ordnung. Mit steigender Ordnung werden diese Gebiete zwar größer, jedoch ändert dies nichts an der Tatsache, dass explizite Verfahren stets ein begrenztes Stabilitätsgebiet aufweisen und somit automatisch eine begrenzte maximale Schrittweite besitzen.

Kapitel 3

Bewegungsgleichungen

3.1 Einleitung

Es gibt eine Vielzahl von Möglichkeiten und Varianten wie sich die Bewegungsgleichungen für eine Mehrkörpersimulation aufstellen und lösen lassen. Im Endeffekt basieren alle Methoden auf zwei unterschiedlichen Formulierungen. Zum einen gibt es die Methoden, die das System mit einem maximalen Satz an Koordinaten beschreiben und Zwangsbedingungen (zum Beispiel die menschlichen Gelenke) in Form von zusätzlichen Gleichungen (holonom) und/oder Ungleichungen (nicht holomon) formulieren. Zum anderen die Methoden, die im Hinblick auf die Größe des Systems optimal sind und das System mit generalisierten Koordinaten beschreiben, so dass sich die Zwangsbedingungen automatisch erfüllen.

Als Beispiel betrachtet man ein System aus zwei starren Körpern in 3 Dimensionen, die durch ein Scharniergelenk miteinander gekoppelt sind. Es existieren also insgesamt 7 Freiheitsgrade. Um dieses System mit generalisierten Koordinaten zu beschreiben, müssen exakt 7 Koordinaten gefunden werden, die die entsprechenden Freiheitsgrade beinhalten. Eine mögliche Wahl wäre, einen der Körper durch seine Position und Orientierung im Raum zu beschreiben, zum Beispiel durch 3 kartesische Koordinaten und 3 Winkel und den verbleibenden Freiheitsgrad durch den Winkel des Scharniergelenkes festzulegen. Mit diesen 7 Koordinaten wäre das System vollständig beschrieben.

Einen Satz von generalisierten Koordinaten für eine allgemeine biomechanische Simulation zu finden ist aber sehr aufwändig und auch nicht immer möglich wenn nichtholonome und geschwindigkeitsabhängige Zwangsbedingungen existieren. Deshalb werden Zwangsbedingungen meistens durch zusätzliche Gleichungen oder Ungleichungen beschrieben, die bei der Integration des Systems erfüllt sein müssen. Dieser Ansatz verlangt allerdings nach einer speziellen Klasse von Integrationsverfahren um während der Integration die Zwangsbedingungen einzuhalten. Ein Nachteil bei der Beschreibung des Systems mit generalisierten Koordinaten ist, dass der Satz der Koordinaten geändert werden muß wenn eine Zwangsbedingung entfernt oder hinzugefügt wird.

Perfekte Zwangsbedingungen sind zudem bei einer biomechanischen Simulation nicht immer erwünscht, denn menschliche Gelenke sind weit davon entfernt ideale Kugel- oder Scharnier-

gelenke zu sein. In dieser Arbeit werden deshalb die Zwangsbedingungen, also alle Gelenke und Gelenkanschläge durch Feder/Dämpfer Elemente beschrieben. Mit diesem Ansatz lassen sich menschliche Gelenke besser nachbilden als durch reine Gleichungen und/oder Ungleichungen. Um zum Beispiel ein Scharniergelenk zu simulieren, können insgesamt 2 Federn verwendet werden, für ein Kugelgelenk hingegen reicht eine Feder. Durch geeignete Wahl der Federn lassen sich auch kompliziertere Gelenke wie zum Beispiel das Knie realisieren. Im Grenzfall sehr hoher Kraftkonstanten ist dieser Ansatz identisch zu der Formulierung mittels Zwangsbedingungen oder durch generalisierte Koordinaten, denn diese Methoden halten die Bedingungen auch nur mit der Genauigkeit ein, mit der das System integriert wird.

Am Beispiel des Stabpendels konnte dieser Grenzübergang gut gezeigt werden. Für sehr hohe Kraftkonstanten liefert das MATLAB Programm im Anhang A die exakte Schwingungsdauer für ein ideales mathematisches Pendel. Für extrem hohe Kraftkonstanten, wenn also die Amplitude der Federschwingung Größenordnungen der gewünschten Fehlertoleranz erreicht (typischerweise $1e^{-6}$), sind diese Schwingungen auch nicht weiter von Interesse. Deshalb ist die Eigenschaft von impliziten Integrationsverfahren, solche Schwingungen *wegzudämpfen*, äußerst hilfreich. Die Simulationsläufe haben ferner gezeigt, dass die Größe der Kraftkonstanten nahezu keinen Einfluß auf die Integrationsdauer hat.

Der Vorteil des Ansatzes mit Feder/Dämpfer Elementen ist nun zum einen, dass sich die Bewegungsgleichungen sehr schnell aufstellen lassen, zum anderen bietet diese Methode für eine biomechanische Simulation eine größere Flexibilität um menschliche Gelenke besser abbilden zu können. Ferner ist die Größe des Systems unabhängig von der Anzahl der Zwangsbedingungen, denn eine zusätzliche Bedingung fügt lediglich zusätzliche Feder/Dämpfer Elemente hinzu, ändert aber an der Größe des Systems nichts.

In den folgenden Abschnitten werden nun die Bewegungsgleichungen für starre Körper aufgestellt. Eine Erweiterung der Bewegungsgleichungen zur Ankopplung von Schwabbelmassen, um die menschlichen Weichteile zu simulieren, ist ohne weiteres möglich. Alle Zwangsbedingungen, um die Gelenke oder auch die Kollisionen zu simulieren, stecken implizit in den Kräften und Drehmomenten und nicht in separaten Gleichungen und Ungleichungen.

Besondere Bedeutung kommt der geeigneten Aufstellung von Bewegungsgleichungen im Hinblick auf eine effiziente Implementierung mit impliziten Integrationsverfahren zu.

3.2 Aufstellung der Bewegungsgleichungen

Nachfolgend bezeichnet KS das körperfeste Intertialsystem und RS das raumfeste Inertialsystem. Der Schwerpunkt eines Körpers ist so gewählt, dass er sich im Ursprung von KS befindet. Hierdurch vereinfachen sich die Bewegungsgleichungen.

Der Zustand eines starren Körpers ist vollständig durch die Kenntnis der folgenden vier Größen bestimmt

$$\begin{aligned}
\boldsymbol{R}(t) &= \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix} \\
\boldsymbol{V}(t) &= \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} \\
\boldsymbol{O}(t) &= \begin{pmatrix} o_{xx} & o_{xy} & o_{xz} \\ o_{yx} & o_{yy} & o_{yz} \\ o_{zx} & o_{zy} & o_{zz} \end{pmatrix} \\
\boldsymbol{L}(t) &= \begin{pmatrix} l_x \\ l_y \\ l_z \end{pmatrix}
\end{aligned}$$

Hierbei sind $\boldsymbol{R}$ und $\boldsymbol{V}$ die Position und die Geschwindigkeit des Schwerpunkts im Raumsystem, $\boldsymbol{O}$ ist die 3x3 Drehmatrix, die den Körper von KS nach RS transformiert, und $\boldsymbol{L}$ ist der Drehimpuls, wiederum bezogen auf das raumfeste System RS. Es ist zwecksmäßig all diese Größen in einem einzigen Vektor zusammenzufassen:

$$\boldsymbol{Y}(t) = \begin{pmatrix} \boldsymbol{R}(t) \\ \boldsymbol{V}(t) \\ \boldsymbol{O}(t) \\ \boldsymbol{L}(t) \end{pmatrix} \tag{3.1}$$

Zur Darstellung des Zustands sind also insgesamt 18 Werte notwendig. Die Reihenfolge der einzelnen Elemente von $\boldsymbol{Y}(t)$ spielt eine entscheidende Rolle in der Effizienz der Integrationsmethode. Hierauf wird später detaillierter eingegangen. Um dieses System nun integrieren zu können benötigt man noch die zeitliche Änderung des Zustandsvektors $\dot{\boldsymbol{Y}}(t)$.

Der starre Körper habe die Masse M und den Trägheitstensor $\boldsymbol{I}_{\text{KS}}$, die zum Beginn der Simulation bekannt seien und im folgenden als konstant angenommen werden. Hierbei ist $\boldsymbol{I}_{\text{KS}}$ der Trägheitstensor im KS. Da sich der Schwerpunkt des Körpers im Ursprung befindet,

erhält man den Trägheitstensor $\boldsymbol{I}_{\text{RS}}$ in RS durch Transformation mit der Orientierungsmatrix $\boldsymbol{O}$:

$$\boldsymbol{I}_{\text{RS}}(t) = \boldsymbol{O}(t)\boldsymbol{I}_{\text{KS}}\boldsymbol{O}(t)^T$$

Die Bewegungsgleichungen für einen starren Körper sind

$$\begin{aligned} \dot{\boldsymbol{V}}(t) &= \boldsymbol{F}(t) \cdot M^{-1} \\ \dot{\boldsymbol{L}}(t) &= \boldsymbol{T}(t) \end{aligned} \tag{3.2}$$

Hier ist $\boldsymbol{F}$ die Summe aller Kräfte die zum Zeitpunkt t auf den Körper einwirken und $\boldsymbol{T}$ ist die Summe aller auf den Körper wirkenden Drehmomente. In den Kräften und Drehmomenten stecken sowohl die Kräfte hervorgerufen durch Kollisionen als auch die Kräfte der Feder/Dämpfer Elemente, die die Gelenke und Gelenkanschläge simulieren und natürlich auch die Gravitationskraft sowie eventuelle Reibungs- oder Luftwiderstandskräfte.

Mit Hilfe dieser Bewegungsgleichungen kann nun die zeitliche Änderung des Zustandsvektors $\boldsymbol{Y}(t)$, also $\dot{\boldsymbol{Y}}(t)$, bestimmt werden. Für die reine Schwerpunktbewegung kann man dies sofort hinschreiben:

$$\begin{aligned} \dot{\boldsymbol{R}}(t) &= \boldsymbol{V}(t) \\ \dot{\boldsymbol{V}}(t) &= \boldsymbol{F}(t) \cdot M^{-1} \end{aligned} \tag{3.3}$$

Für die rotatorische Bewegung ist dies nicht sofort ersichtlich. Zur Bestimmung von $\dot{\boldsymbol{O}}(t)$ berechnet man zunächst die Winkelgeschwindigkeit des Körpers, die mit dem Drehimpuls $\boldsymbol{L}(t)$ folgendermaßen verknüpft ist

$$\boldsymbol{\omega}(t) = \boldsymbol{I}_{\text{RS}}^{-1}(t)\boldsymbol{L}(t)$$

Die Spalten der Rotationsmatrix $\boldsymbol{O}(t)$ sind aber nichts anderes als die transformierten Achsen von KS im RS, das heißt $\boldsymbol{O}(t)$ transformiert den x-Vektor $(1, 0, 0)$ in das RS

$$\boldsymbol{O}(t) \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} o_{xx} \\ o_{yx} \\ o_{zx} \end{pmatrix}$$

Die analoge Betrachtungsweise gilt auch für den y- und z-Vektor. Um $\dot{\boldsymbol{O}}(t)$ zu bestimmen muß nun lediglich die zeitliche Änderung der einzelnen Spaltenvektoren von $\boldsymbol{O}(t)$ bestimmt werden. Für einen beliebigen Vektor $\boldsymbol{r}(t)$ gilt jedoch für seine zeitliche Ableitung einfach

$$\dot{\boldsymbol{r}}(t) = \boldsymbol{\omega}(t) \times \boldsymbol{r}(t)$$

also gilt auch für die zeitliche Änderung der Spaltenvektoren von $\boldsymbol{O}(t)$

$$\frac{d}{dt}\begin{pmatrix} o_{xx} \\ o_{yx} \\ o_{zx} \end{pmatrix} = \boldsymbol{\omega}(t) \times \begin{pmatrix} o_{xx} \\ o_{yx} \\ o_{zx} \end{pmatrix}$$

Zusammengefasst ergibt dies

$$\dot{\boldsymbol{O}}(t) = \left(\boldsymbol{\omega}(t) \times \begin{pmatrix} o_{xx} \\ o_{yx} \\ o_{zx} \end{pmatrix} \quad \boldsymbol{\omega}(t) \times \begin{pmatrix} o_{xy} \\ o_{yy} \\ o_{zy} \end{pmatrix} \quad \boldsymbol{\omega}(t) \times \begin{pmatrix} o_{xz} \\ o_{yz} \\ o_{zz} \end{pmatrix} \right) \tag{3.4}$$

Dieser Ausdruck kann noch besser dargestellt werden wenn man man die Kreuzprodukte durch eine Matrix-Vektor Multiplikation ersetzt. Seien $\boldsymbol{a}$ und $\boldsymbol{b}$ zwei beliebige Vektoren, dann lässt sich das Kreuzprodukt $\boldsymbol{a} \times \boldsymbol{b}$ auch folgendermaßen schreiben

$$\boldsymbol{a} \times \boldsymbol{b} = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix} \cdot \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} \tag{3.5}$$

Damit ist nun $\dot{\boldsymbol{Y}}(t)$ vollständig bestimmt:

$$\dot{\boldsymbol{Y}}(t) = \begin{pmatrix} \dot{\boldsymbol{R}}(t) \\ \dot{\boldsymbol{V}}(t) \\ \dot{\boldsymbol{O}}(t) \\ \dot{\boldsymbol{L}}(t) \end{pmatrix} = \begin{pmatrix} \boldsymbol{V}(t) \\ \boldsymbol{F}(t) \cdot M^{-1} \\ \boldsymbol{\Omega}(t) \cdot \boldsymbol{O}(t) \\ \boldsymbol{T} \end{pmatrix} \tag{3.6}$$

Hierbei sei $\boldsymbol{\Omega}(t)$ die wie in (3.5) konstruierte Matrix des Vektors $\boldsymbol{\omega}(t)$. Für ein N-Körper System erweitert man nun einfach den Zustandsvektor $\boldsymbol{Y}(t)$ um die zusätzlichen Körper:

$$\boldsymbol{Y}(t) = \begin{pmatrix} \boldsymbol{Y}_1(t) \\ \vdots \\ \boldsymbol{Y}_N(t) \end{pmatrix} = \begin{pmatrix} \boldsymbol{R}_1(t) \\ \boldsymbol{V}_1(t) \\ \boldsymbol{O}_1(t) \\ \boldsymbol{L}_1(t) \\ \vdots \\ \boldsymbol{R}_N(t) \\ \boldsymbol{V}_N(t) \\ \boldsymbol{O}_N(t) \\ \boldsymbol{L}_N(t) \end{pmatrix}, \quad \dot{\boldsymbol{Y}}(t) = \begin{pmatrix} \boldsymbol{V}_1(t) \\ \boldsymbol{F}_1(t) \cdot M_1^{-1} \\ \boldsymbol{\Omega}_1(t) \cdot \boldsymbol{O}_1(t) \\ \boldsymbol{T}_1 \\ \vdots \\ \boldsymbol{V}_N(t) \\ \boldsymbol{F}_N(t) \cdot M_N^{-1} \\ \boldsymbol{\Omega}_N(t) \cdot \boldsymbol{O}_N(t) \\ \boldsymbol{T}_N \end{pmatrix} \tag{3.7}$$

3.3 Wieso Quaternionen?

Die Darstellung der Orientierung durch eine 3x3 Matrix $\boldsymbol{O}$ ist zwar sehr anschaulich, jedoch hat sie den Nachteil, dass für die 3 Freiheitsgrade der Rotation insgesamt 9 Werte verwendet werden müssen. Das heißt, es treten insgesamt 6 zusätzliche Zwangsbedingungen auf. Für die Darstellung (3.7) benötigt man also 18 Werte für jeden starren Körper und insgesamt bei einem N-Körper System $N \cdot 18$ Werte.

Aufgrund des stets vorhandenen numerischen Fehlers durch das Integrationsverfahren und die begrenzte Fließkomma-Genauigkeit des Rechners, wird die Matrix $\boldsymbol{O}$ sich immer mehr von einer echten Rotationsmatrix entfernen, also ihre Orthogonalität verlieren. Dieser Effekt ist äußerst unerwünscht und verfälscht zugleich das Simulationsergebnis. Eine mögliche Lösung wäre die Matrix $\boldsymbol{O}$ *hin- und wieder* zu renormalisieren, jedoch werden sich hierdurch wiederum Fehler einschleichen, denn es ist ja nicht eindeutig wie die Matrix renormalisiert werden soll.

Zusätzlich zu dem oben beschriebenen numerischen Driftproblem ist es aber auch wichtig die Orientierung mit möglichst wenigen Werten darzustellen, um das Integrationsverfahren zu beschleunigen. Wie im Kapitel 2.2 über implizite Integrationsverfahren beschrieben, ist die Größe des Differentialgleichungssystems entscheidend für die Ausführungsgeschwindigkeit. Die Lösung dieses Gleichungssystems kann je nach Problem im schlimmsten Fall ein $O(N^3)$ Prozess sein, wobei N die Anzahl der Elemente des Vektors $\boldsymbol{Y}(t)$ ist.

Die Darstellung der Orientierung durch 3 Winkel, wie zum Beispiel durch Eulerwinkel, ist aber aufgrund ihrer fehlenden Eindeutigkeit ungeeignet. Eulerwinkel haben deshalb keine echte Anwendung in der Computersimulation außer für eine anschauliche Darstellung in einfachen Problemen.

Eine weitere Möglichkeit Rotationen im $\mathbb{R}^3$ darzustellen sind Einheitsquaternionen (Quaternionen der Länge 1). Quaternionen begnügen sich mit 4 Werten, um eine beliebige Rotation im $\mathbb{R}^3$ darzustellen. Bei dieser Darstellung hat man also nur die Zwangsbedingung einzuhalten, dass die Norm des Quaternions stets 1 sein muß. Dies bedeutet eine Ersparnis von 5 Elementen gegenüber der Matrixdarstellung. In den folgenden Kapiteln wird ein kurzer Überblick über Quaternionen gegeben und hergeleitet, wie die Bewegungsgleichungen mit Quaternionen aussehen.

3.4 Definition von Quaternionen

Quaternionen sind die Erweiterung der komplexen Zahlen in 4 Dimensionen. Ein Quaternion ist definiert durch

$$\begin{aligned} \boldsymbol{q} &\equiv s + \boldsymbol{i}x + \boldsymbol{j}y + \boldsymbol{k}z & s,x,y,z \in \mathbb{R} \\ &\equiv \begin{pmatrix} s \\ x \\ y \\ z \end{pmatrix} & s,x,y,z \in \mathbb{R} \\ &\equiv (s, \boldsymbol{v}) & s \in \mathbb{R}, \quad \boldsymbol{v} = (x,y,z) \in \mathbb{R}^3 \end{aligned}$$

Und per Definition gilt $\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = \boldsymbol{ijk} = -1, \qquad \boldsymbol{ij} = \boldsymbol{k}$ und $\boldsymbol{ji} = -\boldsymbol{k}$
Quaternionen bestehen also aus einem skalaren Teil $s \in \mathbb{R}$ und einem vektoriellen Teil $\boldsymbol{v} = (x,y,z) \in \mathbb{R}^3$. Mit Hilfe dieser Definition und den folgenden Eigenschaften lässt sich zeigen, dass mit Quaternionen Drehungen im $\mathbb{R}^3$ dargestellt werden können. Das Konjugierte zu einem Quaternion ist definiert durch

$$\boldsymbol{q}^* \equiv (s, -\boldsymbol{v})$$

Für Addition, Subtraktion und Multiplikation zweier Quaternionen gelten aufgrund der Definition

$$\begin{aligned} \boldsymbol{q} &= \boldsymbol{q_1} \pm \boldsymbol{q_2} = (s_1 \pm s_2, \quad \boldsymbol{v_1} \pm \boldsymbol{v_2}) \\ \boldsymbol{q} &= \boldsymbol{q_1} \cdot \boldsymbol{q_2} = (s_1 s_2 - \boldsymbol{v_1} \cdot \boldsymbol{v_2}, \quad \boldsymbol{v_1} \times \boldsymbol{v_2} + s_1 \boldsymbol{v_2} + s_2 \boldsymbol{v_1}) \end{aligned} \tag{3.8}$$

Die Multiplikation ist allgemein nicht kommutativ

$$\boldsymbol{q_1} \cdot \boldsymbol{q_2} \neq \boldsymbol{q_2} \cdot \boldsymbol{q_1}$$

Das neutrale Element einer Quaternion Multiplikation is $\boldsymbol{I} = (1, \boldsymbol{0})$, das heißt

$$\boldsymbol{qI} = \boldsymbol{Iq} = \boldsymbol{q}$$

Die Norm eines Quaternions ist

$$|\boldsymbol{q}| = \sqrt{s^2 + \boldsymbol{v} \cdot \vec{v}} = \sqrt{s^2 + x^2 + y^2 + z^2}$$

Das Inverse eines Quaternions $\boldsymbol{q}^{-1}$, so dass also $\boldsymbol{q}\boldsymbol{q}^{-1} = \boldsymbol{I}$ gilt, ist

$$\boldsymbol{q}^{-1} = \frac{\boldsymbol{q}^*}{|\boldsymbol{q}|}$$

Für Einheitsquaternionen ($|\boldsymbol{q}| = 1$) ist das Inverse eines Quaternions also durch das konjugierte Quaternion gegeben

$$\boldsymbol{q}^{-1} = \boldsymbol{q}^*$$

3.5 Einheitsquaternionen und Rotationen

Es kann gezeigt werden (für einen anschaulichen Beweis siehe [18]), dass eine beliebige Drehung im $\mathbb{R}^3$ mit dem Winkel θ um eine Achse $\boldsymbol{n}$, mit $|\boldsymbol{n}| = 1$, durch ein Einheitsquaternion ($|\boldsymbol{q}| = 1$) repräsentiert werden kann. $\boldsymbol{q}$ hat dann die spezielle Form

$$\boldsymbol{q} = \left(\cos\frac{\theta}{2}, \sin\frac{\theta}{2}\boldsymbol{n}\right) \tag{3.9}$$

Sei $\boldsymbol{r} = (x, y, z) \in \mathbb{R}^3$ und $\boldsymbol{p} = (0, \boldsymbol{r})$ das aus $\boldsymbol{r}$ gebildete Quaternion, so erhält man den um die Achse $\boldsymbol{n}$ mit dem Winkel θ rotierten Vektor $\boldsymbol{r'}$ mit dem durch (3.9) definierten Einheitsquaternion auf folgende Weise

$$\boldsymbol{p'} = \boldsymbol{qpq^{-1}} \tag{3.10}$$

Bei einer Quaternionenmultiplikation (3.8) bleibt $\boldsymbol{p'}$ im skalaren Teil 0, also erhält man den rotierten Vektor $\boldsymbol{r'}$ durch Extraktion des vektoriellen Teils des Quaternions $\boldsymbol{p'} = (0, \boldsymbol{r'})$.

Zu jedem Einheitsquaternion $\boldsymbol{q}$ existiert noch genau ein weiteres Einheitsquaternion das dieselbe Rotation bewirkt, dies ist

$$\boldsymbol{q'} = (-1) \cdot \boldsymbol{q} = (-s, -\boldsymbol{q})$$

Es sei hier angemerkt, dass die Berechnung von $\boldsymbol{r'}$ durch direkte Transformation mittels Quaternionen mehr Operationen benötigt als eine Matrix-Vektor-Multiplikation. Quaternionen werden deshalb lediglich für das Integrationsverfahren verwendet, um die Anzahl der Elemente des Zustandsvektors $\boldsymbol{Y(t)}$ zu reduzieren. Um von der Quaternionendarstellung zur entsprechenden Matrixdarstellung zu gelangen, sind insgesamt 18 Multiplikationen notwendig. Sei das Quaternion $\boldsymbol{q} = (s, x, y, z)$ gegeben, so erhält man die 3x3 Matrix $\boldsymbol{O}$ folgendermaßen

$$\boldsymbol{O} = \begin{pmatrix} 1 - 2(y \cdot y + z \cdot z) & 2(x \cdot y - r \cdot z) & 2(x \cdot z + r \cdot y) \\ 2(x \cdot y + r \cdot z) & 1 - 2(x \cdot x + z \cdot z) & 2(y \cdot z - r \cdot x) \\ 2(x \cdot z - r \cdot y) & 2(y \cdot z + r \cdot x) & 1 - 2(x \cdot x + y \cdot y) \end{pmatrix} \tag{3.11}$$

Diese Umrechnung ist zwar auch aufwändig, jedoch wird die Transformation eines Vektors durch die Orientierungsmatrix $\boldsymbol{O}$ sehr häufig benötigt, wodurch die Berechnung von $\boldsymbol{O}$ aus $\boldsymbol{q}$ vernachlässigbar wird.

3.6 Bewegungsgleichungen mit Quaternionen

Nun wird hergeleitet wie die zeitliche Änderung der Orientierung $\dot{\boldsymbol{q}}(t)$ mit dem Drehimpuls $\boldsymbol{L}(t)$ bzw. der Winkelgeschwindigkeit $\boldsymbol{\omega}(t)$ verknüpft ist.

Die Winkelgeschwindigkeit $\boldsymbol{\omega}(t)$ eines Körpers bedeutet, dass sich der Körper um die Achse $\boldsymbol{\omega}(t)$ mit der Geschwindigkeit $|\boldsymbol{\omega}(t)|$ ($\frac{rad}{s}$) dreht. Wenn $\boldsymbol{\omega}(t)$ in einem Zeitintervall Δt konstant ist, durchstreift der Körper einen Winkelbereich von $\theta = |\boldsymbol{\omega}(t)|\Delta t$. Also wird die Orientierung eines Körpers nach dem Zeitraum Δt durch das folgende Quaternion repräsentiert

$$\boldsymbol{p}(\boldsymbol{\omega}(t)) = \left(\cos\left(\frac{|\boldsymbol{\omega}(t)|\Delta t}{2}\right), \quad \sin\left(\frac{|\boldsymbol{\omega}(t)|\Delta t}{2}\right)\frac{\boldsymbol{\omega}(t)}{|\boldsymbol{\omega}(t)|}\right) \tag{3.12}$$

Sei nun zum Zeitpunkt t_0 die Orientierung eines Körpers gegeben durch das Quaternion $\boldsymbol{q}(t_0)$, dann wird in erster Näherung die Orientierung zum Zeitpunkt $t = t_0 + \Delta t$ gegeben sein durch erst eine Drehung um $\boldsymbol{q}(t_0)$ gefolgt von einer Drehung um das Quaternion $\boldsymbol{p}(\boldsymbol{\omega}(t_0))$

$$\begin{aligned}
\boldsymbol{q}(t_0 + \Delta t) &= \boldsymbol{p}(\boldsymbol{\omega}(t_0)) \cdot \boldsymbol{q}(t_0) \\
&= \left(\cos\left(\frac{|\boldsymbol{\omega}(t)|\Delta t}{2}\right), \quad \sin\left(\frac{|\boldsymbol{\omega}(t)|\Delta t}{2}\right)\frac{\boldsymbol{\omega}(t)}{|\boldsymbol{\omega}(t)|}\right) \cdot \boldsymbol{q}(t_0) \\
\boldsymbol{q}(t) &= \left(\cos\left(\frac{|\boldsymbol{\omega}(t)|(t - t_0)}{2}\right), \quad \sin\left(\frac{|\boldsymbol{\omega}(t)|(t - t_0)}{2}\right)\frac{\boldsymbol{\omega}(t)}{|\boldsymbol{\omega}(t)|}\right) \cdot \boldsymbol{q}(t_0)
\end{aligned}$$

Mit diesem Ausdruck lässt sich nun $\dot{\boldsymbol{q}}(t)$ herleiten. Da $\boldsymbol{q}(t_0)$ konstant ist, erhält man

$$\begin{aligned}
\dot{\boldsymbol{q}}(t) &= \frac{d}{dt}\left[\left(\cos\left(\frac{|\boldsymbol{\omega}(t)|(t - t_0)}{2}\right), \sin\left(\frac{|\boldsymbol{\omega}(t)|(t - t_0)}{2}\right)\frac{\boldsymbol{\omega}(t)}{|\boldsymbol{\omega}(t)|}\right) \cdot \boldsymbol{q}(t_0)\right] \\
&= \left(-\frac{|\boldsymbol{\omega}(t)|}{2}\sin\left(\frac{|\boldsymbol{\omega}(t)|(t - t_0)}{2}\right), \frac{|\boldsymbol{\omega}(t)|}{2}\cos\left(\frac{|\boldsymbol{\omega}(t)|(t - t_0)}{2}\right)\frac{\boldsymbol{\omega}(t)}{|\boldsymbol{\omega}(t)|}\right)\boldsymbol{q}(t_0)
\end{aligned}$$

Hierbei wurde schon berücksichtigt, dass $\dot{\boldsymbol{\omega}}(t) = \boldsymbol{\omega}(t) \times \boldsymbol{\omega}(t) = \boldsymbol{0}$.

An der Stelle $t = t_0$ erhält man somit

$$\begin{aligned}
\dot{\boldsymbol{q}}(t)|_{t=t_0} &= \left(0, \quad \frac{|\boldsymbol{\omega}(t)|}{2}\frac{\boldsymbol{\omega}(t)}{|\boldsymbol{\omega}(t)|}\right) \cdot \boldsymbol{q}(t_0) \\
&= \frac{1}{2}(0, \boldsymbol{\omega}(t)) \cdot \boldsymbol{q}(t_0)
\end{aligned} \tag{3.13}$$

Die zeitliche Änderung eines Quaternions $\boldsymbol{q}(t)$ unter der Wirkung von $\boldsymbol{\omega}(t)$ erhält man also durch eine einfache Quaternionenmultiplikation.

Der Zustandsvektor $\boldsymbol{Y}(t)$ und seine zeitliche Ableitung $\dot{\boldsymbol{Y}}(t)$ haben in der Quaternionendarstellung jetzt die Form

$$\begin{aligned} \boldsymbol{Y}(t) &= \begin{pmatrix} \boldsymbol{R}(t) \\ \boldsymbol{V}(t) \\ \boldsymbol{q}(t) \\ \boldsymbol{L}(t) \end{pmatrix} \\ \dot{\boldsymbol{Y}}(t) &= \begin{pmatrix} \dot{\boldsymbol{R}}(t) \\ \dot{\boldsymbol{V}}(t) \\ \dot{\boldsymbol{q}}(t) \\ \dot{\boldsymbol{L}}(t) \end{pmatrix} = \begin{pmatrix} \boldsymbol{V}(t) \\ \boldsymbol{F}(t) \cdot M^{-1} \\ \frac{1}{2}\,(0, \boldsymbol{\omega}(t)) \cdot \boldsymbol{q}(t) \\ \boldsymbol{T} \end{pmatrix} \end{aligned} \tag{3.14}$$

Der Zustand eines starren Körpers ist jetzt insgesamt durch 13 Werte vollständig bestimmt. Während der numerischen Integration muß nun lediglich sichergestellt werden, dass $\boldsymbol{q}(t)$ immer die Norm 1 behält. Wie jedoch die numerischen Ergebnisse gezeigt haben, ist eine Nachnormierung in dem für die Simulation relevanten Zeitbereich nicht nötig, vorrausgesetzt die Fehlerschranken werden geeignet gesetzt. Die Darstellung der Bewegungsgleichungen durch (3.14) ist zwar im Hinblick auf die Größe des Systems optimal, allerdings hat die für die impliziten Integrationsverfahren benötigte Jacobimatrix $\boldsymbol{J}$ keine besonders einfache Struktur. Die Jacobimatrix $\boldsymbol{J}$ erhält erst dann eine einfachere Darstellung, wenn die Bewegungsgleichungen in ein Differentialgleichungssystem sortiert und transformiert werden, so dass gilt [1]

$$\dot{Y}_i(t) = Y_{i+m_2}(t), \quad i = 1 \ldots m_1, \quad m_2 \geq m_1 \tag{3.15}$$

In diesem Fall ist $\boldsymbol{J}$ von einer speziellen Struktur. Sei N die Dimension des Systems, dann hat die Jacobimatrix für $m_1 = m_2 = \frac{N}{2}$ die folgende Darstellung

$$\boldsymbol{J} = \left(\begin{array}{cccc|ccc} 0 & 0 & \cdots & 0 & \frac{\partial F_{m_1}}{\partial Y_1} & \frac{\partial F_{m_1+1}}{\partial Y_1} & \cdots & \frac{\partial F_n}{\partial Y_1} \\ 0 & 0 & & \vdots & \frac{\partial F_{m_1}}{\partial Y_2} & & & \vdots \\ \vdots & & 0 & 0 & \vdots & & & \frac{\partial F_n}{\partial Y_{m_1-1}} \\ 0 & \cdots & 0 & 0 & \frac{\partial F_{m_1}}{\partial Y_{m_1}} & \cdots & \frac{\partial F_{n-1}}{\partial Y_{m_1}} & \frac{\partial F_n}{\partial Y_{m_1}} \\ \hline 1 & 0 & \cdots & 0 & \frac{\partial F_{m_1}}{\partial Y_{m_1+1}} & \frac{\partial F_{m_1+1}}{\partial Y_{m_1+1}} & \cdots & \frac{\partial F_n}{\partial Y_{m_1+1}} \\ 0 & 1 & & \vdots & \frac{\partial F_{m_1}}{\partial Y_{m_1+2}} & & & \vdots \\ \vdots & & 1 & 0 & \vdots & & & \frac{\partial F_n}{\partial Y_{n-1}} \\ 0 & \cdots & 0 & 1 & \frac{\partial F_{m_1}}{\partial Y_n} & \cdots & \frac{\partial F_{n-1}}{\partial Y_n} & \frac{\partial F_n}{\partial Y_n} \end{array}\right)$$

[1] Dies ist nur eine, von vielen möglichen Darstellungen. Entscheidend ist, dass alle 1. Ableitungen der Bewegungsgleichungen (also $\boldsymbol{V}(t)$ und $\dot{\boldsymbol{q}}(t)$) sich explizit aus dem Zustandsvektor $\boldsymbol{Y}(t)$ ergeben.

Dabei kann in den wenigsten Fällen von einer speziellen Struktur der rechten Submatrizen ausgegangen werden, da durch Kollision und Gelenk alle Körper miteinander in Verbindung stehen können. Das Programm RADAU5 von Hairer und Wanner [9] bietet für solche Differentialgleichungen schon eine entsprechende Schnittstelle und verwendet dann zur Lösung der Gleichungssysteme ein Verfahren, das die spezielle Struktur von $\boldsymbol{J}$ berücksichtigt, wodurch erhebliche Geschwindigkeitssteigerungen erreicht werden. Für Systeme der Größe $N = 200$ bis 300 lässt sich hierdurch das Verfahren um den Faktor 2 bis 3 beschleunigen.

Um (3.14) in die Form (3.15) zu bringen, muß $\dot{\boldsymbol{L}}(t)$ durch einen Ausdruck für $\ddot{\boldsymbol{q}}(t)$ ersetzt werden. Die Anzahl der Elemente des Zustandvektors $\boldsymbol{Y}(t)$ steigt dann zwar von 13 auf 14, jedoch überwiegen die Vorteile der einfachen Struktur der Jacobimatrix bei der Lösung des Gleichungssystems. Um zu $\ddot{\boldsymbol{q}}(t)$ zu gelangen, wird $\dot{\boldsymbol{q}}(t)$ eine weiteres mal differenziert

$$\begin{aligned}
\ddot{\boldsymbol{q}}(t) &= \frac{d}{dt}\dot{\boldsymbol{q}}(t) \\
&= \frac{d}{dt}\left(\frac{1}{2}(0,\boldsymbol{\omega}(t))\cdot\boldsymbol{q}(t)\right) \\
&= \frac{1}{2}\left[(0,\dot{\boldsymbol{\omega}})\cdot\boldsymbol{q} + (0,\boldsymbol{\omega})\cdot(\frac{1}{2}(0,\boldsymbol{\omega})\cdot\boldsymbol{q})\right]
\end{aligned}$$

Es muß jetzt noch $\dot{\boldsymbol{\omega}}$ bestimmt werden.

$$\dot{\boldsymbol{\omega}}(t) = \frac{d}{dt}\boldsymbol{\omega}(t) = \frac{d}{dt}\left(\boldsymbol{O}(t)\cdot\boldsymbol{I}_{\text{KS}}^{-1}\cdot\boldsymbol{O}^T(t)\cdot\boldsymbol{L}(t)\right)$$

Im folgenden wird das Zeitargument weggelassen, um die Sache etwas übersichtlicher darzustellen. Da der Trägheitstensor in KS zeitunabhängig ist, also $\dot{\boldsymbol{I}}_{\text{KS}} = 0$ (starrer Körper), erhält man somit

$$\dot{\boldsymbol{\omega}} = \dot{\boldsymbol{O}}\cdot\boldsymbol{I}_{\text{KS}}^{-1}\cdot\boldsymbol{O}^T\cdot\boldsymbol{L} + \boldsymbol{O}\cdot\boldsymbol{I}_{\text{KS}}^{-1}\cdot\dot{\boldsymbol{O}}^T\cdot\boldsymbol{L} + \boldsymbol{O}\cdot\boldsymbol{I}_{\text{KS}}^{-1}\cdot\boldsymbol{O}^T\cdot\dot{\boldsymbol{L}}$$

Da $\boldsymbol{O}$ eine orthogonale Transformation ist, gilt $\boldsymbol{O}\cdot\boldsymbol{O}^T = \boldsymbol{1}$. Durch Einfügen dieser Identitätsmatrix erhält man folgende Umformung.

$$\dot{\boldsymbol{\omega}} = (\dot{\boldsymbol{O}}\cdot\underbrace{\boldsymbol{O}^T}_{\boldsymbol{1}})\cdot(\underbrace{\boldsymbol{O}}_{\boldsymbol{1}}\cdot\boldsymbol{I}_{\text{KS}}^{-1}\cdot\boldsymbol{O}^T\cdot\boldsymbol{L}) + \boldsymbol{O}\cdot\boldsymbol{I}_{\text{KS}}^{-1}\cdot(\underbrace{\boldsymbol{O}^T}_{\boldsymbol{1}}\cdot\underbrace{\boldsymbol{O}}_{\boldsymbol{1}}\cdot\dot{\boldsymbol{O}}^T)\cdot\boldsymbol{L} + \boldsymbol{O}\cdot\boldsymbol{I}_{\text{KS}}^{-1}\cdot\boldsymbol{O}^T\cdot\dot{\boldsymbol{L}} \quad (3.16)$$

Dieser Ausdruck lässt sich noch weiter vereinfachen. Differenzierung von $\boldsymbol{O} \cdot \boldsymbol{O}^T = \mathbf{1}$ liefert

$$\frac{d}{dt}(\boldsymbol{O} \cdot \boldsymbol{O}^T) = \dot{\boldsymbol{O}} \cdot \boldsymbol{O}^T + \boldsymbol{O} \cdot \dot{\boldsymbol{O}}^T = 0$$

$$\Rightarrow \dot{\boldsymbol{O}} \cdot \boldsymbol{O}^T = -\boldsymbol{O} \cdot \dot{\boldsymbol{O}}^T$$

Ferner ist, $\dot{\boldsymbol{O}} = \Omega \cdot \boldsymbol{O}$, was schon in (3.6) gezeigt wurde. Also erhält man

$$\dot{\boldsymbol{O}} \cdot \boldsymbol{O}^T = (\Omega \cdot \boldsymbol{O}) \cdot \boldsymbol{O}^T = \Omega = \boldsymbol{\omega}\times$$

Setzt man all dies in den letzten Ausdruck (3.16) für $\dot{\omega}$ ein, so erhält man

$$\begin{aligned} \dot{\boldsymbol{\omega}} &= \boldsymbol{\omega} \cdot \boldsymbol{\omega} - \boldsymbol{O} \cdot \boldsymbol{I}_{\text{KS}}^{-1} \cdot \boldsymbol{O}^T \cdot \boldsymbol{\Omega} \cdot \boldsymbol{L} + \boldsymbol{O} \cdot \boldsymbol{I}_{\text{KS}}^{-1} \cdot \boldsymbol{O}^T \cdot \dot{\boldsymbol{L}} \\ &= \boldsymbol{\omega} \times \boldsymbol{\omega} - \boldsymbol{O} \cdot \boldsymbol{I}_{\text{KS}}^{-1} \cdot \boldsymbol{O}^T \cdot \boldsymbol{\omega} \times \boldsymbol{L} + \boldsymbol{O} \cdot \boldsymbol{I}_{\text{KS}}^{-1} \cdot \boldsymbol{O}^T \cdot \boldsymbol{T} \\ &= \boldsymbol{O} \cdot \boldsymbol{I}_{\text{KS}}^{-1} \cdot \boldsymbol{O}^T \cdot (\boldsymbol{T} - \boldsymbol{\omega} \times \boldsymbol{L}) \\ &= \boldsymbol{I}_{\text{RS}}^{-1} \cdot (\boldsymbol{T} - \boldsymbol{\omega} \times \boldsymbol{L}) \end{aligned} \tag{3.17}$$

Nun muß lediglich die Reihenfolge der Elemente im Zustandsvektor $\boldsymbol{Y}(t)$ geändert werden, um auf die Form (3.15) zu gelangen.

$$\begin{aligned} \boldsymbol{Y}(t) &= \begin{pmatrix} \boldsymbol{R}(t) \\ \boldsymbol{q}(t) \\ \boldsymbol{V}(t) \\ \dot{\boldsymbol{q}}(t) \end{pmatrix} \\ \dot{\boldsymbol{Y}}(t) &= \begin{pmatrix} \dot{\boldsymbol{R}}(t) \\ \dot{\boldsymbol{q}}(t) \\ \dot{\boldsymbol{V}}(t) \\ \ddot{\boldsymbol{q}}(t) \end{pmatrix} = \begin{pmatrix} \boldsymbol{V}(t) \\ \frac{1}{2}\,(0, \boldsymbol{\omega}(t)) \cdot \boldsymbol{q}(t) \\ \boldsymbol{F}(t) \cdot M^{-1} \\ \frac{1}{2}\left[(0, \dot{\boldsymbol{\omega}}) \cdot \boldsymbol{q} + (0, \boldsymbol{\omega}) \cdot (\frac{1}{2}(0, \boldsymbol{\omega}) \cdot \boldsymbol{q})\right] \end{pmatrix} \end{aligned} \tag{3.18}$$

Um die Übersicht zu bewahren ist der Ausdruck für $\dot{\boldsymbol{\omega}}$ nicht eingesetzt worden. Diese Differentialgleichungen gelten jetzt nur für einen einzigen starren Körper. Zur Erweiterung auf eine beliebige Anzahl von Körpern, muß man dieses Schema nur konsequent fortsetzen. Um weiterhin die Form (3.15) zu erhalten, erhält man also für ein N-Körper System

$$\boldsymbol{Y}(t) = \begin{pmatrix} \boldsymbol{R}_1(t) \\ \boldsymbol{q}_1(t) \\ \vdots \\ \boldsymbol{R}_N(t) \\ \boldsymbol{q}_N(t) \\ \boldsymbol{V}_1(t) \\ \dot{\boldsymbol{q}}_1(t) \\ \vdots \\ \boldsymbol{V}_N(t) \\ \dot{\boldsymbol{q}}_N(t) \end{pmatrix}, \qquad \dot{\boldsymbol{Y}}(t) = \begin{pmatrix} \boldsymbol{V}_1(t) \\ \dot{\boldsymbol{q}}_1(t) \\ \vdots \\ \boldsymbol{V}_N(t) \\ \dot{\boldsymbol{q}}_N(t) \\ \boldsymbol{F}_1(t) \cdot M^{-1} \\ \ddot{\boldsymbol{q}}_1(t) \\ \vdots \\ \boldsymbol{F}_N(t) \cdot M^{-1} \\ \ddot{\boldsymbol{q}}_N(t) \end{pmatrix} \tag{3.19}$$

Die hier aufgestellten Bewegungsgleichungen gelten ausschließlich für starre Körper. Manchmal ist es bei biomechanischen Simulationen wichtig auch den Einfluß der Weichteile zu berücksichtigen. Um den Rechenaufwand in Grenzen zu halten werden Weichteile mittels zweier starrer Körper realisiert wobei einer der Körper mit einer Feder an den anderen Körper gebunden wird. Die Stärke der Feder bestimmt, wie *schwabbelig* der Einfluß der Weichteile ist. Es ist insofern ausreichend das obige Gleichungssystem um zusätzliche Weichteilkörper zu erweitern, die an die Hauptkörper mit Federn gekoppelt sind.

Kapitel 4

Bestimmung der Masseeigenschaften

In vielen Simulationen wird strikt zwischen der grafischen Darstellung und der eigentlichen Simulation getrennt. Es besteht also keine direkte Verbindung zwischen der 3D-Grafik auf der einen und dem physikalischen Modell auf der anderen Seite. In diesem Kapitel soll nun ein Verfahren vorgestellt werden, das diese Lücke schließt und direkt aus den 3D-Modellen der Körper die Eigenschaften wie Masse, Schwerpunkt und Trägheitstensor berechnet. Unter Körper sind hier alle Objekte zu verstehen, die dynamisch an der Simulation teilnehmen, also die menschlichen Körperteile wie Kopf, Arme und Beine sowie der PKW oder die Reckstange.

4.1 Bisherige Ansätze

Die Messung von Masse und Trägheitstensor bei den Körperteilen des Menschen ist äußerst aufwändig und extrem fehlerbehaftet. In dem historischen Überblick von Jørgen Bjørnstrup ([13] und [14]) über die verschiedenen Messmethoden sowohl an lebenden als auch an toten Menschen wird deutlich wie groß die Schwankungen der einzelnen Verfahren zueinander sowie die Schwankungen des eigentlichen Messprozesses an sich sind. Viele experimentelle Verfahren nehmen eine konstante Dichteverteilung an und erzielen hiermit gute Ergebnisse, die vergleichbar zu den aufwändigen Messmethoden an toten Menschen sind. Es existieren auch ausgereifte Programme, wie zum Beispiel das an der Universität Tübingen entwickelte CALCMAN, das aus einer geeigneten Anzahlen von Eingabedaten, wie zum Beispiel Gewicht, Körpergröße und Geschlecht, sehr gute Abschätzungen für Masse und Trägheitsmomente von menschlichen Körperteilen berechnet. CALCMAN verwendet hierzu Regressionsgleichungen, die aus den bei der NASA durchgeführten Messungen gewonnen wurden (siehe [15], [16] und [17]).

Es kann hier jedoch nicht genug betont werden, dass die Abweichungen dieser Parameter bei selbst gleich großen und gleich schweren Menschen nicht zu vernachlässigen sind. Typische Abweichungen einzelner Parametern, vor allem der Trägheitsmomente, bewegen sich bis hin zu 20%. Eine Simulation mit fixen Parametern durchzuführen wäre also wenig aussagekräftig. Es muss deshalb eine Variation dieser Parameter integraler Bestandteil der Simulation sein, um eine zuverlässige Aussage des Ergebnisses zu erhalten.

4.2 Masseeigenschaften aus Dreiecksnetzen

Das hier in dieser Arbeit entwickelte Verfahren berechnet nun direkt aus dem vorliegenden 3D Gittermodell eines Objektes (z.B. eines menschlichen Körperteils) seine Masseeigenschaften. Dies geschieht unter der Annahme einer konstanten Masseverteilung im Objekt. Vergleiche mit bestehenden Messungen zeigen, dass dies gerechtfertigt ist. Da sowieso eine Sensitivitätsanalyse der Parameter durchgeführt wird, ist die exakte Kenntnis auch nicht weiter notwendig. Allerdings wird diese Methode bei aufwändigen 3D-Modellen, wie zum Beispiel beim PKW, nicht verwendet, da erst jedes Autoteil als 3D-Gitter vorliegen und zusätzlich auch noch die Dichte dieser Teile bekannt sein müsste. Ein solches 3D-Modell zu erstellen wäre allerdings sehr aufwändig und im Endeffekt auch unnötig, denn diese Parameter können recht gut mit Hilfe der Herstellerdaten abgeschätzt werden.

Man könnte eine weitere Verfeinerung bei den Körperteilen eines Menschen erzielen, wenn 3D Gittermodelle der Knochen und Schwabbelmassen getrennt vorliegen würden und die Parameter dann einzeln für den Knochen und die Schwabbelmassen berechnet würden. Hierfür fehlten aber die geeigneten 3D Flächen für das Knochengerüst des gesamten Menschen.

Es soll nun ein Algorithmus hergeleitet werden, der aus dem vorliegenden Dreiecksnetz eines Objekts [1] unter der Annahme einer konstanten Dichteverteilung das Volumen, den Schwerpunkt und den Trägheitstensor bestimmt. Je nach menschlichem Körperteil liegt diese Dichte zwischen 1.0 und 1.1 g/cm^3. Nur der Oberkörper hat eine geringere Dichte von ca. 0.8 g/cm^3.

Damit der Algorithmus vernünftige Werte liefert, muß das Dreiecksnetz vollständig geschlossen sein, das heißt jede Kante muß immer genau 2 Nachbardreiecke besitzen [2]. Ein Objekt kann auch aufgeteilt sein in mehrere geschlossene Flächen oder einen Hohlraum im Innern besitzen (wie zum Beispiel bei einem schweizer Käse). Die folgende Grafik veranschaulicht, welche Typen von Objekten für den Algorithmus zulässig sind.

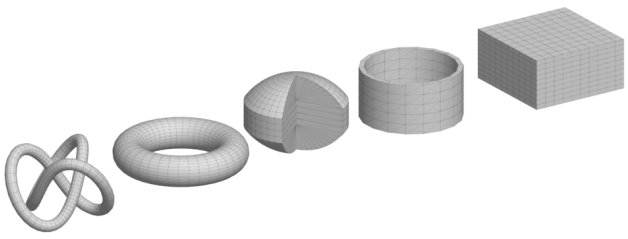


Abbildung 4.1: Zulässige Dreiecksnetze zur Berechnung der Masseeigenschaften

Um Volumen, Schwerpunkt und Trägheitstensor zu bestimmen, müssen die folgenden Integrale über das gesamte Volumen des Dreiecksnetzes bestimmt werden. Da die Dichte $\rho(\mathbf{r})$

[1] Die Verallgemeinerung auf Netze mit mehrseitigen Polygonen ist ohne weiteres auch möglich, hierbei wird das Polygon in einzelne Dreiecke zerlegt.

[2] Es dürfen also keine Löcher im Netz vorhanden sein.

als konstant angenommen wird, kann sie vor das Integral gezogen werden.

$$\begin{aligned}
\text{Volumen:} \quad & Vol &&= \int_V 1 \cdot dV \\
\text{Schwerpunkt:} \quad & R_i &&= \int_V x_i \cdot dV, \quad i = 1,2,3,\ x_1 = x, x_2 = y, x_3 = z \\
\text{Trägheitstensor:} \quad & I_{ij} &&= \rho \cdot \int_V (\mathbf{r}^2 \delta_{ij} - x_i x_k) \cdot dV, \quad \mathbf{r} = (x,y,z)
\end{aligned}$$

Hierbei bezeichnet V die gesamte Integrationsfläche im Innern des Dreiecksnetzes. Da der Trägheitstensor antisymmetrisch ist, müssen insgesamt 10 Integrale bestimmt werden, 1 für das Volumen, 3 für die Schwerpunktberechnung und die restlichen 6 zur Bestimmung des Trägheitstensors.

Um die Integrale zu lösen, muß eine geeignete Parametrisierung gefunden werden. Deshalb wird das Volumen das durch die Dreiecksfläche begrenzt wird zuerst in einzelne Tetraeder aufgegliedert und dann das gesamte Integral als Summe der Integrale über alle Tetraedervolumina berechnet.

$$\int_V f(\boldsymbol{r}) dV = \sum_{\Delta} \left(\int_{\text{V(Tetraeder)}} f(\boldsymbol{r}) dV \right)$$

Hierbei steht $f(\boldsymbol{x})$ repräsentativ für eines der oben genannten Integrale und die Summe erstreckt sich über alle Dreiecke des Netzes. Die folgende 3D-Darstellung einer einfachen Kugel soll dies veranschaulichen. Exemplarisch ist hier ein Dreieck herausgegriffen worden und der entsprechende Tetraeder visualisiert.

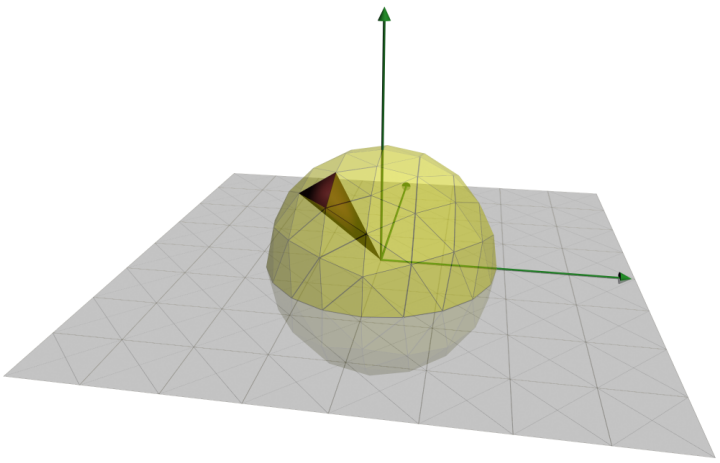


Abbildung 4.2: Veranschaulichung zur Berechnung der Masseeigenschaften

Es ist hierbei gleichgültig wo sich die Spitze des Tetraeders befindet, sie kann innerhalb aber auch außerhalb des Dreiecksnetzes liegen. Für die Bestimmung des Volumens und des Schwerpunkts spielt das keine Rolle. Nachdem der Schwerpunkt bestimmt wurde, wird der Körper

so transformiert, dass der Schwerpunkt im Ursprung liegt. Erst dann wird der Trägheitstensor berechnet. Würde die Schwerpunkttransformation nicht vorgenommen, so hätte der Trägheitstensor natürlich eine andere Gestalt. Das Volumen eines Tetraeders zu parametrisieren wäre allerdings sehr mühsam. Es wird deshalb das Volumenintegral mit dem Satz von Gauss in ein Oberflächenintegral transformiert

$$\int_V (div\mathbf{A})dV = \oint_{\partial V} (\mathbf{A} \cdot \mathbf{n})dF \tag{4.1}$$

Hierbei bezeichnet ∂V den Rand des Volumens V, $\mathbf{n} \in \mathbb{R}^3$ ist die Außennormale von ∂V und $\mathbf{A} : V \to \mathbb{R}^3$ ein C^∞ - Vektorfeld. Theoretisch müssen also für jeden Tetraeder 4 Oberflächenintegrale gelöst werden. Da aber die 3 innen liegenden Tetraederflächen jeweils paarweise vorkommen, jedoch mit entgegengesetzten Normalenvektoren, heben sie sich auf. Es reicht also aus lediglich das Oberflächenintegral der außen liegenden Dreiecksfläche zu bestimmen. Das ursprüngliche Volumintegral reduziert sich somit auf ein einfaches Oberflächenintegral über alle Dreiecksflächen. Die Vektorfelder für die 10 Integrale sind schnell gefunden,

Volumen	$V:$	$\boldsymbol{A} = (x, 0, 0)$	$\Rightarrow$	$div\boldsymbol{A} = 1$
Schwerpunkt	$R_x:$	$\boldsymbol{A} = (\frac{1}{2}x^2, 0, 0)$	$\Rightarrow$	$div\boldsymbol{A} = x$
	$R_y:$	$\boldsymbol{A} = (0, \frac{1}{2}y^2, 0)$	$\Rightarrow$	$div\boldsymbol{A} = y$
	$R_z:$	$\boldsymbol{A} = (0, 0, \frac{1}{2}z^2)$	$\Rightarrow$	$div\boldsymbol{A} = z$
Trägheitstensor	$I_{xx}:$	$\boldsymbol{A} = (0, \frac{1}{3}y^3, \frac{1}{3}z^3)$	$\Rightarrow$	$div\boldsymbol{A} = y^2 + z^2$
	$I_{yy}:$	$\boldsymbol{A} = (\frac{1}{3}x^3, 0, \frac{1}{3}z^3)$	$\Rightarrow$	$div\boldsymbol{A} = x^2 + z^2$
	$I_{zz}:$	$\boldsymbol{A} = (\frac{1}{3}x^3, \frac{1}{3}y^3, 0)$	$\Rightarrow$	$div\boldsymbol{A} = x^2 + y^2$
	$I_{xy}:$	$\boldsymbol{A} = (-\frac{1}{2}x^2y, 0, 0)$	$\Rightarrow$	$div\boldsymbol{A} = -xy$
	$I_{xz}:$	$\boldsymbol{A} = (-\frac{1}{2}x^2z, 0, 0)$	$\Rightarrow$	$div\boldsymbol{A} = -xz$
	$I_{yz}:$	$\boldsymbol{A} = (0, -\frac{1}{2}y^2z, 0)$	$\Rightarrow$	$div\boldsymbol{A} = -yz$

Natürlich gibt es noch weitere Vektorfelder, die die gleiche Divergenz liefern. Die obigen Vektorfelder ergeben aber recht einfache Ausdrücke nach der Bestimmung der Integrale.

Nun muss noch die Parametrisierung einer beliebigen Dreiecksfläche bestimmt werden. Seien $\boldsymbol{a}$, $\boldsymbol{b}$ und $\boldsymbol{c}$ die Eckpunkte eines Dreiecks, so ist eine Parametrisierung dieser Fläche gegeben durch

$$\boldsymbol{s}(t_1, t_2) = \boldsymbol{a} + t_1(\boldsymbol{b} - \boldsymbol{a}) + t_2(\boldsymbol{c} - \boldsymbol{a}) \qquad t_1 = 0 \ldots 1, \quad t_2 = 0 \ldots t_1 \tag{4.2}$$

Und der Normalenvektor ist

$$\boldsymbol{n} = \frac{(\boldsymbol{b} - \boldsymbol{a}) \times (\boldsymbol{c} - \boldsymbol{a})}{|(\boldsymbol{b} - \boldsymbol{a}) \times (\boldsymbol{c} - \boldsymbol{a})|}$$

Nun muss das Oberflächenintegral (4.1) mit der Parametrisierung (4.2) transformiert werden. Das infinitesimale Flächenelement der Parametrisierung (4.2) lautet

$$dxdy = \frac{1}{2}|(\boldsymbol{b}-\boldsymbol{a})\times(\boldsymbol{c}-\boldsymbol{a})|dt_1dt_2$$

Denn $\frac{1}{2}|(\boldsymbol{b}-\boldsymbol{a})\times(\boldsymbol{c}-\boldsymbol{a})|$ ist gerade der Flächeninhalt des Dreiecks. Zusammengefasst erhält man nun

$$\begin{aligned}\oint_{\partial\Delta}(\boldsymbol{A}\cdot\boldsymbol{n})dxdy &= \int_{\partial\Delta}\boldsymbol{A}\cdot\frac{(\boldsymbol{b}-\boldsymbol{a})\times(\boldsymbol{c}-\boldsymbol{a})}{|(\boldsymbol{b}-\boldsymbol{a})\times(\boldsymbol{c}-\boldsymbol{a})|}\cdot\frac{1}{2}|(\boldsymbol{b}-\boldsymbol{a})\times(\boldsymbol{c}-\boldsymbol{a})|dt_1dt_2\\ &= \frac{1}{2}\left((\boldsymbol{b}-\boldsymbol{a})\times(\boldsymbol{c}-\boldsymbol{a})\right)\int_{\partial\Delta}\boldsymbol{A}dt_1dt_2\end{aligned}$$

Die Auswertung dieser Integrale ist nicht weiter schwierig, verlangt nur etwas Sorgfalt. Eine Zusammenfassung findet sich im Anhang B. Exemplarisch werden jetzt noch die Masseeigenschaften Volumen, Schwerpunkt und Trägheitstensor einer Kugel, eines Quaders sowie eines Torus in Abhängigkeit von der Dreiecksanzahl bestimmt und die Ergebnisse mit denen der idealisierten Körper verglichen.

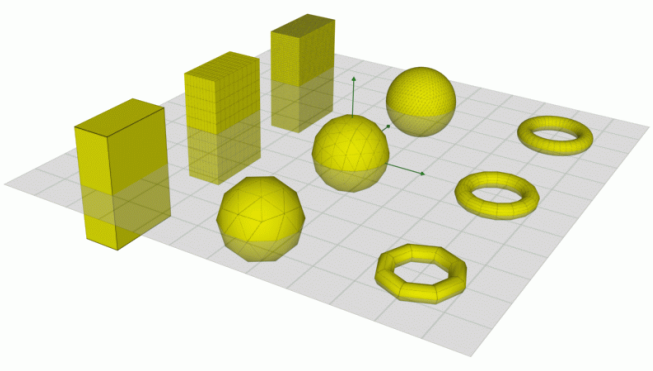


Abbildung 4.3: 9 Dreiecksnetze zur Bestimmung der Masseeigenschaften

Das erste Objekt ist eine Kugel mit Radius $R = 1m$. Für hohe Dreieckszahlen muss das Volumen gegen das einer idealisierten Kugel konvergieren, also $V = \frac{4}{3}\pi R^3$. Aufgrund der Rotationssymetrie ist der Trägheitstensor einer Kugel diagonal und seine Elemente alle identisch $I = \frac{2}{5}MR^2$. Die Dichte wurde für alle Dreiecksnetze auf $\rho = 1.0kg \cdot V^{-1}$ gesetzt, so dass die idealisierte Kugel exakt $1.0kg$ wiegt.

Dreiecke	V	I_{xx}	I_{yy}	I_{zz}
80	3.65871	0.31950	0.31950	0.31950
200	3.95395	0.36390	0.36390	0.36237
8000	4.18303	0.39908	0.39908	0.39908
100820	4.18787	0.39985	0.39985	0.39985
Ideale Kugel	4.18879	0.40000	0.40000	0.40000

Man ersieht aus dieser Tabelle sehr gut, dass mit zunehmender Dreieckzahl die Werte gegen die einer perfekten Kugel streben. Es bedarf allerdings schon einer gewaltigen Anzahl von Dreiecken, um eine idealisierte Kugel auch nur annähernd abzubilden. Der Schwerpunkt sowie die Trägheitstensorelemente I_{xy}, I_{xz}, I_{yz} sind in der Tabelle nicht mit aufgeführt, da in allen 4 Fällen die Genauigkeit in der Größenordnung von EPS lag, wobei EPS prozessorabhängig gegeben ist durch die Zahl für die noch $1 + EPS > 1$ gilt. Bei DOUBLE Genauigkeit ist dies zirka $1.0e^{-16}$. Als nächstes Objekt wurde ein Quader mit dem Seitenverhältniss $l_x = 1m, l_y = 2m, l_z = 3m$ gewählt. Die Orientierung des Quaders wurde so gelegt, dass die Achsen des Quaders mit den x-y-z Achsen des Koordinatensystems übereinstimmen. Deshalb ist der Trägheitstensor diagonal, mit den Momenten

$$I_{xx} = \frac{M}{12}(l_y^2 + l_z^2), \quad I_{yy} = \frac{M}{12}(l_x^2 + l_y^2) \quad \text{und} \quad I_{zz} = \frac{M}{12}(l_x^2 + l_y^2)$$

Die Dichte $\rho = 1.0kg \cdot (l_x \cdot l_y \cdot l_z)^{-1}$ resultiert in einer Gesamtmasse von $1.0kg$.

Dreiecke	V	I_{xx}	I_{yy}	I_{zz}
80	6.00000	1.08333	0.83333	0.41667
200	6.00000	1.08333	0.83333	0.41667
8000	6.00000	1.08333	0.83333	0.41667
Idealer Quader	6.00000	1.08333	0.83333	0.41667

Wie zu erwarten war hängt die Genauigkeit der Werte nicht von der Anzahl der verwendeten Dreiecke ab. In allen 3 Fällen werden also die Masseeigenschaften des idealen Quaders korrekt berechnet.

Als letztes Objekt wurde noch ein Torus mit Aussenradius $R = 0.8m$ und dem Radius des Rings von $r = 0.2m$ gewählt. Das Volumen eines idealisierten Torus ist gegeben durch $V = 2\pi^2 R r^2$ und aufgrund der Rotationssymmetrie um die Hochachse sind die Trägheitsmomente

$$I_{xx} = I_{yy} = \frac{1}{8}M(4R^2 + 5r^2) \quad \text{und} \quad I_{zz} = M(R^2 + \frac{3}{4}r^2)$$

Dreiecke	V	I_{xx}	I_{yy}	I_{zz}
128	0.51200	0.25127	0.25127	0.48792
512	0.59985	0.31896	0.31896	0.61941
8192	0.62963	0.34332	0.34332	0.66673
32768	0.63115	0.34458	0.34458	0.66918
Idealer Torus	0.63165	0.34500	0.34500	0.67000

Die Dichte für alle 4 Dreiecksnetze wurde so gesetzt, dass sich für den idealen Torus ein Gewicht von $1kg$ ergibt. Auch für einen Torus bedarf es wieder einer grossen Anzahl an Dreiecken damit die Masseeigenschaften des idealen Torus erreicht werden. Die Trägheitstensorelemente I_{xy}, I_{xz}, I_{yz} liegen wieder unterhalb der DOUBLE Genauigkeit von EPS.

Kapitel 5

Kollisionserkennung

Unter dem Gesichtspunkt einer schnellen und interaktiven Simulationsgeschwindigkeit ist es mit heutigen Rechnern noch nicht möglich eine Kollision mit deformierbaren Körpern zu realisieren [1]. Bei einer Mensch-PKW Simulation müßte sowohl die Deformation des PKW als auch die Deformation des menschlichen Gewebes berücksichtigt werden. Unabhängig vom Aufwand der Implementierung und Integration deformierbarer Körper müsste dann natürlich auch die genaue Physik bekannt sein, mit der sich diese Körper deformieren. Während dies bei PKW Crashsimulationen mittels finiter Elemente schon erfolgreich von den Autofirmen angewendet wird (auch hier dauern die Simulationen trotz leistungsfähiger Superrechner Stunden, bis hin zu mehreren Tagen), ist dieser Ansatz beim menschlichen Gewebe und Knochen sehr viel komplexer.

Die Kollisionen in biomechanischen Simulationen werden deshalb mittels einer abstandsabhängigen Kraftwirkung simuliert. Hierbei verändern die Körper jedoch nicht ihre Form und bleiben starr. Zu jedem Zeitschritt wird der Abstand aller Körper zueinander bestimmt. Wenn für ein Körperpaar ein gewisser Mindestabstand unterschritten wird, wird eine Kraft wirksam, die qualitativ das gleiche Verhalten simuliert, das bei deformierbaren Körpern zu erwarten ist. Ab welchem Abstand eine Kraft wirkt, hängt davon ab wie hart, bzw. wie weich, ein Körper ist und welche Deformation bei einer gewissen Kraft zu erwarten ist. Bei gleicher Kraftwirkung deformiert sich der Brustkasten eines Menschen natürlich erheblich leichter und stärker als der menschliche Schädel. Beim PKW ist die Motorhaube erheblich weicher als die A-Säule und wird auch erheblich stärkere Deformationen erleiden. Dieses unterschiedliche Verhalten muß dann auch das Kraftgesetz entsprechend nachbilden.

In den Arbeiten von Valentin Keppler [5] und Andreas Sporrer [20] gibt es hierzu verschiedene Abschätzungen wie ein solches Kraftgesetz aussehen kann, und es werden zudem Abschätzungen für die zu verwendeten Kraft- und Dämpfungskonstanten gegeben. Der Vergleich mit realen Menschversuchen und bekannten PKW Unfalldaten lieferte eine gute Übereinstimmung.

Bestehende Arbeiten verwenden einfache Grundkörper, zum Beispiel Ellipsoide, für die ei-

[1] Sollte das Gesetz von Intel Mitbegründer Gordon Moore die nächsten 10 bis 20 Jahre bestand haben, nachdem sich ca. alle 18 Monate die Anzahl der Transistoren pro Chipfläche verdoppelt und hiermit auch die Geschwindigkeit der Prozessoren verdoppelt, so sind deformierbare Netze in interaktiven Geschwindigkeiten frühestens in 10 Jahren zu erwarten.

ne analytische Abstandsbestimmung ohne weiteres möglich ist, und bilden hiermit die PKW Front und den Menschen näherungsweise nach. Der Nachteil dieser Methoden ist allerdings, dass die doch recht komplexen PKW Geometrien und der Mensch nur schlecht repräsentiert werden. Valentin Keppler [5] ging in seiner Arbeit deshalb einen Schritt weiter und verwendet den Algorithmus PQP [21], entwickelt an der Universität Chapel Hill in North Carolina, der für beliebige Dreiecksflächen den minimalen Abstand in Form eines Punktpaares ermittelt. Hiermit ist es erstmals möglich das echte Aussehen des PKW und des Menschen in die Simulation einfließen zu lassen. Ein entscheidender Nachteil dieses Ansatzes liegt in der Tatsache, dass jeweils immer nur ein Punktpaar zurückgeliefert wird. Befinden sich mehrere Punkte innerhalb des Minimalabstands, wie dies bei einem Würfel auf einem Tisch der Fall ist, so werden diese nicht berücksichtigt. Dies ist jedoch im Hinblick auf die Geschwindigkeit und Genauigkeit der Integration von entscheidender Bedeutung, was sich im Falle von Objekten mit langen geraden Kanten sehr schön zeigen lässt. Im Abschnitt 5.2 wird dies an einem Beispiel verdeutlicht.

Theoretisch gibt es zwei Möglichkeiten wie sich ein solcher Abstand für Dreiecksflächen definieren und bestimmen lässt. Die erste Methode ist den minimalen Abstand zweier Körper zu bestimmen, die sich nicht durchdringen. Das Ergebnis ist so immer eindeutig bestimmt. Eine weitere Möglichkeit wäre, den Abstand über die Eindringtiefe, zum Beispiel die minimalste Translation, die Objekte wieder trennt, oder das Überlappungsvolumen sich durchdringender Dreiecksflächen zu definieren.

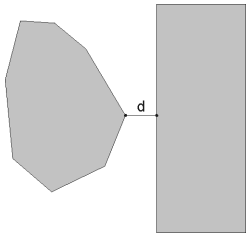


Abbildung 5.1: Abstand

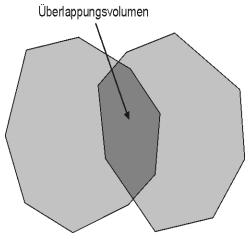


Abbildung 5.2: Überlappungsvolumen

Das letzte Verfahren ist allerdings erheblich aufwändiger zu implementieren und die exakte Bestimmung des Überlappungsvolumens ist zudem bei der Leistung heutiger Rechner absolut indiskutabel. Die Eindringtiefe über die Länge des Vektors zu definieren, der beide Körper wieder trennt, ist auch nicht immer eindeutig und es gibt u.U. mehr als nur einen Lösungsvektor. Im folgenden wird deshalb stets von einem abstandsbasierten Kollisionsalgorithmus ausgegangen, bei dem vorausgesetzt wird, dass sich zwei Körper nicht durchdringen dürfen. Es gibt eine Vielzahl an Algorithmen, die den Minimalabstand für zwei polygonale Körper ermitteln. Einige Verfahren machen sich dabei die spezielle Topologie eines Körpers zunutze, andere hingegen verlangen als Eingabe nur unzusammenhängende Polygone. ObbTree [23], PQP [21], I-Collide [27] und Swift++ [25]/[26] sind hierbei einige der bekanntesten. Diese Verfahren verlangen dabei nicht nach einer speziellen Topologie der Dreiecksflächen und können somit auch aus Polygonsuppen (völlig unzusammenhängende Flächen) bestehen. Dabei wird das 3D-Modell in eine Bounding Volume Hierarchie (BVH) unterteilt um schnell die Kollisionsorte lokalisieren zu können. Ein besonders schnelles Verfahren um BVH zu beschleu-

nigen ist der ADB-Baum Ansatz von Zachmann und Klein [24], der mittels approximativer Methoden die Kollisionserkennung deutlich beschleunigt. In seiner jetzigen Form ist dieses Verfahren aber für physikalisch exakte Simulationen (die mit einer definierten Fehlertoleranz rechnen sollen) ungeeignet, da es Fälle geben kann, in denen Kollisionen übersehen werden. Nicht jedes der obigen Verfahren liefert zudem wirklich den exakten minimalen Abstand zweier Körper, sondern teilweise auch nur eine Näherung. Auch dies ist wieder eine zu starke Einschränkung um in einer physikalisch exakten Simulation eingesetzt zu werden, es sei denn die Näherung ist immer eindeutig. V-Clip [30], Lin-Canny [33], GJK [31], Enhancing GJK [32] und V-Collide [28] hingegen sind Verfahren, die nach einer speziellen Topologie verlangen und auch den exakten Minimalabstand berechnen. Der Vollständigkeit halber sei hier noch das Verfahren DEEP [29] genannt, das näherungsweise die Eindringtiefe zweier Körper bestimmt. Leider ist dieses Verfahren noch nicht sehr robust und auf konvexe geschlossene Oberflächen beschränkt. Die Geschwindigkeit von DEEP liegt noch einige Größenordnungen unter den rein abstandsbasierten Verfahren.

All diese Kollisionsverfahren lassen sich generell dadurch beschleunigen (oder besser ausgedrückt: müssen weniger oft aufgerufen werden), dass man anhand der Translations- und Rotationsgeschwindigkeiten der einzelnen Körper abschätzt, nach welcher Zeit es frühestens zu einer Kollision kommen kann. Ein recht effizienter Algorithmus, der subquadratisch in der Abhängigkeit der Polygonanzahl n ist und von konstanten Translations- und Rotationsgeschwindigkeiten der Körper ausgeht, ist in der Arbeit von Schömer [34] beschrieben. Leider lässt sich dieses Verfahren nicht für eine allgemeine Mehrkörpersimulation anwenden, denn die Geschwindigkeiten von Körpern die mit Gelenken verbunden sind ändern sich permanent. Für Körper, die nicht mit Gelenken verbunden sind, wäre dies jedoch eine Möglichkeit gewisse Kollisionstests zu deaktivieren bis der entsprechende Kollisionszeitpunkt eingetreten ist.

Damit neben dem Punktpaar mit dem kürzesten Abstand noch weitere Punktpaare gefunden werden können, müssen die Körper konvex sein oder aus konvexen Teilstücken bestehen, denn nur so kann ein effizienter Algorithmus implementiert werden. Für Körper ohne spezielle Topologie wäre der Aufwand zur Lokalisierung solch zusätzlicher Punktpaare viel zu hoch als dass er eine praktische Anwendung fände. Ein Körper ist genau dann konvex, wenn sich von jedem beliebigen Punkt in seinem Innern jeder Punkt der Außenfläche sehen lässt. Nachfolgend finden sich ein paar einfache Beispiele für zweidimensionale konvexe bzw. nicht konvexe Objekte.

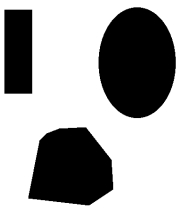


Abbildung 5.3: Konvex

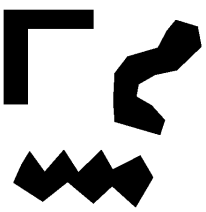


Abbildung 5.4: Nicht konvex

Ein PKW ist natürlich alles andere als ein konvexer Körper. Aus diesem Grund werden die 3D-Modelle zuerst in konvexe Teilstücke zerlegt und für jedes einzelne Teilstück wird dann eine

Abstandsbestimmung zu den jeweils anderen Körpern durchgeführt. An diesem Punkt nutzt man natürlich aus, dass die 3D-Modelle in der Simulation nicht zu *exotisch* sind und sich in eine vernünftige Anzahl konvexer Teilstücke zerlegen lassen. Objekte mit torusähnlicher Form wären für diesen Ansatz schon nicht mehr geeignet und es müsste aus Performancegründen ein anderer Algorithmus gewählt werden. Am Beispiel des 3D-Modells der Mercedes A-Klasse soll dies verdeutlicht werden. Das Modell besteht aus insgesamt 40000 Dreiecken und bildet einen Mercedes A-Klasse PKW sehr genau ab. Das nachfolgende Bild der Frontpartie verdeutlicht den Detailierungsgrad.

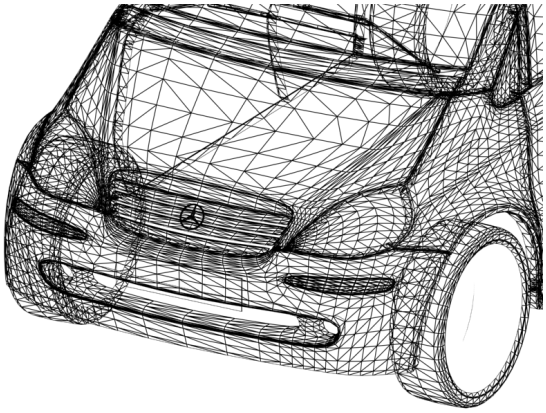


Abbildung 5.5: 3D Gittermodell der Mercedes A-Klasse Frontpartie

Vor dem Simulationsstart wird das 3D Modell eingelesen und in seine konvexen Teilstücke zerlegt. Das folgende Bild zeigt die Zerlegung.

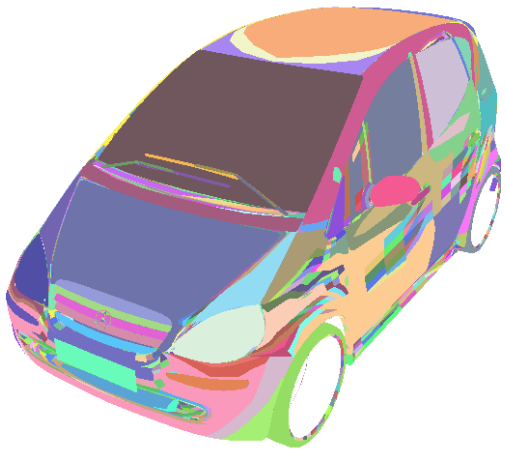


Abbildung 5.6: Konvexe Zerlegung eines 3D-Modells der Mercedes A-Klasse

Ingesamt erhält man bei diesem Modell dann ungefähr 2000 einzelne konvexe Teilstücke. Jedoch müssen nicht bei jedem Integrationsschritt alle Stücke berücksichtigt werden. Durch

einfache Abschätzung der maximalen relativen translatorischen und rotatorischen Geschwindigkeiten können in einer Vorstufe viele Paartests eliminiert werden.

5.1 Abstandsbestimmung konvexer Körper

Algorithmen, die für konvexe Körper den minimalen Abstand bestimmen und in Form eines Punktpaars zurückliefern, gibt es in verschiedensten Varianten. Bei der Wahl eines geeigneten Verfahrens sollten neben seiner Geschwindigkeit auch die Robustheit stehen. Es gibt generell zwei Fälle, in denen alle Algorithmen versagen können. Zum einen ist das der Durchdringungsfall, wenn sich zwei Polyhedra überschneiden, zum anderen wenn sehr hoch polygonale Objekte verwendet werden. Gerade bei Polyhedra mit sehr vielen Polygonen, ab 10000 und mehr, darf die begrenzte Fließkommagenauigkeit der Rechner nicht vernachlässigt werden, da diese Algorithmen sonst nicht mehr korrekt terminieren.

Die wichtigsten Algorithmen sind GJK [31] von Gilbert, Johnson und Keerthi und V-Clip [30] von Brian Mirtich. Andere neuere Algorithmen basieren hierbei entweder auf GJK oder V-Clip und sind mehr oder weniger nur eine Verbesserung der Grundalgorithmen. GJK und V-Clip verwenden hierbei zwei grundverschiedene mathematische Ansätze. GJK bildet die Minkowski Differenz der beiden Polyhedra und sucht dann den Punkt mit dem minimalsten Abstand zum Ursprung. Die Minkowski Differenz zweier konvexer Objekte $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^3$ ist definiert durch

$$\boldsymbol{A} - \boldsymbol{B} = \{\boldsymbol{a} - \boldsymbol{b} \,|\, \boldsymbol{a} \in \boldsymbol{A} \wedge \boldsymbol{b} \in \boldsymbol{B}\}$$

Die Minkowski Differenz ist wiederum konvex. Der minimale Abstand zwischen $\boldsymbol{A}$ und $\boldsymbol{B}$ ist nun direkt durch die Minkowski Differenz gegeben

$$d(\boldsymbol{A}, \boldsymbol{B}) = min\{||\boldsymbol{x}|| \,|\, \boldsymbol{x} \in (\boldsymbol{A} - \boldsymbol{B})\}$$

Die Minkowski Differenz wird hierbei nicht explizit berechnet, sondern GJK verwendet eine Näherung und arbeitet sich iterativ an dieses Minimum heran bis eine gewisse Genauigkeitsforderung erfüllt ist. V-Clip verwendet einen grundsätzlich anderen Weg und ist auch wesentlich anschaulicher. V-Clip zerlegt ein Polyhedra in Vertizes, Kanten und Flächen und wandert dann über die Oberfläche zu dem Punktpaar mit dem kürzesten Abstand.

Im Hinblick auf die Erweiterung zu mehr als nur einem Punktpaar fiel die Wahl auf V-Clip [30] von Brian Mirtich als Basisalgorithmus. V-Clip bietet sich deshalb besonders an weil bestimmte Berechnungen, die zur Erweiterung auf mehrere Punktpaare nötig sind, schon als *Abfallprodukt* vorhanden sind. Der Vorteil von V-Clip liegt weiterhin darin, dass der Überlappungsfall zuverlässig detektiert wird, wenn sich 2 Polyhedra durchdringen und in der Robustheit besonders bei Polyhedra mit vielen Polygonen (ab 10000 Flächen) [2].

[2] V-Clip muß hier allerdings erst um einen Loop Test ergänzt werden, da der Algorithmus sonst nicht terminiert.

Im folgenden wird die Grundidee der sogenannten *Feature* basierten Algorithmen besprochen, zu denen auch V-Clip gehört, die für einen konvexen, geschlossenen, 3-dimensionalen Polygonkörper (kurz Polyhedra) das Punktpaar mit dem kleinsten Abstand bestimmen.

Das Polyhedra wird zuerst in seine sogenannten Features zerlegt, wobei ein Feature entweder der Vertex (V), die Kante (K) oder die Fläche (F) eines Polygons ist. Zu jedem Feature werden die Voronoigebiete und Voronoiebenen bestimmt, die folgendermaßen definiert sind

Definition *Die Menge aller Punkte außerhalb des Polyhedra, die zu dem Feature X den kürzesten Abstand haben, nennt man das Voronoigebiet VR(X). Die Voronoiebene VP(X,Y) zwischen zwei benachbarten Features X und Y ist die Schnittmenge der einzelnen Voronoigebiete, also* $VP(X,Y) = VR(X) \cap VR(Y)$.

Die Gesamtmenge aller Voronoigebiete *VR* füllt dann den gesamten Außenraum des Polyhedra aus. Die benachbarten Features eines Vertex sind hierbei die unmittelbar an den Vertex angrenzenden Kanten, die Nachbarfeatures einer Kante sind die angrenzenden Vertices und Flächen und die benachbarten Features einer Fläche sind die die Fläche begrenzenden Kanten.

Die folgende Grafik verdeutlicht dies für einen einfach Körper. In der Grafik sind exemplarisch die Voronoiebenen für einen Vertex, eine Kante und eine Fläche eingezeichnet. Der Raum, den die Voronoiebenen begrenzen, ist das entsprechende Voronoigebiet.

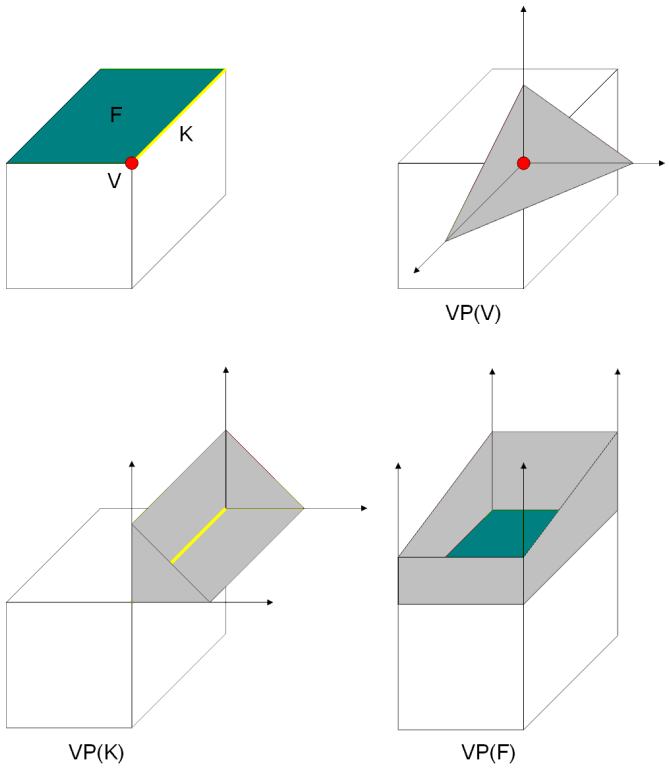


Abbildung 5.7: Zerlegung in Voronoigebiete

Die Grundidee des Algorithms beruht nun auf dem folgenden einfachen Theorem (ein Beweis findet sich zum Beispiel in [30]):

Theorem 1 *Seien X und Y Features zwei sich nicht durchdringender konvexer Polyhedra, und seien x ∈ X und y ∈ Y das Punktepaar der beiden Features X und Y mit dem kürzesten Abstand zueinander, dann ist der Abstand von x zu y der global kleinste Abstand dieser zwei Polyhedra, wenn gilt, x ∈ VR(Y) und y ∈ VR(X).*

Sowohl das Punktpaar (x, y) als auch das Featurepaar (X, Y) müssen hierbei nicht eindeutig sein. Für einen Würfel, der flach auf einem Tisch aufliegt, gibt es sogar unendlich viele Punktepaare, für die das Theorem gültig ist. Es wird hier lediglich sichergestellt, dass es sich um den global kleinsten Abstand handelt, denn der ist ja von entscheidender Bedeutung für die korrekte physikalische Simulation. Der Algorithmus führt eine iterative Suche nach dem Featurepaar (X, Y) solange durch, bis das obige Theorem zutrifft. Ist es für ein Featurepaar noch nicht erfüllt, so wird ein neues Paar bestimmt und die Suche solange fortgesetzt bis das Theorem erfüllt ist. Bei der Wahl eines neuen Featurepaares wird derart vorgegangen, dass der Abstand des neuen Featurepaares zum vorherigen Featurepaar stets kleiner oder gleich ist. V-Clip terminiert in einem von fünf möglichen Zuständen, V-V, V-K, V-F,K-K und K-F, wobei V-Clip nur mit einem K-F (Kante-Fläche) Featurepaar terminiert, wenn sich die Polyhedra durchdringen. Für die übrigen vier Featurepaare wird dann das Punktpaar mit dem kürzesten Abstand bestimmt.

Beim allerersten Aufruf startet V-Clip mit einem beliebigen Featurepaar und von dort *wandert* man dann bis zu dem Feature mit dem kürzesten Abstand. Die nachfolgenden zwei Grafiken sollen dieses Feature *wandern* verdeutlichen. Als Testobjekte wurden zwei identische Kugeln mit 3500 Polygonen und zwei verschiedenen Orientierungen gewählt. Das Start-Featurepaar ist in beiden Fällen dasselbe.

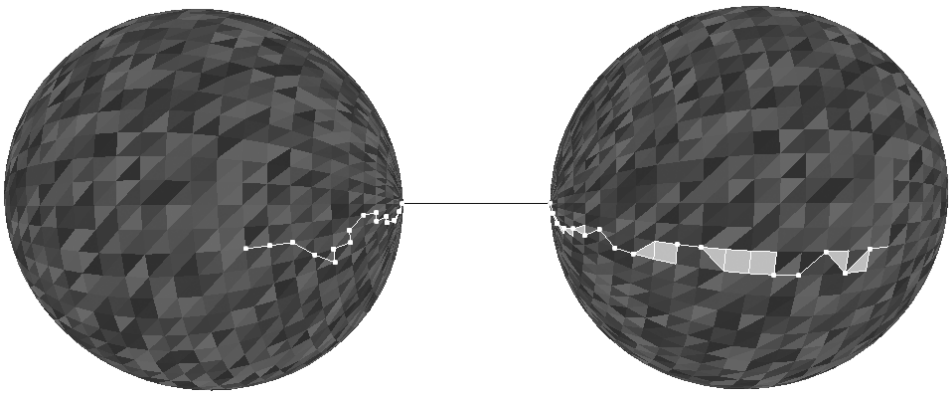


Abbildung 5.8: Verdeutlichung des Feature *Wandern*

In diesem ersten Fall wurden insgesamt 88 Featurepaare durchlaufen, bis der kürzeste Abstand lokalisiert war.

Hier noch ein weiteres Beispiel mit dem gleichen Start-Featurepaar und lediglich unterschiedlicher Orientierung der Kugeln zueinander. Diesmal wurden insgesamt 154 Featurepaare durchlaufen, da das Start-Featurepaar bei der linken Kugel genau auf der gegenüberliegenden Seite lag.

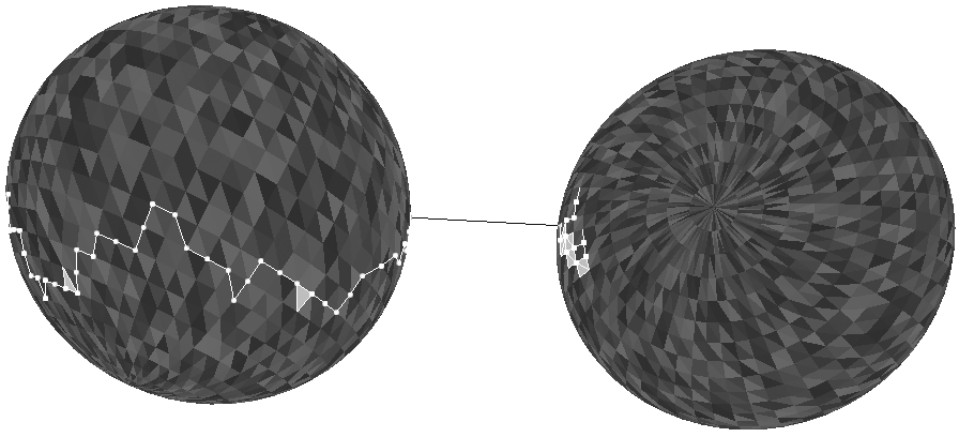


Abbildung 5.9: Weiteres Beispiel mit unterschiedlicher Orientierung der Kugeln zueinander

154 Featurepaare ist auch schon nahe an dem *worst case* Fall, der bei dem Detailgrad dieser zwei Modelle zu erwarten ist. Der schlechteste Fall wäre der, dass die Start-Featurepaare jeweils auf der dem kürzesten Featurepaar gegenüber gelegenen Seite liegen und erst einmal um die ganze Kugel gewandert werden muß. Hinsichtlich der Komplexität der Modelle von 3500 Polygonen ist dies aber ein sehr schneller Algorithmus. Als groben Richtwert für die maximal zu erwartende Anzahl von Iterationen kann man bei hinreichend gleichmässiger Flächenverteilung folgende Abschätzung machen

$$\text{Maximale Anzahl an Iterationen} \approx 2 \cdot (\sqrt{n_1} + \sqrt{n_2})$$

Hierbei sind n_1 und n_2 die Anzahl der Polygone des ersten bzw. des zweiten Körpers. Selbst bei sehr hochaufgelösten Körpern mit bis zu 50000 Polygonen ist die maximale Anzahl an Iterationen nicht größer als 1000. Der *worst case* tritt aber so gut wie nur einmal auf (zum Beispiel bei der Initialisierung), wenn man sich die Bewegungskohärenz zunutze macht und die Körper sich zwischen zwei sukzessiven Aufrufen zum Kollisionsalgorithmus nicht zu schnell gegeneinander verdrehen und verschieben.

Um die Bewegungskohärenz auszunutzen, wird das zuletzt bestimmte Featurepaar sukzessive bei den weiteren Abstandsbestimmungen während der Simulation verwendet. Der maximale Zeitschritt, mit dem die biomechanische Simulation berechnet wird, beträgt typischerweise $1ms$. Selbst bei Objekten, die sich mit 1000 U/min drehen würden, hätte man so lediglich eine maximale Winkeländerung von 6 Grad pro Zeitschritt, das heißt V-Clip muß bei sukzessiven Abstandbestimmungen nur sehr kurze Wege auf den Polyhedra zurücklegen bis das kürzeste Featurepaar gefunden wird. Auch bei den relativen Translationsgeschwindigkeiten zueinander

werden nie mehr als 100 m/s erreicht, was in einer maximalen Verschiebung pro Zeitschritt von 10cm resultiert. Hieraus erklärt sich auch die sehr hohe Geschwindigkeit, die von der Komplexität der Polyhedra so gut wie unabhängig ist.

Der maximal zu verwendende Zeitschritt muß mit Sorgfalt gewählt werden und natürlich mit den maximal zu erwartenden relativen Translations- und Winkelgeschwindigkeiten verknüpft sein, da es sonst vorkommen kann, dass Kollisionen unentdeckt bleiben.

Als Beispiel sei eine unendlich ausgedehnte Platte mit eine Dicke von 1 cm gegeben, auf die sich eine Kugel mit einem Durchmesser von 1 cm mit einer senkrecht zur Platte gerichteten Geschwindigkeit von 60 m/s zubewegt. Ab einer Distanz der Kugel zur Platte von 1 cm und weniger soll eine repulsive Kraft wirken.

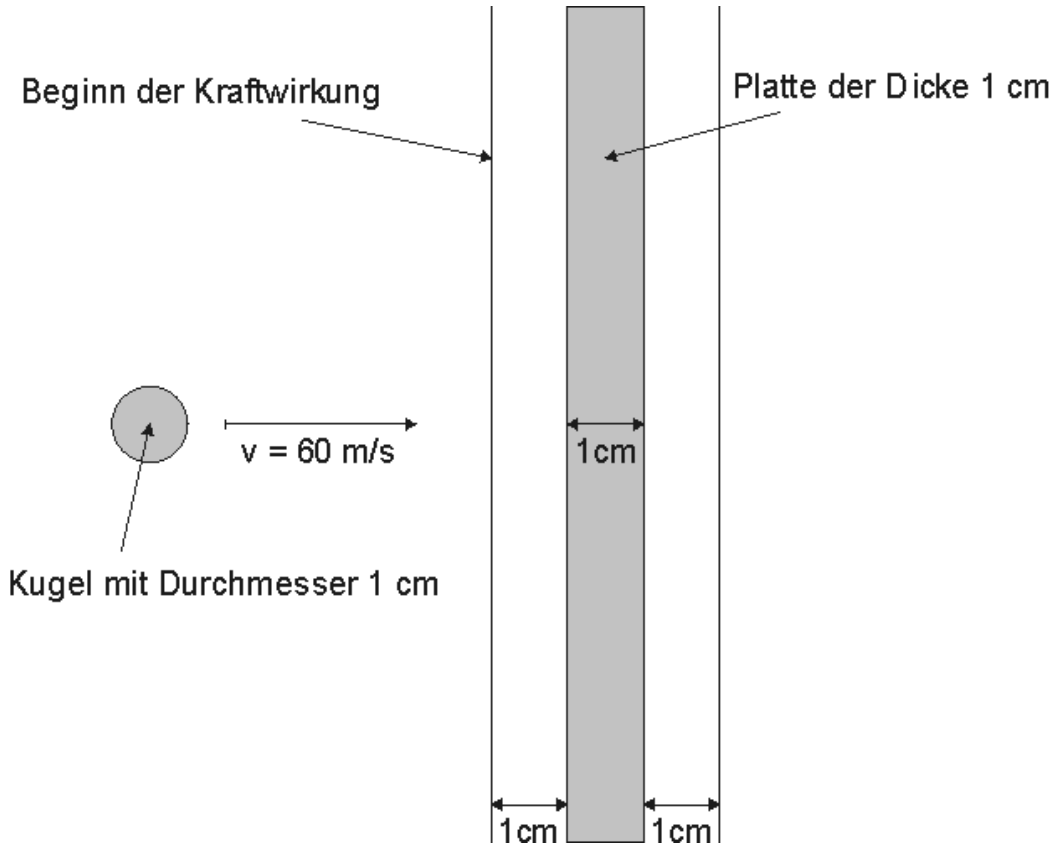


Abbildung 5.10: Konfiguration, in der es bei falsch gewählten maximalen Zeitschritten (für $\Delta t \geq 1ms$) zu einer Durchdringung der Platte kommen kann. Um dies zu verhindern muß die maximale Schrittweite unter Berücksichtigung der maximal zu erwartenden Geschwindigkeit kleiner als 1 cm sein.

Bei einem maximalen Zeitschritt von 1 ms für das Integrationsverfahren beträgt die Distanz pro Zeitschritt also 6 cm. Das heißt es können Fälle eintreten, in denen die Kugel aufgrund des zu großen Zeitschritts durch die Platte fliegt und keinen Rückstoß erfährt.

Die beiden folgenden Grafiken sollen dies verdeutlichen.

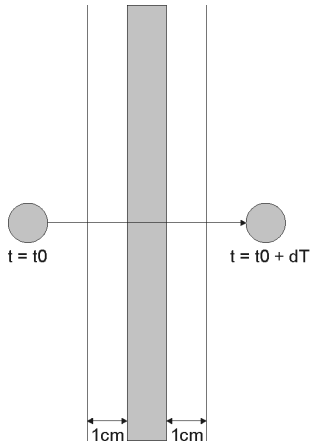


Abbildung 5.11: Bei $\Delta t = 1$ms wird in diesem Ausgangsfall die Platte übersehen.

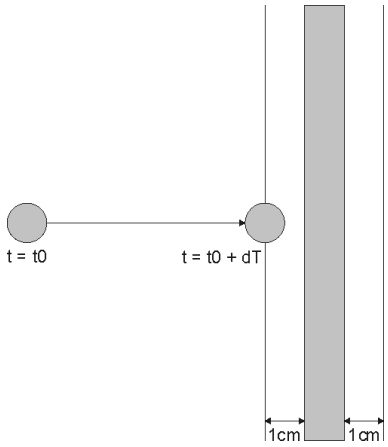


Abbildung 5.12: Anfangssituation, bei der die Platte *gesehen* wird; das Integrationsverfahren kann die Schrittweite entsprechend anpassen.

Um dieses Übersehen zu verhindern, muß der maximale Zeitschritt derart gewählt werden, dass die Platte auch bei diesen Geschwindigkeiten (bzw. der maximal zur erwarteten Geschwindigkeit) registriert wird, das heißt $\Delta t_{max} \leq 0.1$ ms.

Während der Implementierung des V-Clip Algorithmus kam es manchmal zu Situationen, in denen V-Clip nicht terminierte. Es stellte sich heraus, dass gewisse Featurepaarsequenzen unendlich oft durchlaufen wurden. Deutlich sichtbar war dies bei sehr flachen Objekten mit vielen Polygonen, bei denen V-Clip teilweise in eine Periode mit 16 sich wiederholender Featurepaaren lief. Dies ist eine direkte Folge der begrenzten Fließkommagenauigkeit, mit der gerechnet wurde. V-Clip wurde deshalb um ein Programmstück ergänzt, das solch wiederholende Featuresequenzen detektiert.

5.2 Mehrpunktkollisionen

Der schnellste und robusteste Algorithmus ist nutzlos im Hinblick auf die Integrationsgeschwindigkeit und die Integrationsgenauigkeit, wenn bei der Simulation jeweils nur ein Kollisionspunktepaar berücksichtigt wird. An zwei Beispielen soll dies gezeigt werden.

Zur Verdeutlichung der Problematik wurde als erstes das folgende Szenario simuliert: Eine flache Platte mit der Masse 1 kg, der Länge 1 m, einer Breite von 0.5 m und einer Dicke von 0.01 m wurde aus einer Höhe von 1 m auf einen flachen Boden fallen gelassen. Die Platte hatte einen Anfangswinkel von 45 Grad. Die Reaktionskraft war derart gewählt, dass der Boden sehr stark dämpfte, so dass die Platte schnell zur Ruhe kam. Reibungskräfte wurden hierbei nicht berücksichtigt. Die Kraft und Dämpfung wurden bei einem Abstand von 1 cm von Platte zu Boden wirksam und wirkten linear mit der Distanz $F \sim d$ bzw. der relativen Geschwindigkeit $F \sim v_z$. Als Integrationsverfahren wurde RADAU5 verwendet, die relative Toleranz betrug $1e^{-10}$ und die absolute Toleranz $1e^{-6}$ und $1e^{-7}$. Das folgende Bild zeigt die Anfangs- und Endsituation

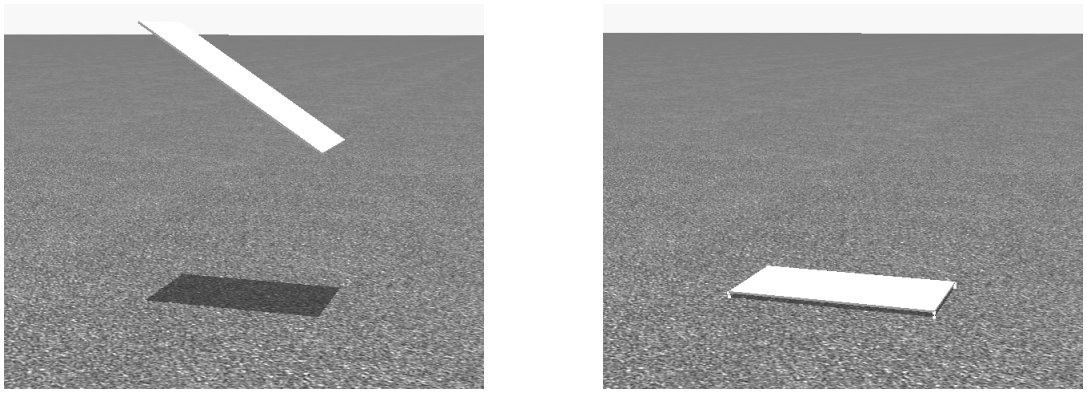


Abbildung 5.13: Simulation einer flachen Platte

Insgesamt wurde das System 4 Sekunden lang integriert, so dass am Ende der Simulation die Platte flach auf dem Boden lag. Zuerst wurde die Simulation unter Berücksichtigung nur eines Punktpaares integriert, also auch wenn die Platte schon flach am Boden liegt wird immer nur ein Punktpaar (abwechselnd einer der Eckpunkte mit dem kürzesten Abstand zum Boden) eine Kraft erzeugen, die die Platte über dem Boden hält. Im 2. Fall wurde das System derart integriert, dass alle Punkte, die näher als 1 cm zum Boden waren, mitberücksichtigt wurden, so dass an allen 4 Eckpunkten eine Kraft wirkt, wenn die Platte flach am Boden liegt.

Schon für diesen sehr einfachen Fall sind die Unterschiede in der Integrationszeit sehr deutlich. Viel wichtiger ist jedoch die Tatsache, dass bei zu großen Toleranzen (in diesem Fall $AbsTol > 1e^{-5}$) die Situation am Ende der Simulation mit nur einem Punktpaar nicht identisch mit der Mehrpunktmethode ist. Dies wird schnell ersichtlich wenn man sich überlegt, dass bei der Einpunktmethode stets Drehmomente auftreten, die versuchen die Platte in Rotation zu versetzen. Bei der Mehrpunktmethode sind diese Effekte erheblich geringer, insofern ist die

resultierende Bewegung anders, wenn die Toleranz nicht entsprechend klein genug gewählt wird. In schwach gedämpften Situationen sind diese Effekte besonders stark.

Um einen Vergleich beider Methoden zu erhalten, werden neben der reinen Simulationszeit auch die Anzahl der Integrationsschritte n_{st}, die Anzahl der Jacobimatrix Berechnungen n_{jac} sowie die Anzahl der Funktionsaufrufe n_{fcn} zur Bestimmung von $\dot{\boldsymbol{Y}}(t)$ aufgelistet.

	$AbsTol$	Zeit	n_{st}	n_{fcn}	n_{jac}
Nur ein Kollisionspunkt	$1e-6$	1.28s	15402	384117	7874
Mehrere Kollisionspunkte	$1e-6$	0.57s	8730	161957	4316
Nur ein Kollisionspunkt	$1e-7$	2.15s	23306	592408	11052
Mehrere Kollisionspunkte	$1e-7$	0.97s	13235	271743	5957

Die Einpunktmethode benötigt ungefähr den doppelten Aufwand im Vergleich zur Mehrpunktmethode. Gerade in Phasen mit flächigen Kontakten muß bei der Einpunktmethode die Schrittweite sehr stark reduziert werden, um die geforderte absolute Toleranz einzuhalten.

Viel stärker ist der Unterschied jedoch bei komplexeren Mehrkörpersystemen mit mehreren Gelenken. Zur Verdeutlichung wurde als weiteres Beispiel ein anfangs stehender Mensch genommen, der kraftlos (also ohne Muskelaktivität) in sich zusammensackt. Auf die genaue Implementierung des Menschmodells wird in Kapitel 6 eingegangen. Der Bodenkontakt simuliert eine glatte Fläche ohne Reibung. Dieses System wurde für eine Sekunde integriert und jeweils einmal mit der Einpunkt- und einmal mit der Mehrpunktmethode simuliert. Als relative Toleranz wurde durchgehend $RelTol = 1e^{-11}$ gewählt.

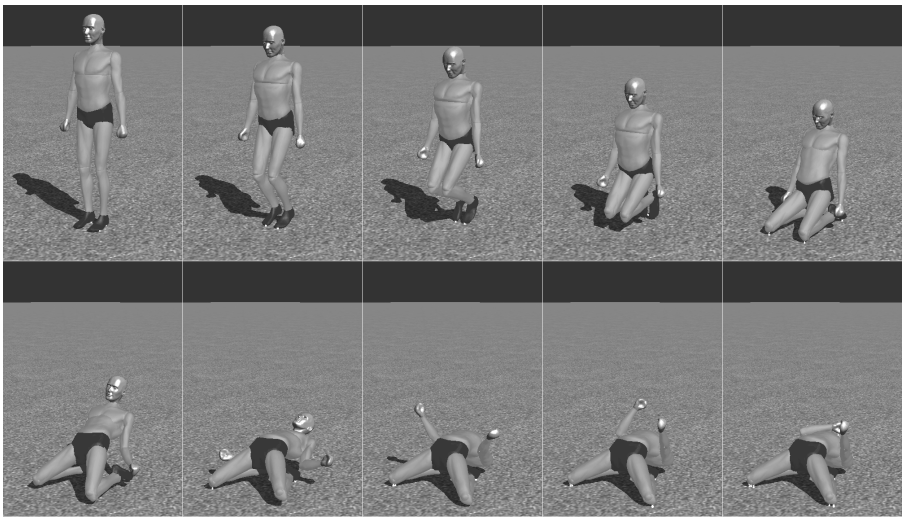


Abbildung 5.14: Simulation eines kraftlos in sich zusammensackenden Menschen

	$AbsTol$	Zeit	n_{st}	n_{fcn}	n_{jac}
Nur ein Kollisionspunkt	$1e-5$	65s	8458	1191260	4969
Mehrere Kollisionspunkte	$1e-5$	21s	2795	396664	1703
Nur ein Kollisionspunkt	$1e-6$	215s	25705	4330633	18469
Mehrere Kollisionspunkte	$1e-6$	26s	3222	485217	2074

Gerade in der Standphase, wenn der Fuß noch vollen Kontakt zum Boden hat, ist die Methode mit nur einem Kollisionspunkt erheblich langsamer. Die unterschiedlichen Kollisionskräfte und Drehmomente erzwingen eine Reduzierung der Schrittweite, um die gleiche Genauigkeit zu erreichen wie bei der Mehrpunktmethode. Phasen in denen solche flächigen Kontakte nicht auftreten, werden gleichschnell integriert. Wichtiger jedoch ist die Tatsache, dass die Endstellung des Menschen in beiden Fällen nicht die gleiche ist.

Abbildung 5.15: Mehrpunktmethode

Abbildung 5.16: Einpunktmethode

Deutlich sichtbar ist die unterschiedliche Endstellung der Arme. Was auf den Bildern nicht so gut zu erkennen ist, ist die unterschiedliche Position der Knie und eine leicht gedrehte Stellung.

Aus Performancegründen ist es natürlich nicht möglich wirklich alle Punkte die innerhalb der Kollisionsdistanz liegen zu berücksichtigen. Bei hoch polygonalen Modellen wäre dieser Ansatz völlig unpraktisch und, wie sich im Fall des auf den Boden fallenden Menschen gezeigt hat, auch unnötig. Versuche haben ergeben, dass es ausreicht nur die Features in der unmittelbaren Umgebung des kürzesten Featurepaares auf Kontakt zu überprüfen.

Vom kürzesten Featurepaar ausgehend werden alle Nachbarflächenfeatures beider Körper lokalisiert und dann gegeneinander auf Kontakt überprüft. Die folgende Grafik soll dies im Zweidimensionalen verdeutlichen.

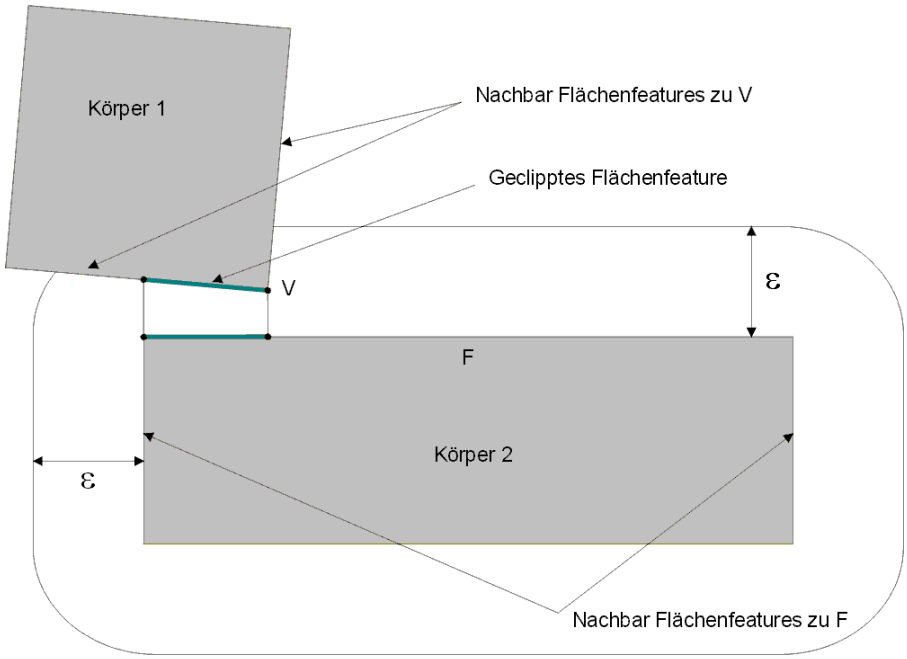


Abbildung 5.17: Mehr als nur ein Punktepaar innerhalb der Kontaktdistanz

Die Kontaktdistanz ab der Kräfte wirksam werden ist hier mit ϵ gekennzeichnet. Das Featurepaar mit dem kürzesten Abstand zueinander ist das Vertexfeature V von Körper 1 und das Flächenfeature F von Körper 2. Nachdem dieses Featurepaar lokalisiert wurde, werden alle Nachbarflächenfeatures zu V und alle Nachbarflächenfeatures zu F (sowie F selbst) gegeneinander geprüft. In diesem 2 dimensionalen Beispiel sind das 2 Flächenfeatures für Körper 1 und 3 Flächenfeatures für Körper 2.

Um festzustellen ob 2 Flächenfeatures F_1 und F_2 innerhalb der Kontaktdistanz ϵ sind, wird eine Fläche gegen das andere Flächenfeature geclippt. Dieser Clipvorgang bestimmt die Schnittmenge von F_1 mit dem Voronoigebiet des anderen Flächenfeatures $VR(F_2)$. Als Ergebnis erhält man stets ein Polygon oder eine leere Menge, wenn F_1 nicht in $VR(F_2)$ enthalten ist. Nun werden die Distanzen aller Eckpunkte des Polygons zum Flächenfeature F_2 berechnet und überprüft ob sie innerhalb von ϵ liegen.

Statt alle Punkte der geclippten Fläche auf Kontakt zu überprüfen, kann auch der Schwerpunkt dieses Polygons berechnet werden. Der Schwerpunkt eines Polygons G ist gegeben durch das folgende Flächenintegral

$$\boldsymbol{R} = \int_G \boldsymbol{x} d\boldsymbol{x}, \quad \boldsymbol{R}, \boldsymbol{x} \in \mathbb{R}^3$$

Dieses Integral für jeden Zeitschritt im 3-dimensionalen auszuwerten wäre viel zu aufwändig. Deshalb wird das Polygon zuerst in die Ebene projeziert. Wenn $\boldsymbol{n}_x$ und $\boldsymbol{n}_y$ zwei zueinander orthogonale Normalenvektoren sind, die in der Polygonebene liegen und wenn $\boldsymbol{p}$ ein Punkt ist, der innerhalb der Polygonebene liegt, dann ist die Projektion gegeben durch

$$\boldsymbol{p}_{2D} = \begin{pmatrix} p_{2D_x} \\ p_{2D_y} \end{pmatrix} = \begin{pmatrix} \boldsymbol{p} \cdot \boldsymbol{n}_x \\ \boldsymbol{p} \cdot \boldsymbol{n}_y \end{pmatrix}$$

Der Schwerpunkt in zwei Dimensionen ergibt sich analog

$$\boldsymbol{R}_{2D} = \int_{G_{2D}} (\boldsymbol{p}_{2D}) \boldsymbol{dF}$$

Mit Hilfe des Stoke'schen Integralsatzes für die Ebene kann das Integral weiter vereinfacht werden. Der Satz von Stoke lautet

$$\int_F (\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y}) dxdy = \int_{\partial F} (Pdx + Qdy) \tag{5.1}$$

Das Flächenintegral wird also in 2 Linienintegrale über den Rand des Polygons gewandelt. Mit der Wahl von

$$\begin{pmatrix} P \\ Q \end{pmatrix} = \begin{pmatrix} -xy \\ 0 \end{pmatrix} \quad \text{und} \quad \begin{pmatrix} P \\ Q \end{pmatrix} = \begin{pmatrix} 0 \\ xy \end{pmatrix}$$

lassen sich die x- bzw. y-Komponente des Schwerpunkts bestimmen, so dass man erhält

$$R_{2D_x} = \int_{\partial F} (-xy) dx \quad \text{und} \quad R_{2D_y} = \int_{\partial F} (xy) dy$$

Dieser 2D-Punkt muß nun wieder zurück in die Ebene des ursprünglichen Polygons projeziert werden

$$\boldsymbol{R} = \begin{pmatrix} nx_x \cdot R_{2D_x} + ny_x \cdot R_{2D_y} \\ nx_y \cdot R_{2D_x} + ny_y \cdot R_{2D_y} \\ nx_z \cdot R_{2D_x} + ny_z \cdot R_{2D_y} \end{pmatrix} + d \cdot \boldsymbol{n}$$

$\boldsymbol{n}$ ist hierbei der Normalenvektor und d der Abstand des Polygons.

Kapitel 6

Menschmodell

Die Implementierung des eigentlichen Menschmodells ist die größte Herausforderung bei der Realisierung einer biomechanischen Simulation mit dem Computer. Betrachtet man die Komplexität des menschlichen Körpers, so können alle Modelle natürlich nur zu einem gewissen Grad den echten Menschen abbilden ohne dass man sich in einem nicht zu verifizierbaren Parameterdschungel verirrt. Höherparametrische Modelle machen erst dann Sinn, wenn gewisse mechanische Parameter des Menschen besser bekannt sind. Die Implementierung des Menschmodells in dieser Arbeit lehnt sich deshalb eng an schon bestehende Modelle an, so dass auch ein direkter Vergleich mit anderen Arbeiten möglich ist. Es werden jedoch wichtige Neuerungen im Bereich der Gelenkmodellierung vorgestellt, die zu einer wesentlich realeren Bewegung führen. Mit welchen Parametern lässt sich nun ein Mensch beschreiben?

Zum einen müssen die anthropometrischen Daten des Menschen berücksichtigt werden, also die Abmessungen, Massen, Schwerpunktpositionen sowie Trägheitstensoren jedes einzelnen Körperteils. Für diese Daten existieren sehr fundierte Arbeiten von verschiedensten Quellen, die für den durchschnittlichen Menschen des Westens ihre Gültigkeit besitzen. Das umfassenste Werk hierzu stammt von der Nasa [15]. An der Universität Tübingen wurde mit Hilfe der NASA Daten das Programm CALCMAN3D entwickelt, das mittels Regressionsgleichungen erlaubt aus einfachen Eingabedaten wie Körpergröße, Gewicht und Geschlecht die anthropometrischen Daten für einen beliebigen Menschen zu berechnen. Will man sich jedoch nicht nur auf eine Quelle verlassen dann bietet die Arbeit von Jørgen Bjørnstrup [13] einen fundierten Überblick über alle auf diesem Bereich existierenden Messungen. Sehr gut ist in dieser Arbeit die Fehlerabschätzung der verschiedenen Methoden, die auf eine starke Schwankung der Messergebnisse sowie der anthropometrischen Daten beim Menschen hindeutet. Typische Schwankungen zwischen zwei vergleichbaren Individuen können bis zu 20% betragen.

Anders sieht es allerdings bei der Implementierung der menschlichen Gelenke aus, die erheblich aufwändiger und komplexer zu realisieren sind. Aufgrund der hohen Komplexität der Gelenke lässt sich ein exaktes Abbild der menschlichen Anatomie auf heutigen Rechnern noch nicht effizient realisieren, denn um ein 100-prozentiges Abbild zu erhalten müssten wirklich alle Knochen, Muskeln und Bänder nachgebildet werden. Dieser extreme Schritt ist aber keineswegs nötig, denn man kann zeigen, dass in den meisten biomechanischen Simulationen eine solch exakte Gelenkstruktur nicht erforderlich ist.

Um den Vergleich mit bestehenden Arbeiten zu ermöglichen und um vor allem den Parameterbereich in einem vernünftigen Rahmen zu halten, werden alle Gelenke in dieser Arbeit deshalb als ideale Kugel- oder Scharniergelenke mittels Feder/Dämpfer Elementen realisiert. Um diese Idealisierung zu erreichen, müssen die Federn dementsprechend steif gewählt werden.

Bei der Implementierung der Gelenke sind drei Aspekte zu berücksichtigen: zum einen muß ein Gelenk durch Gelenkanschläge in seiner Bewegungsfreiheit derart eingeschränkt werden, dass keine unanatomischen Bewegungen ausgeführt werden können; desweiteren müssen der Gelenkwiderstand und das Gelenkmoment (hervorgerufen durch die aktivierten Muskeln) berücksichtigt werden, die die Bewegung eines Gelenks innerhalb seiner Bewegungsfreiheit in der Bewegung hemmen bzw. aktiv steuern. Hierbei ist zu erwähnen, dass der Bewegungsraum der durch aktive Gelenkmomente erreicht werden kann in der Regel kleiner ist als der Raum, der durch die Gelenkanschläge festgelegt ist. Auf alle drei Punkte wird im folgenden genauer eingegangen. Wie exakt die Implementierung der aktiven Bewegungen im Endeffekt geschieht hängt stark vom Anwendungsfall ab. Ist man an den genauen Aktivitäten der beim Gehen verantwortlichen Muskeln interessiert, so müssen selbstverständlich alle Muskeln dieses Bewegungsapparates berücksichtigt werden. Kommt es hingegen nur auf die reine Bewegung und die in den Gelenken auftretenden Momente an, so reicht es alle für ein Gelenk zuständigen Muskeln über ein kummuliertes Gelenkmoment zu realisieren. Eine sehr detaillierte Behandlung des menschlichen Gehens findet sich zum Beispiel in der Arbeit von Michael Günther [38], die in der Biomechanik Abteilung der TAT an der Universität Tübingen verfasst wurde. Michael Günther simulierte hierbei alle Muskeln des Gehapparats einzeln und konnte somit ein synthetisches Modell des menschlichen Gehens simulieren, allerdings nur in 2 Dimensionen. Der Ansatz in dieser Arbeit verwendet hingegen zur Simulierung von Muskelkräften ein einzelnes Gelenkmoment.

Die Gelenkanschläge eines Körpergelenks sind je nach Gelenktyp und vor allem je nach Körperhaltung unterschiedlichen Ursprungs. Gelenkanschläge können zum einen durch die inneren Strukturen wie Bänder, Sehnen und Knochen hervorgerufen werden, aber auch extern in bestimten Körperstellungen durch Kontakt von Körperteilen untereinander wie zum Beispiel bei der Flexion, also der Beugung des Ellenbogens. Bei Menschen mit besonders ausgeprägtem Bizeps ist der Gelenkanschlag ausschließlich durch den Kontakt des Unterarms mit dem Bizeps gegeben, interne Strukturen des Ellenbogengelenks spielen hierbei fast keine Rolle. Anders sieht es wiederum bei der Extension, der Streckung des Ellenbogens aus, hier sind ausschließlich die Knochen und Bänder des Gelenks für den Anschlag verantwortlich. Erschwert wird die Implementierung zusätzlich dadurch, dass die Anschläge eines Gelenks auch von der unmittelbaren Stellung von Nachbargelenken beeinflußt werden können, wie am Beispiel des Knie- und Hüftgelenks in Abbildung 6.1 auf der folgenden Seite dargestellt ist.

Bei voll gestrecktem Knie beträgt der maximale Flexionswinkel der Hüfte 90 Grad. Je mehr das Kniegelenk nun gebeugt wird, desto größer wird auch die maximale Beugung der Hüfte. Bei maximal gebeugtem Kniegelenk (ca. 120 Grad) lässt sich das Hüftgelenk aktiv bis 120 Grad beugen. Eine weitere Steigerung dieses Bewegungsfreiraums ist beim Normalmenschen nicht mehr aktiv mit an diesen Gelenken beteiligten Muskeln möglich, sondern nur extern oder unter Zuhilfenahme anderer Muskeln. Dieser passive Winkel beträgt dann bis zu 145 Grad.

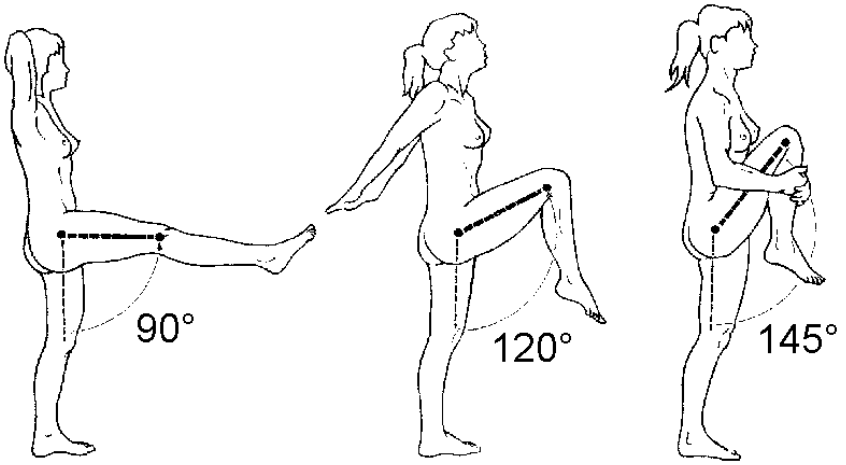


Abbildung 6.1: Gelenkanschläge der Hüfte in Abhängigkeit der Stellung des Kniegelenks. Die Gelenkanschläge bei 90 und 120 Grad sind aktiv, bei 145 Grad jedoch passiv. Aus Kapandji [39].

Die Ursache dieser gegenseitigen Abhängigkeit der Gelenkanschläge liegt in der ischiochoralen Muskulatur, die über 2 Gelenke verläuft. Bei der Kniestreckung wird diese Muskulatur gedehnt, deshalb wird bei einer Hüftbeugung der Gelenkanschlag reduziert.

Nicht zuletzt sind die Gelenkanschläge von Mensch zu Mensch einer sehr großen Streuung unterworfen, wie man am direkten Vergleich eines Zirkusartisten mit einem Büromenschen eindrucksvoll sehen kann. Es kann deshalb nicht genug betont werden, dass eine Sensitivitätsanalyse bei biomechanischen Simulationen und deren Interpretation von größter Bedeutung ist, um überhaupt vernünftige Aussagen treffen zu können. Analog zur Implementierung des Gelenktyps (Kugel- oder Scharniergelenk) werden auch die Gelenkanschläge durch Feder/Dämpfer Elemente realisiert, die *aktiv* werden sobald sich ein Gelenk außerhalb seines Bewegungsbereichs befindet. Hierbei kann über die Härte der verwendeten Feder/Dämpfer die Anschlagshärte eines Gelenks individuell eingestellt werden.

Erschwert wird die Implementierung von Gelenkanschlägen vor allem durch das Fehlen zuverlässiger Messdaten, denn während die Kenntnis von absoluten Gelenkanschlägen aufgrund von sehr detaillierten Standardwerken, wie von Kapandji [39], gut dokumentiert vorliegt, gibt es nahezu keine Daten über die genaue *Härte* und *Dämpfung* eines Anschlags. Vor allem der Anteil von dissipativen und elastischen Beiträgen ist nur sehr ungenau bekannt.

Eine weiterer wichtiger Aspekt ist die Modellierung der Gelenkwiderstände, die gerade bei stark passiven Simulationen, wie einer Fußgänger-PKW Unfallsimulation, von Bedeutung sind und in der aktive Muskelkräfte in den meisten Fällen keine Rolle spielen. Die Gelenkwiderstände teilen sich auf in elastische und in dissipative Anteile. Für die elastischen Widerstände sind die (gestreckten) Muskeln, Bänder und Sehnen verantwortlich. Die dissipativen Elemente werden hauptsächlich durch Reibungsanteile, also die Reibung von Muskeln, Sehnen und Bändern untereinander und am Gewebe hervorgerufen. Die Messung der genauen Dämpfungskonstanten

erweist sich als recht schwierig und äußerst fehleranfällig. Einfache Experimente am Knie- und Schultergelenk versuchen durch Pendelversuche auf die Dämpfungskonstanten zu schließen. Dabei stellte sich heraus, dass bei allen Gelenken nach ungefähr 5 Vollschwingungen die Schwingung abgeklungen ist und eine Gesamtschwingungsdauer von ca. 6 bis 7 Sekunden auftrat. Einen guten Fit dieser Kurven erhält man durch einen linearen Ansatz für das Drehmoment in Abhängigkeit der Winkelgeschwindigkeit.

$$T_{Reibung} = D \cdot \dot{\phi}$$

Hierbeit ist $\dot{\phi}$ die Winkelgeschwindigkeit in rad/s. Die Dissipationskonstante D ist je nach Gelenk im Bereich 0.15 bis 0.3. Eng verknüpft mit den Gelenkwiderständen ist die Implementierung von aktiven Muskelkräften. Zusammen mit den Gelenkanschlägen spielen die aktiven Muskelkräfte die zentrale Rolle bei Simulationen, die die Bewegung des Menschen bewußt simulieren und rekonstruieren, wie am Beispiel des Reckturners in Kapitel 7 gezeigt wird. Neben den absoluten Größen der Muskelkräfte ist es vor allem wichtig, den Kraft-Längen und den Kraft-Geschwindigkeits Zusammenhang eines Muskels zu berücksichtigen. Die Gruppe von Muskeln, die aktiv und bewußt vom Menschen gesteuert werden kann, nennt man die quergestreifte Skelettmuskulatur. Für diese Gruppe von Muskeln gelten sehr charakteristische Kraft-Längen und Kraft-Geschwindigkeits Zusammenhänge, die sich von den anderen Muskeltypen, zum Beispiel dem Herzmuskel, deutlich unterscheiden. Die hier dargestellten Kurven basieren auf der Arbeit von Zajac [40].

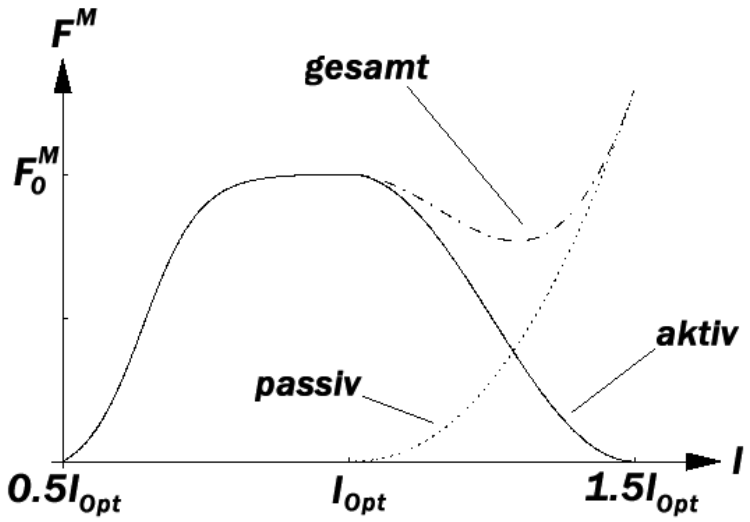


Abbildung 6.2: Kraft-Längen Abhängigkeit der Skelettmuskulatur bei isometrischer Kontraktion und maximaler Aktivierung. Nach Zajac [40].

Die durchgezogene Kurve stellt die vom Muskel bei maximaler Aktivierung $q = 1$ aktiv gelieferte, also durch den Menschen bewußt gesteuerte Kraft bei einer Kontraktion der Muskelfasern ohne Längenänderung (isometrische Kontraktion) dar. Bei geringeren Aktivierungen

$q < 1$ wird die Kurve linear nach unten skaliert. Diese Kraft ist bei der optimalen Muskelfaserlänge l_{opt} am größten und ist mit F_0^M gekennzeichnet. Jede Abweichung von der optimalen Länge l_{opt} resultiert in einer Abnahme der aktiven Kraft. Bei ca. dem 0.5-fachen bzw. dem 1.5-fachen der optimalen Länge l_{opt} geht die bereitgestellte Kraft gegen Null. Unabhängig von der Aktivierung der Muskeln ergibt sich die resultierende Gesamtkraft zusätzlich noch durch den Beitrag der Überstreckung des Muskelgewebes, die in einer passiven Kraft resultiert, wenn das Gewebe über die Länge von l_{opt} hinaus gestreckt wird. Diese Kräfte nehmen annähernd exponentiell zu bis hin zu $1.5 \cdot l_{opt}$; eine weitere Streckung des Gewebes führt unweigerlich zur Zerstörung der Muskelfasern.

Einen weiteren wichtigen Einfluß auf die resultierende Gesamtkraft hat bei sehr dynamischen Bewegungsabläufen auch die Geschwindigkeit der Muskelbewegung. Die folgende Abbildung bezieht sich auf einen voll aktivierten Muskel mit $q = 1$ bei seiner optimalen Muskelfaserlänge von l_{opt}.

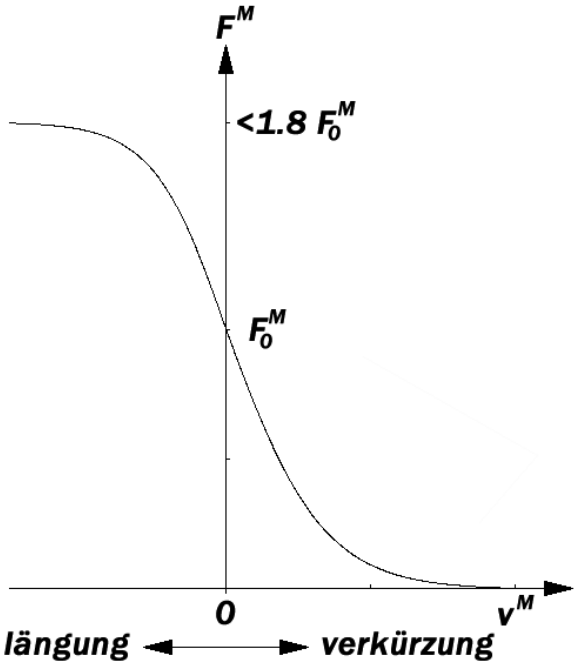


Abbildung 6.3: Kraft-Geschwindigkeits Zusammenhang ausgehend von der optimalen Muskelfaserlänge l_{opt}. Nach Zajac [40].

Die Kraft hängt in hohem Masse davon ab, ob der Muskel gelängt oder verkürzt wird und dies umso stärker je schneller diese Bewegung ausgeführt wird. Bei der Kontraktionsgeschwindigkeit Null ergibt sich genau die gleiche Kraft wie in Abbildung 6.2. Wird der Muskel nun *schnell* verkürzt so geht seine bereitgestellte Kraft sehr schnell gegen null (Typischerweise bei einer Geschwindigkeit von $1m/s$). Wird der Muskel jedoch getreckt, kann sich seine Kraft auf das 1.3- bis 1.8- fache seiner statischen Maximalkraft F_{max} erhöhen. Diese Kurve skaliert ebenfalls linear mit der Aktivierung der Muskeln.

Die Gelenkanschläge, Gelenkwiderstände und aktiven Muskelkräfte bilden zusammen mit den anthropometrischen Daten der einzelnen Körperteile dann das Menschmodell. Je nach Simulationstyp muß der Mensch jetzt noch in eine geeignete Anzahl von Elementen untergliedert werden. Hierbei wurde für alle Simulationen durchgehend ein 16-gliedriges Modell mit insgesamt 15 Gelenken implementiert. Man erhält somit folgenden Aufteilung:

- Kopf (Halswirbelsäule)
- Brust (Brustwirbelsaüle)
- Bauch (Lendenwirbelsäule)
- Becken (Kreuzbein)
- Linker/Rechter Oberschenkel
- Linker/Rechter Unterschenkel
- Linker/Rechter Fuß
- Linker/Rechter Oberarm
- Linker/Rechter Unterarm
- Linke/Rechte Hand

Die folgende Grafik verdeutlicht die Untergliederung anschaulich, unterschiedlich gefärbte Elemente sind jeweils eigenständige Körperteile.

Abbildung 6.4: Einteilung des Menschen in verschiedene Körperteile

Eine Erweiterung auf eine feinere Unterteilung ist jedoch ohne weiteres möglich. Für die in dieser Arbeit durchgeführten Simulationen reicht das 16-gliedrige Modell jedoch vollkommen aus.

6.1 Gelenkanschläge

Im folgenden wird auf die Realisierung von Gelenkanschlägen eingegangen und eine Auflistung über die Bewegungsfreiheiten und die Winkelbereiche aller Gelenke gegeben, die für dieses Menschmodell berücksichtigt werden.

6.1.1 Realisierung

Um einen zuverlässigen Gelenkanschlag zu realisieren, muß von der klassischen Methode der winkelabhängigen Implementierung Abstand genommen werden. Bei einfachen Gelenken mit nur einer Drehachse, also einem Scharniergelenk, ist dies ohne Einschränkung möglich. Hier lässt sich der Winkel zwischen zwei Gelenken schnell und vor allem eindeutig bestimmen und beim Überschreiten des Gelenkanschlags eine Kraft oder ein Drehmoment proportional der Winkelüberschreitung realisieren. Bei Kugelgelenken, bei denen auch eine Rotation erlaubt ist, treten jedoch unweigerlich Probleme auf, denn bei Rotationen um mehr als eine Achse ist die Angabe mit nur 3 Winkeln nicht mehr eindeutig.

Um diese Probleme zu umgehen, wurde ein mechanisch geometrischer Weg eingeschlagen, der eine elegante Methode ist, um Gelenkanschläge zu implementieren. Hierzu wird an dem Drehpunkt eines Gelenks eine Fläche aufgespannt, die dem Bewegungsraum des Gelenkes entspricht und mit der Orientierung des ersten Körpers transformiert wird. An den selben Punkt wird dann ein Vektor gesetzt, der mit der Orientierung des zweiten Körpers transformiert. Die folgende Grafik soll dies verdeutlichen. Hierbei ist die Form der verwendeten Fläche beliebig wählbar. Für die meisten Gelenke des Menschen bieten sich runde Kegelflächen mit elliptischer Grundfläche oder mit nur 4-seitiger Kegelgrundfläche an.

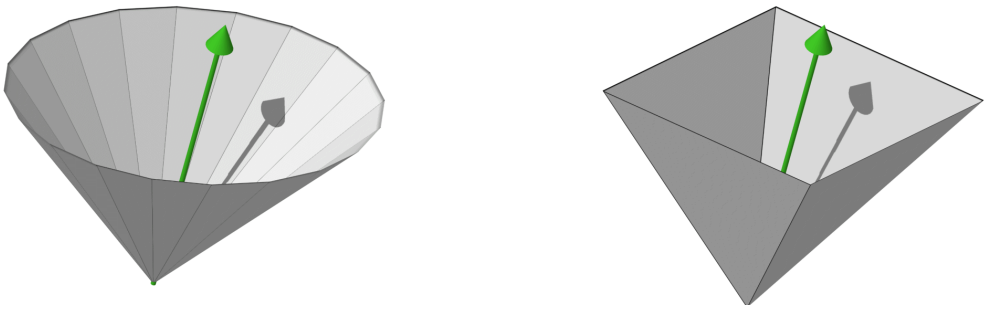


Abbildung 6.5: Verschiedene Möglichkeiten des geometrischen Gelenkanschlags

Befindet sich der Vektor außerhalb der Kegelfläche, wirkt auf beide Körper ein gleich großes jedoch entgegengesetztes Drehmoment, das proportional zum Abstand des Vektors von der Kegelfläche ist. Dieses Drehmoment versucht die Körper wieder innerhalb der Kegelflächenbegrenzung zurückzubewegen. Für alle Gelenke wurde hierbei ein lineares Kraftgesetz verwendet, das ein Drehmoment proportional zum Abstand des Vektors von der Kegelfläche wirken lässt.

6.1.2 Auflistung der Gelenkanschläge

Im folgenden werden die Gelenktypen und Gelenkanschläge für alle 15 Gelenke des hier verwendeten Modells besprochen. Bei der Einstellung der Gelenkanschläge diente das Buch von Ibrahim A. Kapandji [39] als Referenz. Alle Winkelangaben beziehen sich hierbei auf die Auslenkung aus der Nulllage. Die Nulllage ist ein stehender Mensch mit seitlich am Körper herabhängenden Armen und zum Körper hin gerichteten Handinnenflächen. Die Winkelbereiche sind, wo erforderlich, in aktive und passive Bereiche unterteilt. Die aktiven Bereiche sind die Winkel, die der Mensch mit eigener Muskelkraft erreichen kann. Die passiven Winkelanschläge sind immer etwas größer und können nur mit Fremdeinwirkung oder durch die Kraft anderer Muskelgruppen erreicht werden. Bei den Anschlägen werden immer nur die wichtigsten Bewegungen berücksichtigt, die in der Simulation auch eine Rolle spielen. So ist zum Beispiel beim Ellenbogen- und Handgelenk auf die rotatorischen Freiheitsgrade verzichtet worden, da sie bei den hier simulierten Fällen einen vernachlässigbaren Einfluß haben.

Hüft- und Kniegelenk spielen eine sehr zentrale Rolle bei biomechanischen Simulationen. Auf eine realitätsnahe Implementierung ist deshalb besonders zu achten. Das Kniegelenk wurde als Scharniergelenk, das Hüftgelenk als Kugelgelenk realisiert. Auf eine Rotation des Kniegelenks wurde hierbei verzichtet, die Ausschläge sind jedoch der Vollständigkeit halber mit aufgeführt.

Hüftgelenk	aktiv	passiv	Bemerkung
Extension (gestreckte Hüfte)	0	0-20	Bei gestrecktem Knie
Extension (gestreckte Hüfte)	20	20-40	Bei gebeugtem Knie
Flexion (gebeugte Hüfte)	90	> 90	gestrecktes Knie
Flexion (gebeugte Hüfte)	120	> 140	gebeugtes Knie. Passiver Anschlag bis an die Brust.
Abduktion	45		
Adduktion	30		Dieser extreme Winkel wird aber nie erreicht, da der Oberschenkel vorher an das andere Bein stößt.
Außenrotation	60		
Innenrotation	30-40		

Kniegelenk	aktiv	passiv	Bemerkung
Extension (Strecken)	0	5-10	
Flexion (Beugen)	120-140	-	Das Beugemass ist abhängig von der Stellung des Hüftgelenks.
Außenrotation	40		Rotationswinkel jeweils bei 90 Grad gebeugtem Knie.
Innenrotation	30		

Je nach Simulation kommt dem Sprunggelenk eine mehr oder weniger wichtige Rolle zu. Bei der Simulation des schwingenden Reckturners spielt das Sprunggelenk keine Rolle, jedoch ist

es von zentraler Bedeutung bei einer Fußgänger-PKW Unfallsimulation, solange der Mensch noch Kontakt mit dem Boden hat.

Sprunggelenk	aktiv	passiv	Bemerkung
Dorsal Flexion (Fuß anheben)	20 - 30	-	
Plantar Flexion (Fuß senken)	30 - 50	-	

Die exakte Realisierung des Schultergelenks ist sehr komplex, wie man am Beispiel der Umführbewegung (Zirkumduktion) des Arms gut erkennen kann. Der Bewegungsraum entspricht einem unregelmässig geformten Konus.

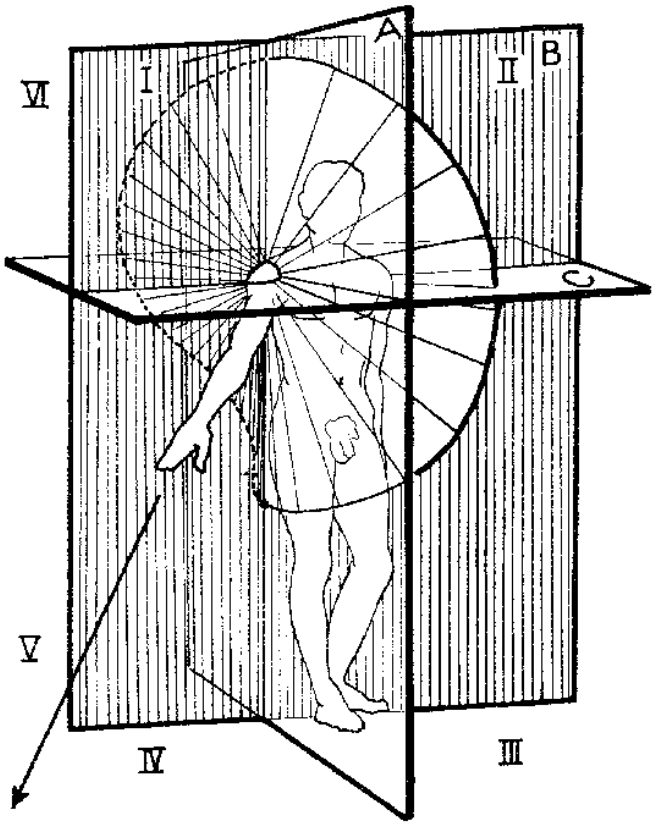


Abbildung 6.6: Bewegungsraum des Schultergelenks bei einer Zirkumduktion des Arms. Aus Kapandji [39].

Die Realisierung des Schultergelenks wird zusätzlich erschwert durch die unweigerliche Drehung um die Längsachse bei einer Bewegung der Schulter um 2 Achsen. Dieser Ablauf ist bekannt unter dem *Paradoxon* von Codman. Dieser komplizierte Verlauf lässt sich erst dann realisieren, wenn das Schultergelenk unter Berücksichtigung aller beteiligten Muskeln und

Sehnen simuliert wird, denn nur so lässt sich das Codman Paradoxon reproduzieren. Im Rahmen dieser Arbeit wurde das Schultergelenk jedoch als Kugelgelenk realisiert. Die folgende Grafik verdeutlicht das (scheinbare) Paradoxon.

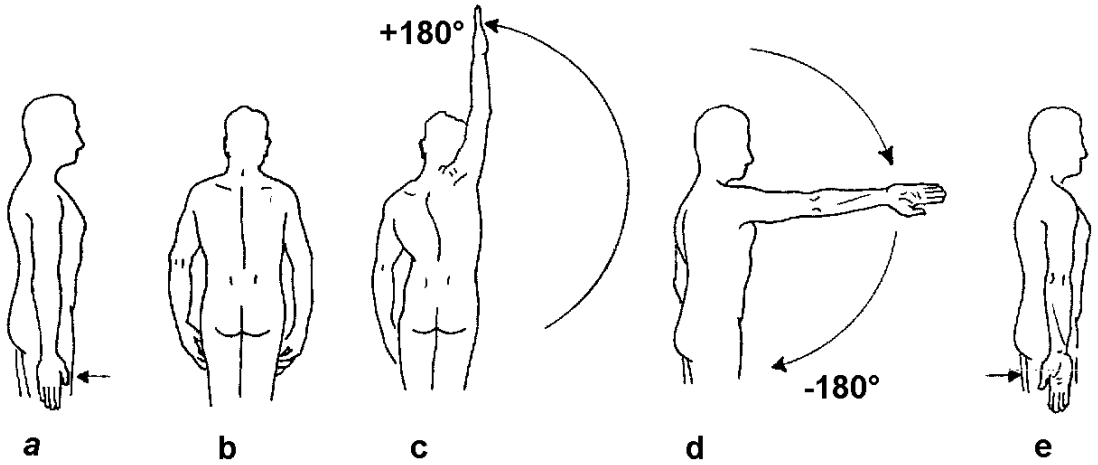


Abbildung 6.7: Das *Paradoxon* nach Codman. Aus Kapandji [39].

Schulter	aktiv	passiv	Bemerkung
Extension	45-50		
Flexion	180		
Abduktion	180		
Adduktion	30		
Außenrotation	80		
Innenrotation	110		

Da eine Drehung des Unterarms nicht berücksichtigt wird reicht es aus, das Ellenbogengelenk als Scharniergelenk zu implementieren. Der Vollständigkeit halber sind aber auch hier die maximalen Rotationswinkel mit aufgelistet.

Ellenbogen	aktiv	passiv	Bemerkung
Extension	145		
Flexion	0		
Supination	90		
Pronation	85		

Das letzte Gelenk des Schulter/Arm/Hand Komplexes ist schließlich das Handgelenk. Da die Hand nur als starrer Körper simuliert wird, macht die Realisierung des Rotationsfreiheitsgrads

keinen Sinn. Das Handgelenk wird deshalb als Kugelgelenk ohne Rotationsfreiheit implementiert. Die Sensitivitätsanalysen haben jedoch gezeigt, dass die Hand auch starr an den Unterarm angekoppelt werden kann und hierdurch das Simulationsergebnis nur sehr schwach beeinflußt wird.

Handgelenk	aktiv	passiv	Bemerkung
Dorsal Flexion	85		
Vola Flexion	85		
Radiale Abduktion	15		
Ulnare Abduktion	15		

Die Wirbelsäule in ihre *Bestandteile* zu zerlegen und jeden Wirbel einzeln zu simulieren wäre erheblich zu aufwändig. Stattdessen wird die Wirbelsäule in 3 Hauptgelenke untergliedert wie dies auch in der Funktionellen Anatomie praktiziert wird. Die maximalen Ausschläge dieser 3 Gelenke resultieren dann in einer Bewegung, die die fein unterteilte Wirbelsäule mit ihren 32 Einzelwirbeln approximiert.

Lendenwirbelsäule	aktiv	passiv	Bemerkung
Extension	35		
Flexion	60		
Laterail Flexion	$\pm$20		

Brustwirbelsäule	aktiv	passiv	Bemerkung
Extension	20		
Flexion	10		
Laterail Flexion	$\pm$15		

Halswirbelsäule	aktiv	passiv	Bemerkung
Extension	45-50		
Flexion	90		
Laterail Flexion	$\pm$45		
Rotation	$\pm$80-90		Die Rotation bewirkt die Drehung des Halses und Kopfs.

6.2 Gelenkwiderstände und Muskelkräfte

Wie bereits erwähnt werden in dieser Arbeit die Gelenkwiderstände und Muskelkräfte derart realisiert, dass in den Gelenken Momente angreifen, die der kummulierten Kraft aller an diesem Gelenk beteiligten Muskeln und Sehnen entsprechen, einzelne Muskeln werden hierbei nicht simuliert. Dabei sollen aber die schon besprochenen Muskel-Länge und Muskel-Geschwindigkeits Zusammenhänge nicht unberücksichtigt bleiben. Ferner wird auch zwischen beugenden und streckenden Muskeln unterschieden, das heißt jede Bewegungsrichtung des Gelenks kann individuell eingestellt werden. Gerade für das Kniegelenk ist dies von entscheidender Bedeutung, denn die Beinbeugemuskulatur ist nur halb so kräftig wie die Beinstreckermuskulatur. Es soll hier jedoch betont werden, dass dieses Modell jederzeit verfeinert und leicht in das bestehende Programm integriert werden kann.

In diesem Modell gilt es also die folgenden Punkte geeignet zu parametrisieren:

- Elastischer Gelenkwiderstand, hervorgerufen durch (Über-) Streckung von Muskeln und Sehnen aus der Nulllage heraus.
- Dissipativer Gelenkwiderstand, resultierend aus der Reibung von Muskeln und Sehnen untereinander und am Gewebe.
- Aktive, also bewußt gesteuerte Muskelkraft

Das gesamte auf ein Gelenk wirkende Moment, ist somit die Summe dieser 3 Beiträge

$$T_{Gelenk} = T_{elastisch} + T_{dissipativ} + T_{aktiv}$$

Das gesamte elastische Moment $T_{elastisch}$ eines Gelenks ist die Summe aller Einzelmomente der am Gelenk beteiligten Muskeln und Sehnen. Es wird hervorgerufen durch überstreckte (verlängerte) Muskeln und Sehnen, wobei ein überstreckter Muskel eine nahezu exponentielle Abhängigkeit (siehe Abbildung 6.2) und eine gestreckte Sehne einen linearen Kraft-Längenverlauf besitzen. Einen weiteren Einfluß auf die absolute Größe des Moments hat natürlich auch noch der jeweilige Hebelarm der Muskeln, der je nach Gelenkstellung stark variieren kann. Bei welchen Gelenkstellungen elastische Momente auftreten können hängt individuell vom Gelenk ab. Zum Beispiel besitzt das Ellenbogengelenk innerhalb der Gelenkanschläge nahezu kein elastisches Moment, das Sprunggelenk hingegen ist extrem elastisch; schon eine kleine Auslenkung aus der Nulllage erzeugt eine rücktreibende Kraft. Individuell für jede Gelenkachse und auch jede Drehrichtung wird deshalb der folgende Ansatz gemacht.

$$T_{elastisch} = A \cdot \alpha + B \cdot \alpha^2$$

Je nach Wahl der Konstanten A und B kann hierbei mehr der Einfluß der überstreckten Muskeln oder der Einfluß der überstreckten Sehnen eingestellt werden.

Der dissipative Gelenkwiderstand $T_{dissipativ}$ ist sowohl unabhängig von der Aktivierung der Muskulatur als auch unabhängig von der momentanen Länge der Muskeln und Sehnen und hängt ausschließlich von der Größe des Gelenks (also der Größe der reibenden Flächen) und der momentanen Gelenkgeschwindigkeit $\dot{\alpha}$ ab. Es wurde der folgende Ansatz gewählt:

$$T_{dissipativ} = D \cdot \dot{\alpha}$$

Je nach Gelenk liegt die Dissipationskonstante bei $D = 0.1 - 0.3N/ms$. Der Einfluß der Gelenkreibung ist somit nicht sehr stark; die menschlichen Gelenke haben zwar nicht die Reibeigenschaften eines Kugellagers, jedoch wird hierdurch kaum Energie verloren.

Das aktive Moment T_{aktiv} eines Gelenks hängt sowohl von der aktuellen Gelenkstellung, der Gelenkwinkelgeschwindigkeit als auch von der Aktivierung der Muskulatur ab. Da an der Gelenkbewegung immer mehr als nur ein Muskel beteiligt ist, ist je nach Stellung ein Muskel mehr oder weniger aktiv und stellt je nach Hebelarm einen unterschiedlichen Beitrag zum Gesamtmoment. Aufgrund des unterschiedlichen und komplizierten Zusammenspiels dieser Muskeln muß die Moment-Winkel Kurve für jedes Gelenk individuell angepasst werden. Realisiert wird dies, indem durch experimentell bestimmt Momente bei bestimmten Winkeln ein Polynom gefittet wird [1]. Es wurde die folgende Parametrisierung für alle aktiv simulierten Gelenke gewählt, die die Winkel-Moment Abhängigkeit sehr gut approximiert

$$T_{aktiv} = A + B \cdot x + C \cdot x^2 + D \cdot x^3$$

Hierbei müssen die Koeffizienten A, B, C und D durch einen Kurvenfit bestimmt werden, bei dem 3 bis 4 Punkte vorgegeben werden. Üblicherweise werden dabei die Momente an den Gelenkanschlägen sowie das Maximum vorgegeben und durch diese Punkte die Kurve gefittet.

Das zur Verfügung gestellte Drehmoment liegt natürlich nicht instantan an, denn vom bewußt gesteuerten Gedanken (z.B. Knie strecken, Hüfte beugen, ...) bis hin zur aktivierten Muskulatur verstreicht eine gewisse Zeit. Diese Verzögerung wird durch eine Differentialgleichung 1. Ordnung beschrieben.

$$\dot{a}(t) = \frac{1}{\tau} \cdot [u(t) - (\beta + (1 - \beta) \cdot u(t)) \cdot a(t)] \tag{6.1}$$

Hierbei ist $u(t)$ das neuronale Eingangssignal und $a(t)$ die momentane Aktivität der Muskulatur zum Zeitpunkt t. τ ist die Zeitkonstante, die die Geschwindigkeit des Aktivierungsprozesses festlegt. Bei maximalem Eingangssignal $u(t) = 1$ baut sich die Aktivität mit der Zeitkonstante τ auf. Die Relaxierung der Muskulatur geschieht in der Regel etwas langsamer, berücksichtigt wird dies durch die Konstante β. Bei minimalem Eingangssignal $u(t) = 0$ baut sich somit die Aktivität der Muskulatur mit der Zeitkonstante $\frac{\tau}{\beta}$ wieder ab. Hierbei ist die Zeitkonstante $\tau = 0.05s$ und β mit 0.2 angesetzt. Am Winkel-Moment Verlauf der Hüftmuskulatur, wird dies exemplarisch dargestellt. Folgend ist der Verlauf der Beugemuskulatur

[1] Es können natürlich auch andere Funktionen, z.B. Splines verwendet werden. Bei sehr komplexen Moment-Winkel Verläufen kann auch ein Datensatz mit beliebig vielen Stützstellen verwendet werden, der linear zwischen diesen Punkten interpoliert.

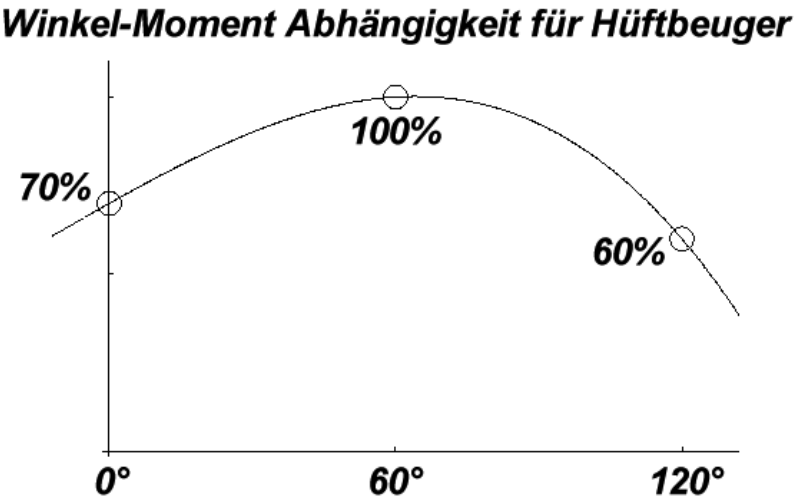


Abbildung 6.8: Abhängigkeit des bereitgestellten Drehmoments vom Gelenkwinkel für die Hüftbeugemuskulatur. Die Beugemuskulatur entwickelt bei 60 Grad ihr maximales Drehmoment, nimmt dann aber an den Gelenkanschlägen bis auf 70% bei gestreckter Hüfte und 60% bei maximal gebeugter Hüfte ab. Das maximale Drehmoment liegt zwischen 50 bis 200Nm.

Und in der nächsten Grafik der Verlauf der Streckermuskulatur

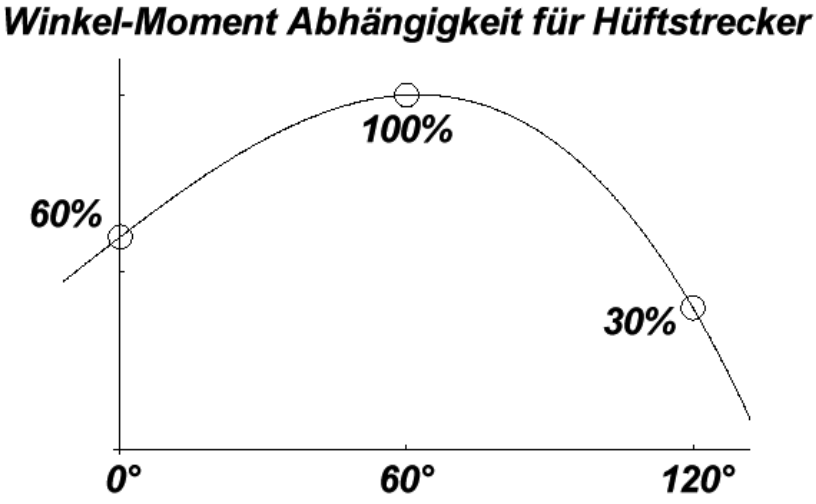


Abbildung 6.9: Abhängigkeit des bereitgestellten Drehmoments vom Gelenkwinkel für die Hüftstreckermuskulatur. Die Streckermuskulatur entwickelt im Gegensatz zur Beugemuskulatur nahezu ein doppelt so starkes Drehmoment, typischerweise 100 bis maximal 400Nm. Allerdings nimmt das Drehmoment an den Gelenkanschlägen schneller ab, 60% vom Maximum bei gestreckter Hüfte und lediglich 30% bei maximal gebeugter Hüfte.

Kapitel 7

Vorwärtssimulation

Ein weites Forschungsgebiet der Biomechanik und Computeranimation ist die Synthetisierung und Rekonstruktion von menschlichen Bewegungsabläufen, wie dem Laufen oder Reckturnen. Den Biomechaniker interessiert hierbei besonders der zeitliche Verlauf der inneren Kräfte und Momente die benötigt werden um den gewünschten Bewegungsablauf zu erreichen. Der Computeranimateur dagegen möchte eine möglichst realistisch aussehende Bewegung und ist weniger an den physikalischen Größen interessiert. Dabei sollte die Animation natürlich eine physikalische Validierung besitzen um eine Aussage darüber machen zu können, ob die Bewegung überhaupt möglich ist. Insofern müssen für eine vernünftige Rekonstruktion von Bewegungsabläufen diese sowohl *real aussehen* als auch im Rahmen des physikalischen Modells möglich, also validierbar sein.

Hierzu gibt es die verschiedensten Methoden. Ein sehr vielversprechender Ansatz, der sich auch besonders im Rahmen dieser Arbeit einsetzen ließe ist die Verwendung von Kontrollalgorithmen zur Bewegungsrekonstruktion. Die Kontrollalgorithmen werden hierbei durch physikalisches und biomechanisches Wissen sowie durch Intuition entwickelt. Beeindruckende Animationen finden sich in Animating Human Athletics [35] von Hodgins, Wooten, Brogan und O'Brien. Hier werden am Beispiel des Laufens und Fahrradfahrens sehr real aussehende Animationen erreicht und dies mit relativ wenigen Kontrollalgorithmen. Interaktive Geschwindigkeiten im Zusammenspiel mit dem hier vorliegenden Verfahren sind hierbei denkbar. Das Hauptproblem dieses Ansatzes ist, dass die Kontrollalgorithmen aufwändig per Hand implementiert und die benötigten Parameter durch viel Feinarbeit justiert werden müssen. Insofern sind diese Ansätze für Bewegungen die im Rahmen des Kontrollalgorithmus liegen sehr gut geeignet, versagen jedoch wenn die Bewegung auch nur leicht abweicht. So wird der Kontrollalgorithmus zur Simulation des menschlichen Laufens nicht aus einer anfangs stehenden Position gestartet werden können, er ist somit nicht selbst startend, sondern die initiale Position und Geschwindigkeit muß schon dem Laufen entsprechen.

Ein anderer, wesentlich zeit- und rechenintensiverer Ansatz, verwendet Optimierungsalgorithmen um einen vorgegebenen Weg zu rekonstruieren. Der gewünschte Bewegungsablauf kann hierbei per Hand oder bei komplexeren Bewegungen durch Videoaufzeichnungen vorgegeben werden. Mittels inverser Dynamik und/oder Optimierungsalgorithmen werden dann die Momente rekonstruiert. Bis jetzt ist dieser Ansatz aber nur für verhältnismässig einfache Be-

wegungen verwendet worden und interaktive Geschwindigkeiten sind hiermit bei der Leistung heutiger Rechner nicht zu erwarten. In der Arbeit von Andrew Witkin und Michael Kass [36] werden einfache Beispiele beschrieben. Die Erweiterung auf komplexere Systeme ist hierbei allerdings alles andere als trivial. In der Arbeit von Anthony C.Fang [37] wird eine Methode vorgestellt wie sich dieser Ansatz beschleunigen lässt. Inwiefern diese Methode jedoch für allgemeine Systeme einsetzbar ist, ist unklar.

Am Beispiel der Vorwärtssimulation eines Reckturners soll in diesem Kapitel die Universalität und Leistungsfähigkeit des in dieser Arbeit entwickelten Verfahrens demonstriert werden. Dabei wird nicht eine Vorwärtssimulation mittels Kontrollalgorithmen realisiert sondern die Bewegung interaktiv per Hand gesteuert. Zuvor soll aber kurz auf die Probleme der inversen Dynamik zur Bewegungsrekonstruktion eingegangen werden.

7.1 Probleme bei inverser Dynamik

Bei der Bestimmung der Momente mittels inverser Dynamik wird der Bewegungsablauf einer realen Person zuvor mittels mehrerer Kameras aufgenommen. Hierzu werden Marker auf die Haut über den Gelenken angebracht. Dies wird nachfolgend anhand eines Reckturners aufgezeigt. Die folgenden Bilder entstammen den Examensarbeiten von Dirk Haberkamp [42] und Thomas Göth [43], die am Institut für Sportwissenschaft der Universität Koblenz-Landau verfasst wurden. In Dirk Haberkamps Arbeit wird die biomechanische Bewegungsanalyse einer kompletten Riesenfelge besprochen. Zur Aufnahme der Bewegung wurden insgesamt 4 Kameras mit einer Verschlußzeit von $1ms$ verwendet. Thomas Göth verglich die Bewegung zweier Turner bei einem Kippaufschwung.

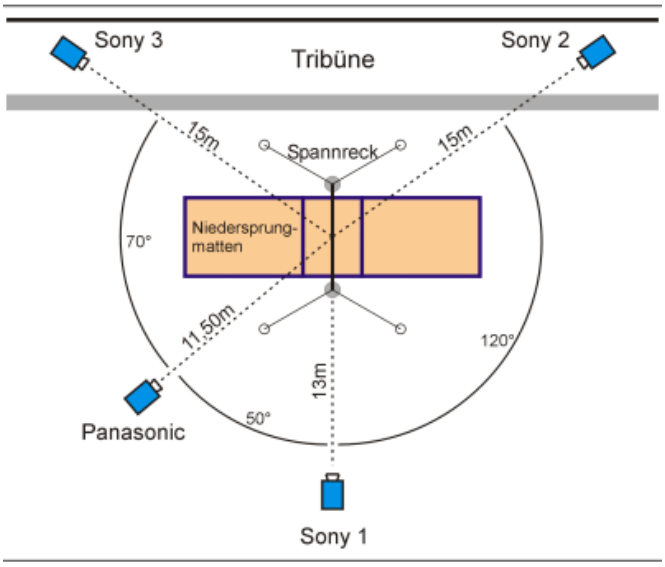


Abbildung 7.1: Kamerakonfiguration zur 3D-Aufnahme eines Reckturners. Aus der Arbeit von Dirk Haberkamp [42].

Die Befestigung der Marker sollte so nah wie möglich an dem jeweiligen Gelenkdrehpunkt erfolgen, da es sonst zu Abweichungen von der tatsächlichen Drehachse kommt. In dem nachfolgenden Bild ist dies zu sehen.

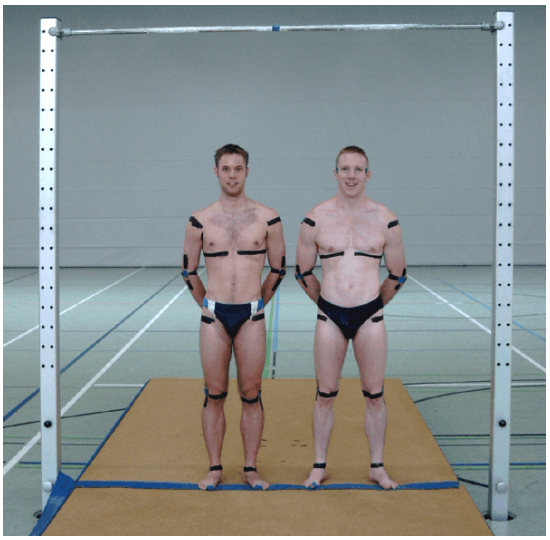


Abbildung 7.2: Markerpositionen zur 3-dimensionalen Aufnahme des Bewegungsablaufs beim Reckturnen. Links: Dirk Haberkamp, rechts: Sebastian Quirbach. Aus der Arbeit von Thomas Göth [43].

Die Marker wurden an den folgenden Punkten angebracht:

- Ellenbogengelenk links/rechts
- Schultergelenk links/rechts
- Brustkorb links/rechts
- Hüftgelenk links/rechts
- Kniegelenk links/rechts
- Knöchel links/rechts
- Ferse links/rechts
- Fußspitze links/rechts
- Schläfe links/rechts

Die Bilder der einzelnen Kameras wurden zur weiteren 3D-Positionsanalyse dem Programm SIMI Motion [44] der Firma SIMI Reality Motion Systems GmbH zugeführt. Aus der Kenntnis der 3 Kamerapositionen, ihrer Blickrichtungen und der Öffnungswinkel kann dann die Position jedes einzelnen Markers berechnet werden. Man erhält somit über den gesamten Zeitraum des

Bewegungsablaufs die 3D Position aller Marker. Die folgende Grafik zeigt die Bilder von 3 der insgesamt 4 Kameras zusammen mit der rekonstruierten Position der Marker im Programm SIMI.

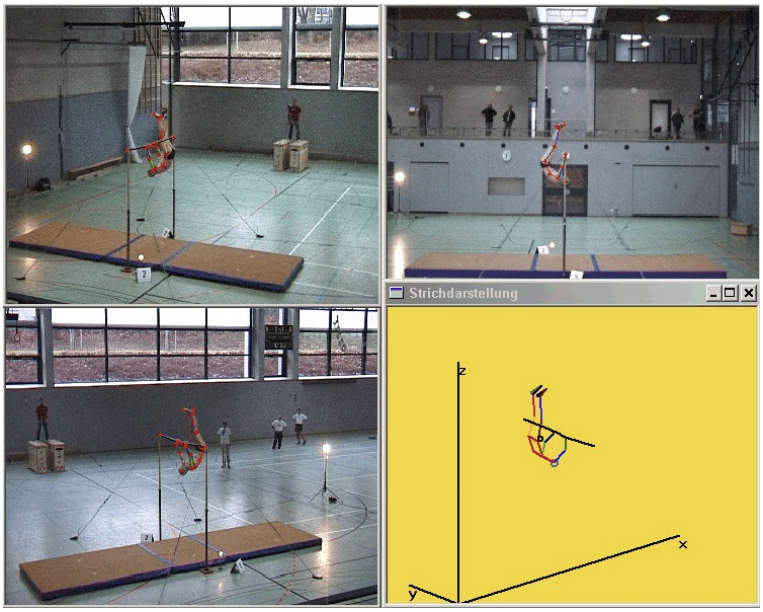


Abbildung 7.3: Einzelbilder von 3 der insgesamt 4 Kameras sowie die analysierte Bewegung im Programm SIMI. Aus der Arbeit von Dirk Haberkamp [42].

Aus den zeitlichen Verläufen der Positionen der einzelnen Körperelemente lassen sich dann direkt die Geschwindigkeiten und Beschleunigungen durch Differenzierung mittels finiter Differenzen berechnen. Mit einem biomechanischen Modell des Reckturners werden aus den so erhaltenen Beschleunigungen direkt die Kräfte und Drehmomente berechnet. Dieser Ansatz birgt jedoch eine ganze Reihe von Problemen: So sind die Eingabedaten (also die Positionen der Marker) nur annähernd mit den echten Gelenkpunkten gleichzusetzen, denn diese Marker werden ja auf die Haut aufgeklebt, die während der Bewegung mehr oder weniger starke Relativbewegungen zum eigentlichen Gelenkdrehpunkt durchführen. Auch die nicht konstanten Gelenkdrehpunkte, wie sie zum Beispiel im Knie und der Schulter existieren, können mit einem einfachen Marker nicht aufgezeichnet werden. Desweiteren erhält man unweigerlich Messfehler bei der Aufnahme der Markerpositionen, die in einem Rauschen resultieren. Bei der Berechnung der Beschleunigungen wird durch die zweimalige Differenzierung dieses Rauschen stark verstärkt weshalb es unumgänglich ist die Beschleunigungen mittels geeigneter Filter zu glätten. Ein weiteres Problem der inversen Dynamik ist die extrem hohe Sensitivität der ermittelten Momente bezüglich des verwendeten biomechanischen Modells. An einem sehr einfachen Fall wurde dies bereits eindrucksvoll von Karin Gruber und Hanns Ruder in [41] beschrieben. Selbst minimale Modifikationen in der Realisierung der Schwabbelmassen resultieren in völlig verschiedenen zeitlichen Momentverläufen. Die so erhaltenen Kräfte und Momente sind deshalb wenig aussagekräftig und haben mit den tatsächlich auftretenden Kräften wenig gemeinsam.

7.2 Simulation eines Reckturners

Ein anderer Ansatz zur Bewegungsanalyse und Bewegungsrekonstruktion ist die Vorwärtssimulation. Hierbei werden die aktiven Momente in den Gelenken vorgegeben aus denen wiederum die Bewegung resultiert. Eine exakte Bewegungsrekonstruktion ist in diesem Ansatz anfangs also nicht gegeben und kann erst durch geeignete Optimierungsverfahren erreicht werden. Wie diese Kräfte und Momente vorgegeben werden, liegt im Ermessen des Anwenders und hängt mit der Zielsetzung zusammen. Eine häufig verwendete Methode ist, durch geeignete Kontrollalgorithmen oder PID Regler für die Momente Bewegungen zu synthetisieren oder rekonstruieren, bei denen die Winkelpositionen der einzelnen Gelenke als Zielwerte vorgegeben werden. Ein weiterer hier vorgestellter Weg ist, diese Momente manuell per Hand vorzugeben. Aufgrund der Interaktivität der Simulation lassen sich somit schnell und effektiv viele verschiedene Simulationsszenarien *ausprobieren*. Diese Bewegungen können nachträglich mit Optimierungsverfahren weiter verbessert werden. Die Sensitivität dieser Momentverläufe bezüglich des zugrundeliegenden Menschmodells ist wesentlich geringer als bei der inversen Dynamik.

7.2.1 Vorwärtssimulation einer Riesenfelge

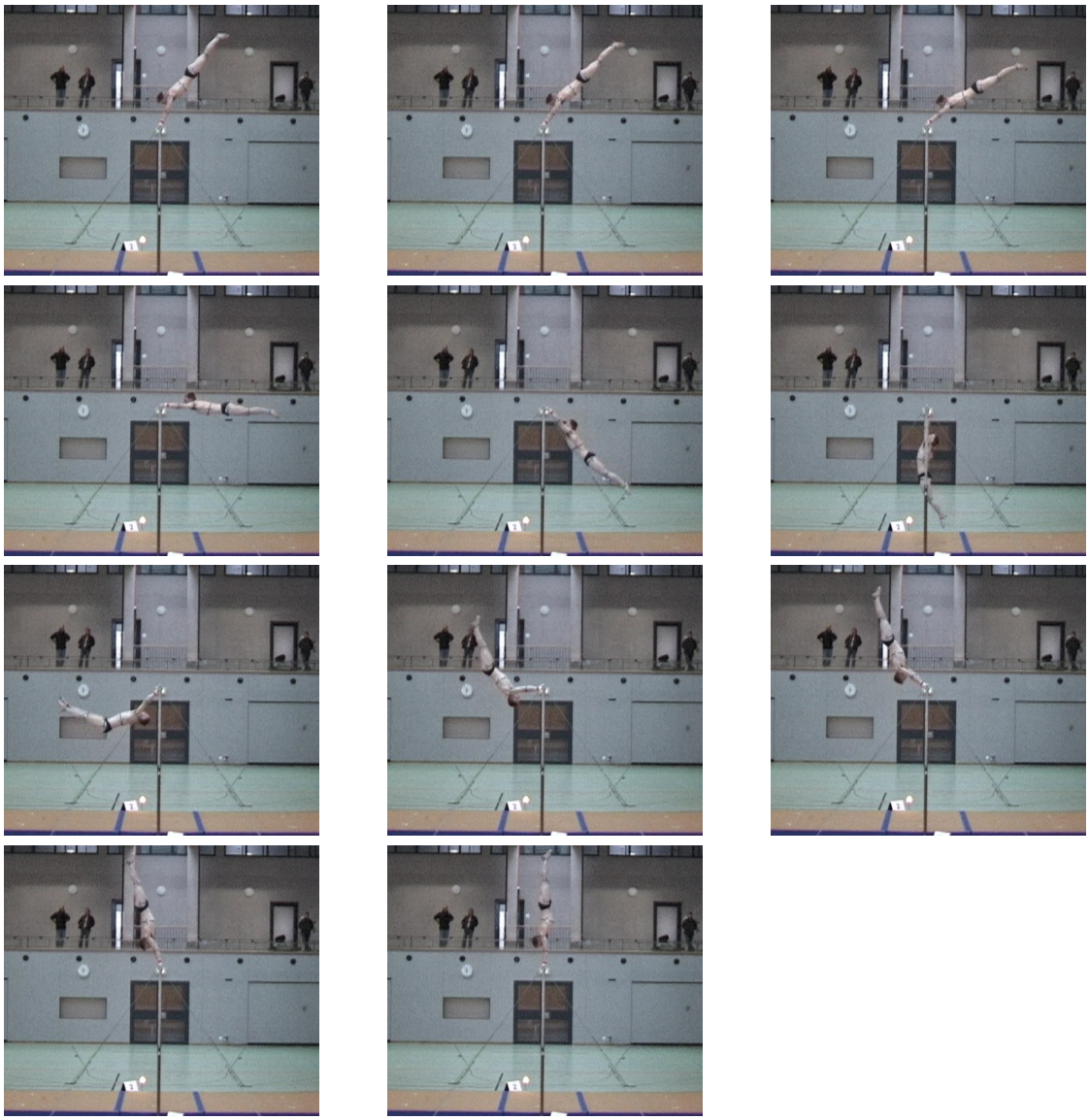
Am Beispiel eines aktiv per Hand simulierten Reckturners soll nun eine Vorwärtssimulation demonstriert werden. Der Reckturner bietet sich hier sehr gut an, da die Implementierung nicht übermässig aufwändig ist (es reichen hier insgesamt 2, maximal aber 4 Momente zu einer flüssig aussehenden Simulation) und die Bewegungsanalyse von Reckturnern bereits intensiv studiert wurde, man hier also auf einen großen Datenbestand zurückgreifen kann.

Die simulierten Momente wirken um das Knie-, Hüft-, Schulter- und Ellenbogengelenk, wobei dem Hüft- und Schultergelenk die zentrale Bedeutung zukommt [1]. Alle anderen Gelenke können während der Bewegung als starr angenommen werden. Dass dies gerechtfertigt ist, ist durch Videoaufzeichnungen belegt. Die Vorgabe dieser 4 Momente erfolgt mittels eines analogen 4-Achsen Eingabegeräts, so dass man jedes Moment individuell regeln kann. Die Geschwindigkeit der Simulation erlaubt eine interaktive Steuerung, da eine Simulationsgeschwindigkeit von ca. $10 - 30\%$ der Echtzeit erreicht wird. Die Implementierung der Momente geschah wie in Kapitel 6 beschrieben. Simuliert werden soll ein anfangs ruhender, am Reck hängender Turner, der einen Aufschwung mit anschließender Riesenfelge durchführt. Die nachfolgende Bildersequenz mit 26 Einzelbildern zeigt in Abständen von $0.2s$ den Ablauf einer Riesenfelge (insgesamt also 5 Sekunden). Diese Riesenfelge wurde von Sebastian Quirbach durchgeführt, der mit der KTV-Koblenz in der 2. Bundesliga turnt und mir auch bei der simulierten Riesenfelge wichtige Informationen gab auf was man beim Aufschwung besonders achten muß. Es ist hierbei anzumerken, dass der Turner im ersten Bild bereits eine leichte kinetische Energie besitzt, die er durch den Sprung vom Boden zur Reckstange erhielt. Leider hatte ich kein Video und Bildmaterial für einen anfangs ruhenden Turner. Es soll hier aber vor allem verdeutlicht werden wie der generelle Bewegungsablauf bei einer Riesenfelge,

[1] Die meisten Figuren am Reck können auch nur mit dem Hüft- und Schultergelenk geturnt werden, wobei die Arme und die Beine gerade gestreckt bleiben.

durch geeignete Veränderung des Schwerpunktabstands zur Reckstange (Pendelverkürzung!) erfolgt, um dem Turner mehr Energie zuzuführen.

Abbildung 7.4: Bildsequenz zum Turnen der Riesenfelge. Jedes Einzelbild entspricht 0.2s. Geturnt wurde die Riesenfelge von Sebastian Quirbach.



Fortsetzung der Bildsequenz zum Turnen der Riesenfelge. Jedes Einzelbild entspricht 0.2s. Geturnt wurde die Riesenfelge von Sebastian Quirbach.

Wohl gemerkt kann die Riesenfelge auch mit modifizierter Technik geturnt werden, so dass sich der Bewegungsablauf von der obigen Bildsequenz deutlich unterscheiden würde. Auch die erneute Aufnahme der gleichen Turnübung mit dem gleichen Turner kann eine deutliche Abweichung ergeben. Am Kippaufschwung hat Thomas Göth [43] in seiner Arbeit den Bewegungsablauf zweier unterschiedlich trainierter Turner studiert. Bildsequenzen sollen den Unterschied am Aufschwung demonstrieren.

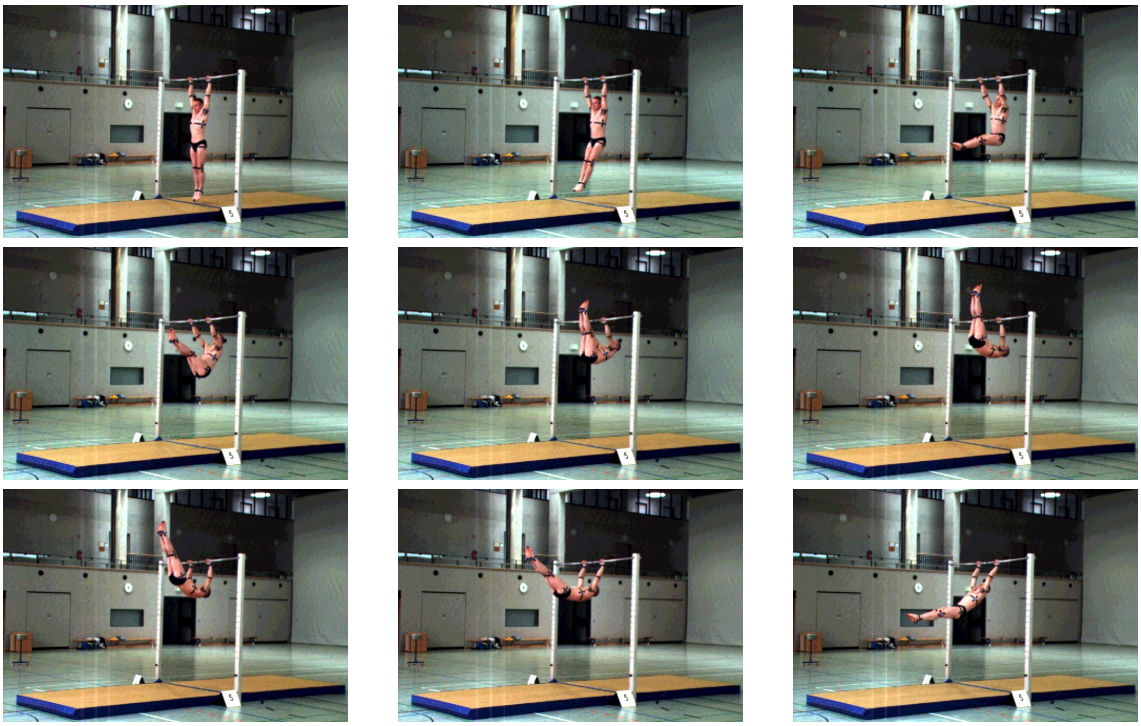


Abbildung 7.5: Aufschwung des besser trainierten Turners Sebastian Quirbach.

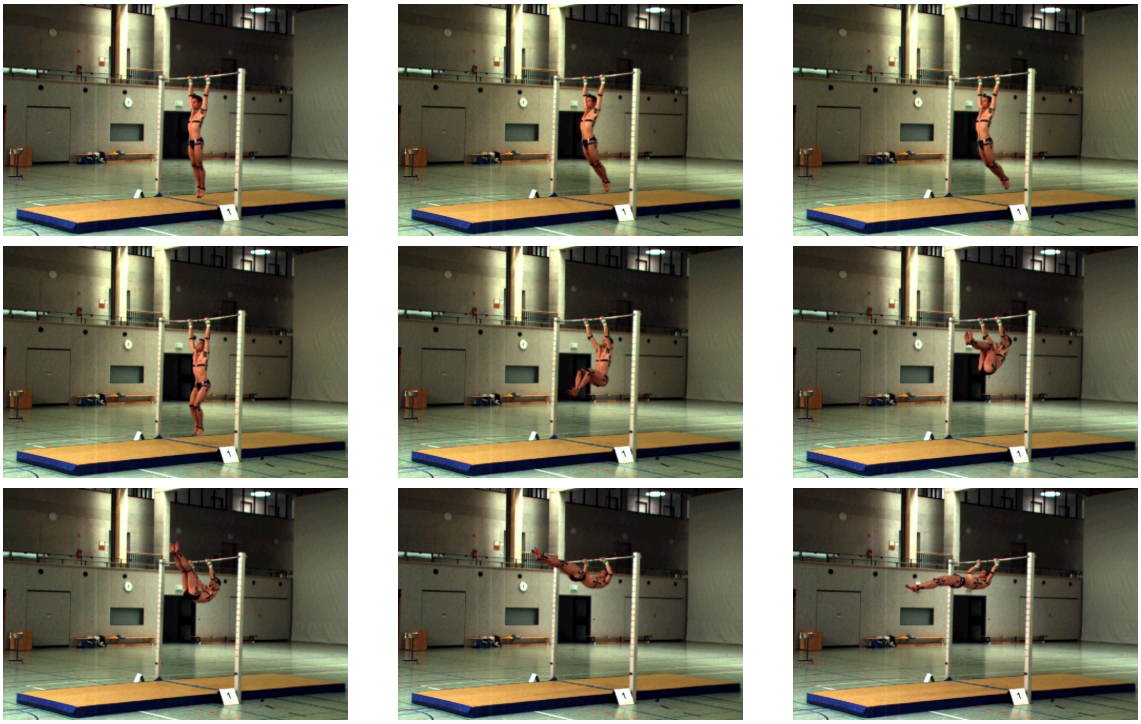


Abbildung 7.6: Aufschwung des zweiten Turners Dirk Haberkamp.

Interessant ist anzumerken, dass Sebastian Quirbach nahezu ohne Knieeinsatz den Aufschwung realisiert wohingegen Dirk Haberkamp ein großes Moment im Knie verwendet.

Ziel soll es nun sein, mit dem in dieser Arbeit entwickelten Verfahren den in Abbildung 7.4 dargestellten Bewegungsablauf zum Turnen einer Riesenfelge zu simulieren bzw. zu reproduzieren um hieraus Kenntnis über die benötigten Momente zu erlangen. Hierbei wurde das biomechanische Menschmodell so implementiert wie es in den vorherigen Kapiteln beschrieben wurde (Aufgliederung in 16 Einzelkörpern und Berücksichtigung von Gelenkanschlägen). Auf eine 100-prozentige Reproduzierung der Bewegung wird hierbei nicht Wert gelegt, vielmehr soll eine Kenntnis der für diese Bewegung benötigten Momente und Kräfte gewonnen werden, die bei solch einer Turnübung zu erwarten sind. Die mittels des inversen Dynamik-Moduls der SIMI Software gewonnenen Momente reproduzieren zwar die gewünschte Bewegung, jedoch haben Sie mit den tatsächlich auftretenden Momenten in den Gelenken wenig gemeinsam.

Für alle Gelenkverbindungen wurden extrem harte und schwach gedämpfte Feder/Dämpfer Elemente verwendet, um bessere Vergleichsmöglichkeiten mit den Standard Mehrkörpersystemen zu ermöglichen und um den Energieverlust in Grenzen zu halten. Dabei wurde für alle Gelenke ein linearer Ansatz verwendet

$$F_{Gelenk} = K \cdot d + D \cdot \dot{d}$$

Hierbei ist d der Abstand zweier Gelenkpunkte eines Gelenks und $\dot{d}$ die relative Geschwindigkeit zwischen beiden Punkten. Mit der Wahl von $K = 1e^8 N/m$ und $D = 2000 Ns/m$ bewegen sich die Gelenkpunkte von zwei Körpern bei den in dieser Simulation auftretenden Kräften nie mehr als $1mm$ auseinander [2]. Der Energieverlust betrug weniger als 0.5% der Gesamtenergie pro Sekunde.

Bis auf das Knie-, Hüft-, Schulter- und Armgelenk wurden alle anderen Gelenke als starre Gelenke implementiert. Realisiert wird dies, indem ein Gelenk mit insgesamt 3 Feder/Dämpfer Elementen aufgebaut wird, so dass keine Freiheitsgrade mehr existieren.

Entscheidender Bedeutung kommt auch der Realisierung der Gelenkanschläge und der Stangenkraft zu. Für die Gelenkanschläge wurde ein nichtlinearer Ansatz zur Rückstellkraft verwendet:

$$F_{Gelenkanschlag} = K \cdot d^3 + D \cdot d \cdot \dot{d}$$

Hierbei ist d ein Mass dafür wie weit ein Gelenk sich ausserhalb seines Bewegungsfreiraums befindet und $\dot{d}$ ist die relative Geschwindigkeit. Wie d und $\dot{d}$ gemessen werden ist völlig beliebig. In dieser Simulation ist d der kürzeste Abstand des normierten Vektors von der Kegelfläche (also einfach die Projektion) wie es in Abbildung 6.5 dargestellt ist. Es wurde für alle Gelenke eine Federkonstante von $K = 2000 N/m^3$ und eine Dämpfungskonstante von $D = 200 Ns/m^2$ verwendet. Nur beim gestreckten Knieanschlag wurde $K = 10000 N/m^3$ und $D = 400 Ns/m^2$ gewählt.

[2] Bei diesem schwach gedämpften System kamen die BDF-basierten Verfahren wie VODE und CVODE bereits an ihre Stabilitätsgrenzen was in höheren Integrationszeiten resultierte. Eine weitere Reduzierung der Dämpfungskonstante wird von diesen Verfahren nicht mehr effizient integriert, hier kann nur noch RADAU5 verwendet werden.

Für die Stangenkraft wurde ein linearer Ansatz verwendet, wie er nach Messungen von Dirk Haberkamp in seiner Examensarbeit gerechtfertigt ist. Die nachfolgende Grafik zeigt die gemessene Stangenauslenkung und die aus der aufgezeichneten Bewegung berechnete Stangenkraft.

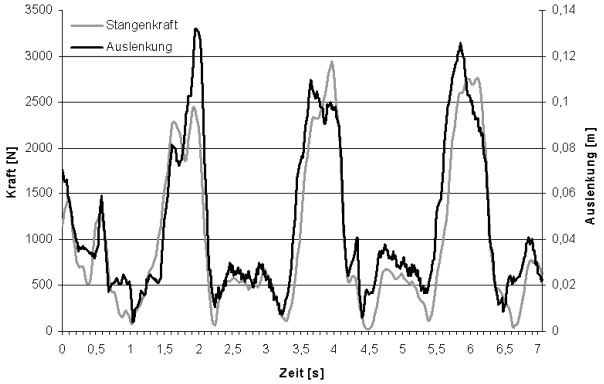


Abbildung 7.7: Gemessene Stangenauslenkung und berechnete Stangenkraft in Abhängigkeit der Zeit. Aus der Arbeit von Dirk Haberkamp [42].

Bei einer radial nach außen gerichteten Kraft von ca. $2000N$ erfährt die Stange eine Auslenkung von ca. $0.1m$. Ein Nachfedern wurde hierbei nicht beobachtet, was auf eine relativ hohe Dämpfung hindeutet.

$$F_{Reckstange} = K \cdot d + D \cdot \dot{d}$$

Für die Simulation wurde eine Federkonstante $K = 20000N/m$ und eine Dämpfungskonstante von $D = 500Ns/m$ gewählt. Das biomechanische Menschmodell wurde eng an den Turner Sebastian Quirbach angepasst, der 1.77 m groß und zum Zeitpunkt der Aufnahme 76.9 kg wog. Die Massen der Körperteile sind in der folgenden Tabelle gelistet.

Körperteil	Masse [kg]
Kopf	5.0
Brustkorb	10.1
Bauch	14.6
Hüfte	11.6
Oberschenkel	8.3
Unterschenkel	4.1
Fuß	1.3
Oberarm	2.1
Unterarm	1.7
Hand	0.3

Der Reckturner wurde mit dem RADAU5 Verfahren integriert. Die absolute Fehlertoleranz wurde auf $AbsTol = 1e^{-6}$ und die relative auf $RelTol = 1e^{-11}$ gesetzt. Um eine vernünftige Reaktionszeit zu ermöglichen ist die Simulation gebremst, so dass die Simulation nie schneller als 10% der Echtzeit abläuft. Die nachfolgenden Bilder zeigen den Bewegungsablauf in Abständen von $0.2s$.

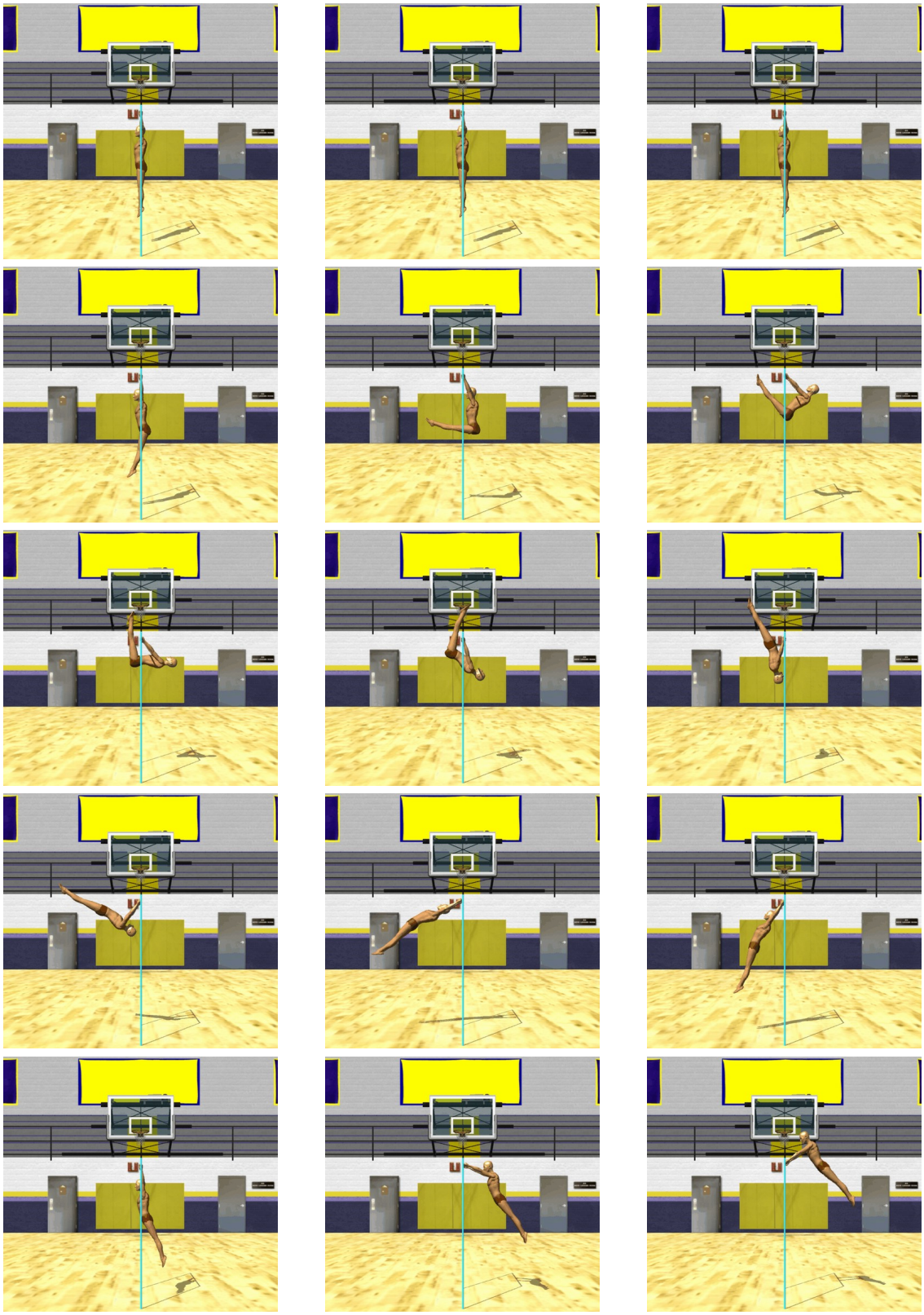
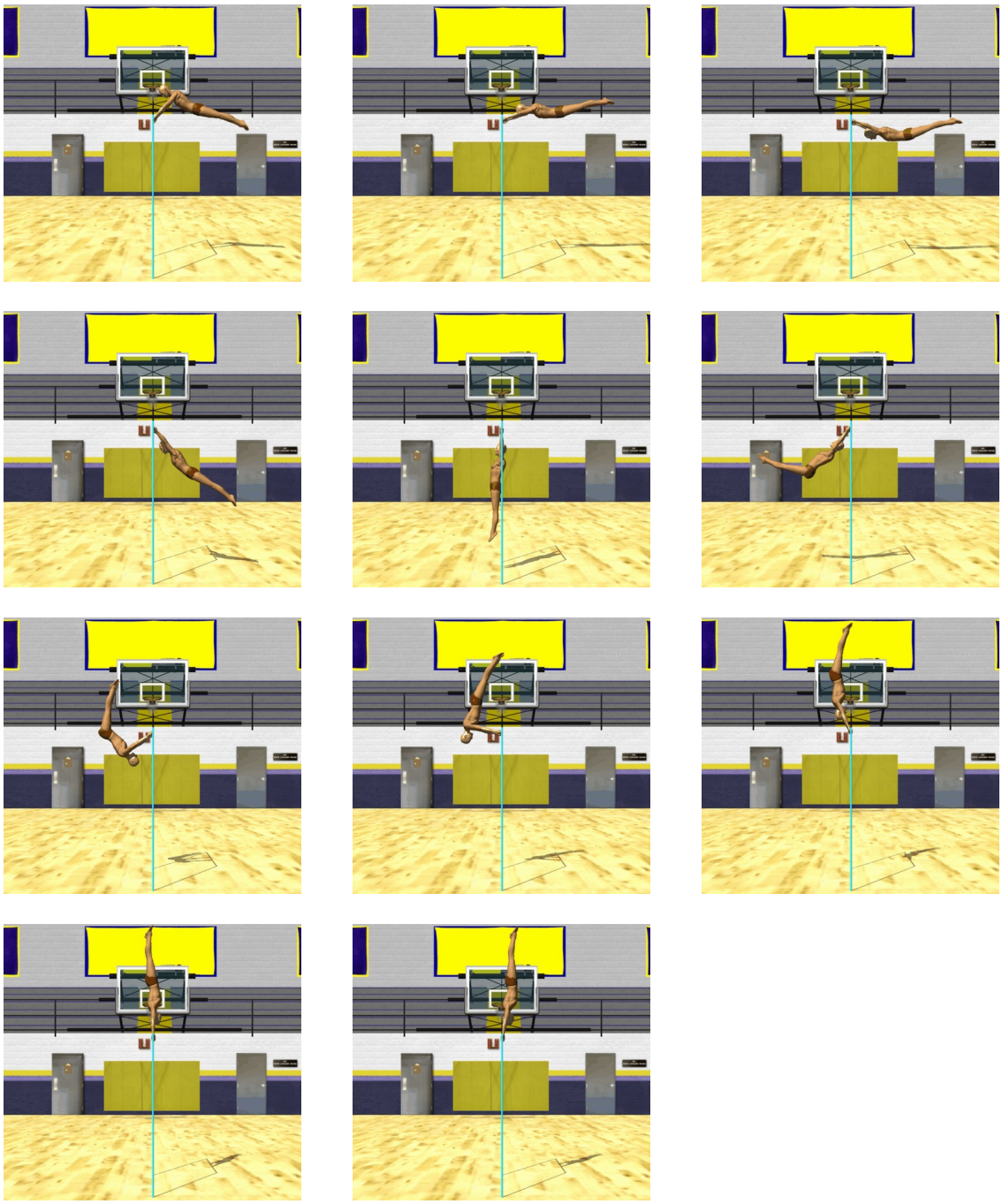


Abbildung 7.8: Simulierte Riesenfelge. Jedes Einzelbild entspricht 0.2s.



Fortsetzung der simulierten Riesenfelge.

7.2.2 Bewegungsanalyse Riesenfelge

Anhand von diversen Grafiken soll in diesem Kapitel nun eine kurze Bewegungsanalyse der Riesenfelge erfolgen. Leider hatte ich keinen Zugriff auf die reinen Rohdaten der Messung von Dirk Haberkamp. Hierdurch ist der direkte Vergleich zwischen der simulierten (7.4) und der aufgezeichneten (7.8) Riesenfelge etwas problematisch, da ich nur bestehende Grafiken verwenden konnte. Neben dem rein optischen Vergleich des Bewegungsablaufes können desweiteren auch der zeitliche Verlauf des Körperschwerpunktabstands zur Reckstange sowie die Gesamtenergie herangezogen werden. Wie man sieht ist die simulierte Riesenfelge etwas anders geturnt worden als die aufgezeichnete, insofern sind die beiden Bewegungsabläufe aus Abbildung 7.4 und 7.8 nicht deckungsgleich und auch nicht ganz synchron. Dennoch beinhalten beide Sequenzen die drei grundlegenden Elemente zum Turnen einer Riesenfelge: Der schnelle Aufschwung unter großem Einsatz von Hüft- und Schultergelenk gefolgt von einer weiteren Energiegewinnung beim Rückschwung mit starkem Einsatz des Schultergelenks [3] und letztlich der erneute Aufschwung (wieder mit Hüft- und Schultergelenk Einsatz) mit anschließendem Handstand. Nachfolgend ist der zeitliche Verlauf des Körperschwerpunktabstands (KSP) zur Reckstange dargestellt. Leider konnten beide Kurven nicht in einer Grafik dargestellt werden, da wie erwähnt kein Zugriff auf die Rohdaten im aufgezeichneten Fall möglich war. Der dargestellte Zeitbereich umfasst den Aufschwung bis hin zum Ende des Rückschwungs.

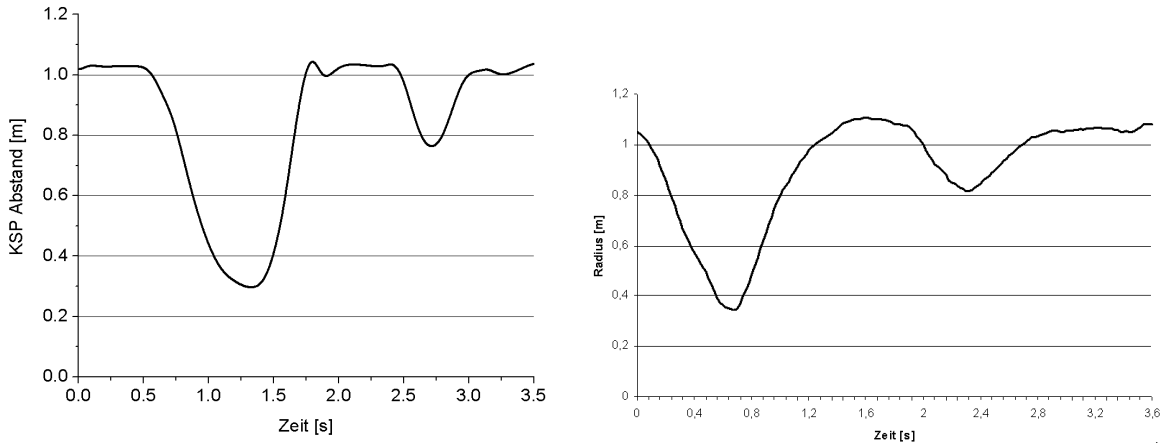


Abbildung 7.9: Vergleich des KSP-Abstands zur Reckstange. Links die simulierte Riesenfelge; rechts die aufgezeichnete Riesenfelge von Sebastian Quirbach.

Trotz der unterschiedlichen Achslängen zueinander erkennt man sehr schön, dass sich beide KSP-Verläufe gut miteinander decken. Vor allem stimmt die zeitliche Dauer der zwei Phasen der KSP-Abstandsänderung zur Reckstange in beiden Fällen gut überein. Beim ersten Aufschwung dauert dies ca. $1.2s$ und beim zweiten Ranholen etwa $0.5s$. Der KSP-Abstand liegt in der Anfangssituation in beiden Fällen bei ca. $1.03m$. Gut zu erkennen ist, dass in

[3] Sebastian Quirbach wies darauf hin, dass der Rückschwung so wie er in 7.4 geturnt wurde unüblich ist. Der Rückschwung der simulierten Riesenfelge ist hier lehrbuchmässiger.

der simulierten Riesenfelge der Schwerpunkt beim Aufschwung etwas näher an die Stange gebracht wird ($0.29m$) als in der realen Riesenfelge ($0.34m$). Dieser kürzere KSP-Abstand sollte bei korrektem Einsatz der Momente in einem höheren Energiegewinn resultieren. Der zeitliche Verlauf der Gesamtenergie ist in der nachfolgenden Grafik verglichen. Der simulierte und der aufgenommene Fall weichen wesentlich stärker voneinander ab.

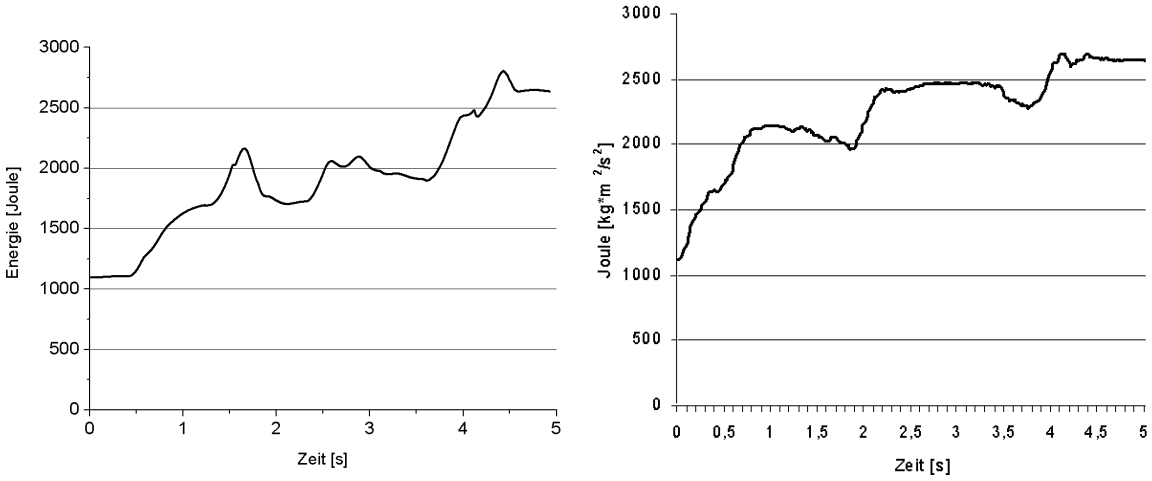


Abbildung 7.10: Energieverläufe. Links simuliert; rechts aufgezeichnet.

Dies hat mehrere Gründe: In der Grafik zum simulierten Fall ist die potentielle Energie der ausgelenkten Reckstange mitberücksichtigt, wohingegen Dirk Haberkamp nur die kinetischen/rotatorischen und potentiellen Anteile der einzelnen Körperteile berücksichtigt. Ein weiterer Unterschied ist, dass im simulierten Fall auch die potentiellen Energien der Gelenkanschläge und Muskelspannungen berücksichtigt sind, was im aufgezeichneten Fall nicht geschah (Es existierte ja auch kein biomechanisches Menschmodell!). Insofern ist der direkte Vergleich der beiden Gesamtenergieverläufe mit Einschränkungen möglich. Dennoch erkennt man sehr gut, dass mit der Körperstreckung (großer KSP-Abstand zur Reckstange) immer eine Reduzierung der Gesamtenergie einhergeht. Dies resultiert zum einen in dem dissipativen Anteil der Reckstangendeformation, aber auch in der Tatsache, dass aktiv Momente aufgebracht werden müssen (die die Bewegung bremsen!) um in eine flüssige Bewegung zu gelangen. Zur Verdeutlichung ist in Abbildung 7.11 die Gesamtenergie und der KSP-Abstand der Simulation in einer Grafik dargestellt.

Besonders interessant ist auch die Kenntnis der Belastung des Turners während der Übung. Beim Reckturnen ist die Kraft mit der der Turner an der Reckstange zieht ein wichtiges Kriterium, denn wird eine bestimmte Maximalkraft (abhängig vom Turner) überschritten, kann sich der Turner nicht mehr an der Stange halten. Die Stangenkraft sollte in Phasen der Körperstreckung und beim Durchqueren des tiefsten Punkts am stärksten sein. In Abbildung 7.12 sieht man den Verlauf der Stangenkraft der simulierten Riesenfelge zusammen mit dem KSP-Abstand zur Reckstange.

Der erwartete Kraftverlauf wird hier recht gut bestätigt. In den Phasen der gestreckten Körperhaltung erreicht die Kraft ein Maximum am untersten Punkt. Desweiteren erkennt man auch

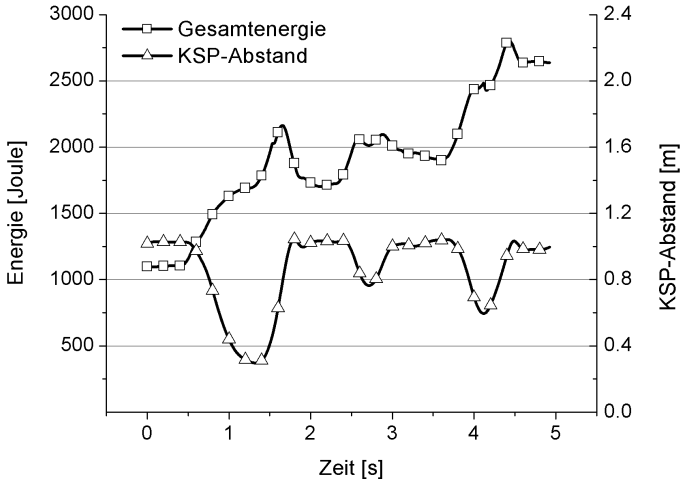


Abbildung 7.11: Zeitlicher Verlauf der Gesamtenergie und des KSP-Abstandes.

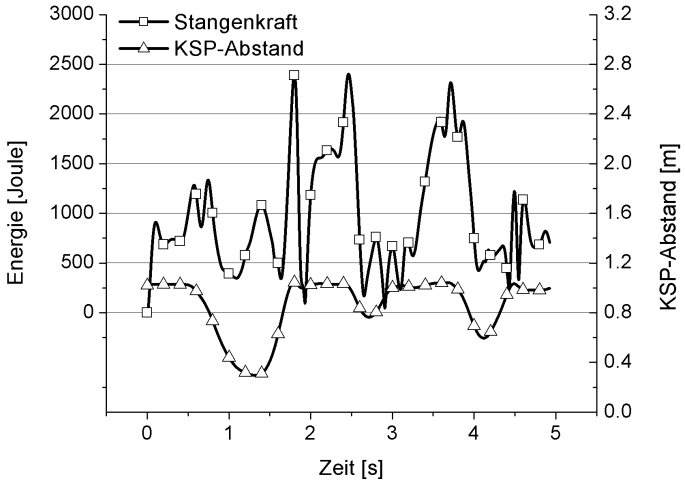


Abbildung 7.12: Zeitlicher Verlauf des KSP-Abstands und der Stangenkraft.

eine Stangenkrafterhöhung in den Phasen der Körperstreckung. Interessant ist, dass schon beim Aufschwung kurzzeitig Kräfte von bis zu $2500N$ erreicht werden; dies entspricht nahezu dem 3.5-fachen Körpergewicht des Turners. Bei Spitzensportlern kann diese Kraft durch nachfolgende, schneller geturnte Riesenfelgen bis auf das 7- bis 8-fache des Körpergewichts ansteigen. In den ersten 0.5 Sekunden der Simulation hing der Turner noch bewegungslos an der Reckstange. Sehr gut erkennt man, dass in diesem Fall die Reckstange genau mit der Gewichtskraft des Turners belastet wird ($754N \equiv 76.9kg$). Dass die Stangenkraft in den ersten $0.2s$ über die $750N$ ansteigt, resultiert aus der eingestellten Anfangssituation: Der Turner ist anfangs an einer unausgelenkten Reckstange befestigt. Dieses System muß sich natürlich erst einmal einschwingen.

Nun ist es natürlich interessant zu wissen, mit welchen Kräften und Momenten der simulierte Turner diese Riesenfelge geturnt hat. Als Maximalmomente wurden für das Schultergelenk in der Flexion und Extension jeweils $180Nm$ eingestellt. Für das Hüftgelenk wurden $130Nm$ in

der Extension und $180Nm$ in der Flexion verwendet. Dies entspricht einem gut trainierten Sportler, liegt aber noch deutlich unter den Maximalmomenten eines Spitzenturners. Dabei erfolgte der Momentaufbau nicht instantan, sondern mittels der Differentialgleichung 6.1. Die Maximalmomente werden allerdings nur bei einer bestimmten Winkelstellung erreicht und fallen dann wie in den Grafiken 6.8 und 6.9 dargestellt ab. Die folgende Grafik zeigt den Verlauf der verwendeten Momente im Hüft- und Schultergelenk über den gesamten Bewegungsablauf.

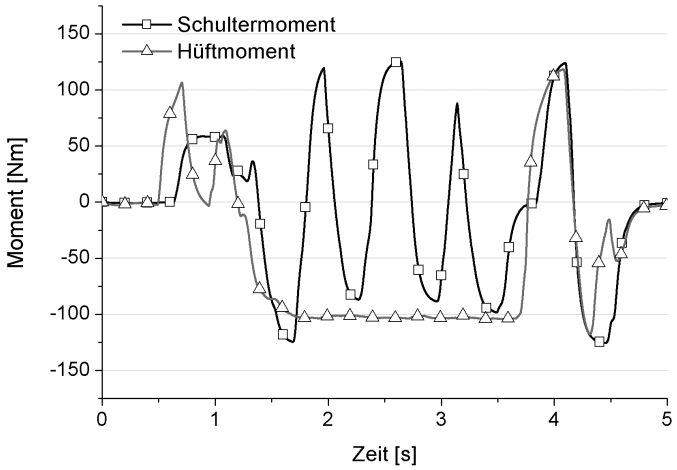


Abbildung 7.13: Zeitlicher Verlauf der Momente im Hüft- und Schultergelenk.

Beim Aufschwung erkennt man sehr gut, dass zuerst die Hüfte gehoben wird bevor der Arm-Schulter Winkel verkleinert wird. Nur auf diese Weise kommt man in eine fließende Bewegung. Wohlgemerkt kann man auch mit anderen Momenten in diese Positionen gelangen jedoch vom Standpunkt der Energieeffizienz aus gesehen ist der obige Momentverlauf optimal.

7.3 Zusammenfassung

Am Beispiel des Reckturners wurde in den vorherigen Kapiteln dargestellt wie leistungsfähig und universell das in dieser Arbeit entwickelte Verfahren ist. Hierbei hatte man während und nach der Simulation freien Zugang zu allen wichtigen Simulationsgrößen wie KSP-Abstand, Gesamtenergie, Stangenkraft sowie den Gelenkmomenten.

Wünschenswert wäre es nun dieses Verfahren zur Rekonstruktion von beliebigen Bewegungsabläufen einzusetzen. Hierzu ist allerdings der Einsatz von geeigneten Optimierungsverfahren unumgänglich. Allerdings müssen auch die Optimierungsverfahren mit geeigneten Anfangswerten gestartet werden, um in einer vernünftigen Zeit eine Lösung zu erhalten. Die mittels inverser Dynamik rekonstruierten Momente haben zwar wenig mit den in den Gelenken tatsächlich auftretenden Momenten gemeinsam, jedoch wäre es denkbar die Momente über einen bestimmten Zeitraum zu mitteln und als Anfangswerte dem Optimierungsverfahren zu liefern. Kontrollalgorithmen sind eine weitere Möglichkeit, hier ist das Aufstellen von guten Algorithmen für komplexe Bewegungsabläufe aber sehr kompliziert.

Kapitel 8

PKW-Unfall Simulation

Als abschließender Anwendungsfall soll in diesem Kapitel die Simulation und Parameteranalyse von realen Fußgänger-PKW Unfällen erfolgen. Über den Nutzen und die Bedeutung einer computergestützten PKW-Unfallsimulation braucht hier wohl nicht näher eingegangen zu werden. Als Hauptkriterium dafür, wie hoch das Verletzungsrisiko bei einem bestimmten PKW-Typ ist, werden aufwändige Dummyversuche unternommen und aus den so gewonnenen Erkenntnissen und Ergebnissen wird oftmals auf die globale Sicherheit eines PKW insgesamt geschlossen [1]. Neben der schwer abzuschätzenden Übertragbarkeit auf reale Fußgänger-PKW Unfälle ist das Hauptproblem der Crashversuche mit Dummys das Fehlen einer breit gefächerten Parameteranalyse. Der Crashdummy wird oft nur ein einziges Mal mit einer gewissen Geschwindigkeit angefahren und danach werden die Belastungen der einzelnen Körperteile ausgewertet. Hiernach ist dann oftmals aber schon Schluß! Die Frage, ob bei einer Wiederholung des Versuchs mit einer gleichen oder leicht modifizierten Anfangsstellung von Fußgänger oder PKW der Crashversuch gleiche oder ähnliche Ergebnisse liefert, bleibt hierbei völlig unbeantwortet. Die so gewonnenen Ergebnisse lassen daher nur eine begrenzte Übertragung auf das Verletzungsrisiko für einen bestimmten PKW-Typ zu. Wie noch gezeigt wird stellen die in dieser Arbeit gewonnenen Erkenntnisse aus Simulationsläufen die Aussagekräftigkeit eines einzelnen Dummyversuchs stark in Zweifel, denn schon leicht voneinander abweichende Anfangsstellungen erzeugen stark unterschiedliche Belastungen. So kann bei konstanter PKW Geschwindigkeit das Verletzungsrisiko des Kopfs je nach Stellung des Fußgängers um den Faktor 3 variieren und somit einen Unterschied zwischen Tod und Leben bedeuten. Die entscheidende Frage ist natürlich inwiefern sich diese Ergebnisse überhaupt bei einer Wiederholung des Crashversuchs reproduzieren lassen und inwiefern die Ergebnisse von den Anfangsbedingungen abhängen. Mit Dummyversuchen ist dies jedoch mit einem kaum vertretbaren Aufwand verbunden. Genau hier sind die computergestützten Simulationen von unschätzbarem Wert.

Als weitere Demonstration der Leistungsfähigkeit des in dieser Arbeit entwickelten Verfahrens soll für bestimmte Parameter eine Sensitivitätsanalyse einer Unfallsimulation erfolgen. Unter Sensitivitätsanalyse ist hier die Variation von Parametern über einen großen Bereich gemeint

[1] Neben der PKW-Insassen Sicherheit ist dies auch ein gern verwendetes Außhängeschild der Autohersteller. Von nur wenigen einfachen Crashdummyversuchen jedoch auf die globale Sicherheit des PKW zu schließen wird ist natürlich mehr als fraglich.

und nicht die Sensitivität eines Parameters während eines einzelnen Simulationsablaufs. Hierbei wird nicht ein spezieller, authentischer Unfall rekonstruiert, sondern das Verletzungsrisiko soll in Abhängigkeit von bestimmten Parametern ermittelt werden. Aufgrund der Interaktivität der Simulation lässt sich ein verhältnismässig großer Parameterbereich untersuchen, da je nach Integrationsgenauigkeit ($AbsTol = 1e^{-7}$) ein Simulationslauf weniger als 15 Sekunden benötigt. Aufgrund der durch Kollisionen hervorgerufenen hohen Kräfte und der starken Änderung dieser Kräfte in einem kurzen Zeitraum sind natürlich nicht die gleiche Simulationsgeschwindigkeit wie beim Reckturner zu erwarten, obwohl die Größe beider Systeme ähnlich ist.

8.1 Simulationsszenario

Bei der Parameteranalyse von Fußgänger-PKW Unfallsimulationen stellt sich natürlich die Frage was für Ergebnisse man überhaupt auswerten soll. Von zentraler Bedeutung ist natürlich, ob ein Unfall tödlich verläuft oder nicht. Genau dies soll in diesem Kapitel untersucht werden. Als Haupttodesursache bei Fußgänger-PKW Unfällen ist neben der Aortaruption [2] vor allem die Verletzung des Kopfs ausschlaggebend. Maßgeblich für eine zum Tode führende Kopfverletzung sind die auf den Kopf wirkende Kraft und deren Wirkungsdauer. Als Richtwert für eine kritische (mit hoher Wahrscheinlichkeit zum Tode führende) Verletzung gilt eine maximale Kopfbeschleunigung von 50 g bis 100 g, wenn diese länger als 2 bis $5ms$ besteht. Als Standard zur Klassifizierung von Kopfverletzungen hat sich die Definition des HIC (Head Injury Criteria), das von J. Versace [45] 1971 eingeführt wurde, bewährt. Das HIC ist definiert durch

$$HIC = MAX_{t_1,t_2}\left[(t_2 - t_1)\cdot\left[\frac{1}{t_2 - t_1}\cdot\frac{1}{9.81\frac{m}{s^2}}\cdot\int_{t_1}^{t_2} a(t)dt\right]^{2.5}\right]$$

Das HIC besitzt die Einheit Sekunde, die üblicherweise nicht mit angegeben wird. Hierbei heißt MAX_{t_1,t_2}, dass der Ausdruck in der eckigen Klammer für alle möglichen $0 \leq t_1 \leq T - t_2$ und $t_2 - t_1 \leq T$ bestimmt werden muß, wenn T die gesamte Simulationsdauer ist. Der HIC ist dann das Maximum all dieser Auswertungen. Anders als über einen Brute-Force Algorithmus ist dies nicht möglich. Es hat sich aber herausgestellt, dass das Zeitintervall $\Delta t = t_2 - t_1$ ausschließlich im Bereich $5ms < \Delta t < 20ms$ liegt. So gut wie in allen Fällen liegt das HIC auch in dem Zeitintervall, in dem auch die maximale Kraft auf den Kopf auftritt. Kritisch zum HIC könnte man anmerken, dass eine kleine Kraft, die über einen langen Zeitraum wirkt, den gleichen HIC Wert liefert wie eine große Kraft, die nur kurz wirkt und dies nicht zu den selben Verletzungen führt. Mediziner gehen aber davon aus, dass beide Fälle dieselben Kopfverletzungen hervorrufen. Unter dieser Annahme ist die Definition des HIC als Verletzungsrisiko natürlich gerechtfertigt. Was der HIC Wert natürlich nicht berücksichtigt sind die rotatorischen Beschleunigungen die in bestimmten Unfallsituationen nicht zu vernachlässigen sind. Hierfür hat sich allerdings noch kein verifizierbarer Standard etabliert. Als Richtwert nimmt man für einen HIC Wert größer 1000 an, dass dies zu einer

[2] Die Aortaruption ist ein Riss in der Wand der Aorta, die unweigerlich zu inneren Blutungen führt.

tödlichen Verletzung führt, ein HIC Wert kleiner 1000 führt in 50 $\%$ der Fälle zum Tode. Die folgende Grafik zeigt exemplarisch den zeitlichen Verlauf der Kopfbeschleunigung bei einer Unfallsimulation mit einer PKW-Geschwindigkeit von $10m/s$ und einem Aufprall direkt von hinten, der Fußgänger hat dem Auto den Rücken zugekehrt. Siehe rechtes Bild in Abbildung 8.2.

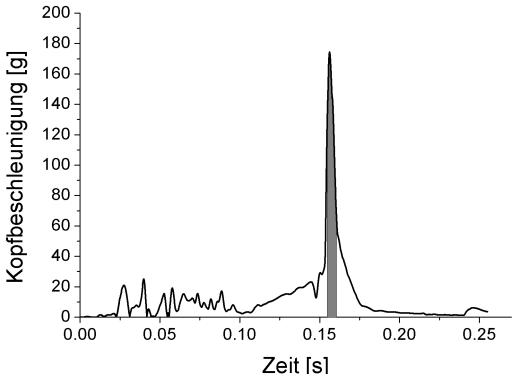


Abbildung 8.1: Zeitlicher Verlauf der Kopfbeschleunigung. Die graue Fläche unter der Kurve entspricht dem Intervall in dem der HIC berechnet wird.

Den HIC Wert als Vergleichskriterium heranzuziehen ist wie bereits erwähnt nicht ganz unproblematisch. So steckt in diesem Wert ja schon der Kraft-Deformationszusammenhang der nur eine Näherung darstellt, er ist aber als Vergleichskriterium der Simulationen untereinander sehr gut geeignet. Als weiteres (rein intuitiv mit der Verletzung korrelierendes) Kriterium wird auch die senkrechte Relativgeschwindigkeit v_{rel-n} zwischen Kopfschwerpunkt und Aufschlagstelle am PKW zum Zeitpunkt des ersten Kontakts herangezogen. Damit hat man ein Mass, das unabhängig vom verwendeten Kraftgesetz ist.

Nun lassen sich natürlich eine Vielzahl an Parametern variieren, angefangen von der Anfahrgeschwindigkeit, Anfahrrichtung und Geometrie des PKW sowie den individuellen anthropometrischen Daten wie zum Beispiel Körpergröße und Körpergewicht bis hin zu unterschiedlichen Anfangsstellungen des Fußgängers. Interessant ist ferner auch eine Variation der Stoßparameter beim Kraft-Deformationszusammenhang zwischen Fußgänger und PKW, der ja die größte Näherung in der Simulation darstellt.

Im Rahmen dieser Arbeit beschränke ich mich jedoch auf die Analyse von nur vier Parametern, die aber zugleich eine zentrale Rolle beim Verletzungsrisiko spielen. In einer ersten Testreihe wird untersucht, inwiefern das HIC und v_{rel-n} abhängig sind von den Kraft- und Dämpfungskonstanten im Kraft-Deformationszusammenhang zwischen Fußgänger und PKW. Die Anfahrgeschwindigkeit und die Position des Fußgängers bleiben hierbei konstant. Eine weitere Testreihe untersucht die Kopfbelastung in Abhängigkeit von der Anfahrgeschwindigkeit des PKW sowie der Fußgängerposition. Die Geschwindigkeit des PKW wird in Schritten von $3m/s$ von $7m/s$ bis $13m/s$ variiert. Dies ist in etwa der Geschwindigkeitsbereich in dem zum schon mit tödlichen Verletzungen zu rechnen ist. Bei der Position des Fußgängers wird der Winkel

zum Auto in 1 Grad Schritten von 0 bis 360 Grad verändert. Der Fußgänger ist dabei in einer stehenden Position und in Ruhe. Zusätzlich werden beide Testreihen mit zwei unterschiedlichen PKW Geometrien durchgerechnet. Die folgenden Bilder zeigen die Ausgangspositionen sowie die verwendeten Fahrzeuggeometrien.

Abbildung 8.2: Anfangsstellung zur Untersuchung der Kopfbelastung bei unterschiedlichen Kraftkonstanten.

Das Auto im linken Bild wird fortan als PKW A und das im rechten Bild als PKW G bezeichnet. Die Geometrien dieser beiden PKW sind sehr verschieden. Die nachfolgende Grafik zeigt deren Seitenansicht.

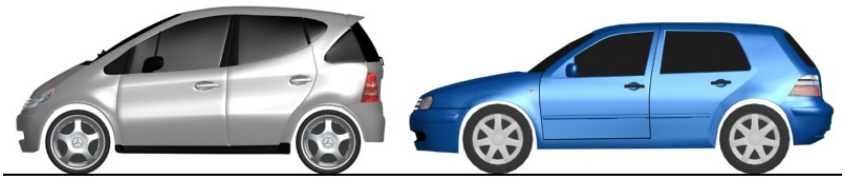


Abbildung 8.3: Seitenansicht von PKW A (links) und PKW G (rechts).

Für die anthropometrischen Daten des Fußgängers wurden die gleichen Größen benutzt, die auch in der Simulation des Reckturners verwendet wurden. Dies entspricht recht gut einem durchschnittlichen europäischen Mann.

Um eine bessere Vergleichsmöglichkeit mit bestehenden Simulationen zu erhalten wird der Kraft-Deformations Zusammenhang aus der Arbeit von Valentin Keppler [5] verwendet. Valentin Keppler nimmt eine quadratische Kraft-Deformations Relation an, die die Belastungen im Vergleich zu authentischen Unfällen sehr gut reproduziert.

$$F_{coll} = C_K \cdot d^2 + C_D \cdot d \cdot \dot{d}$$

Hierbei ist d der Abstand zwischen dem PKW und einem Körperteil und $\dot{d}$ die relative Geschwindigkeit der Kollisionspartner zueinander.

8.2 Simulationen

8.2.1 Variation der Kraft- und Dämpfungskonstanten

In der ersten Testreihe wird die Abhängigkeit der Kopfbelastung bei verschiedenen Kraft- und Dämpfungskonstanten zwischen Fußgänger und PKW untersucht. Die Anfahrgeschwindigkeit des PKW beträgt $8m/s$ und der Fußgänger wird in einer ruhenden und stehenden Position direkt von hinten angefahren, so wie es in Abbildung 8.2 dargestellt ist. Die Federkonstante wurde von $C_K = 200000$ bis $1200000N/m^2$ in Schritten von $25000N/m^2$ variiert. Die Dämpfungskonstante von $C_D = 5000$ bis $35000N/ms$ in Schritten von $10000N/ms$.

Um eine hohe Genauigkeit der Simulation zu erzielen, wurde durchgehend mit einer absoluten Toleranz von $1e^{-7}$ gerechnet und zusätzlich wurde nach jedem erfolgreichen Integrationsschritt die Jacobimatrix neu bestimmt. Bei solch hohen Genauigkeitsanforderungen ist ein Verfahren mit höherer Ordnung als RADAU5 jedoch effizienter. Deshalb wurde das Programm RADAU, das auch von den RADAU5 Autoren entwickelt wurde, verwendet, das eine variable Ordnung der impliziten Runge-Kutta Verfahren bis zur maximalen Ordnung von 13 zulässt. Dies beschleunigte die Integration um den Faktor 2.

In den folgenden Abbildungen 8.4 und 8.5 sind jeweils der HIC Wert sowie die Relativgeschwindigkeit v_{rel-n} in Abhängigkeit der Kraft- und Dämpfungskonstanten C_K und C_D dargestellt. Die ersten Grafiken beziehen sich auf PKW G. Charakteristisch für diese PKW Geometrie sind die verhältnissmässig flache Motorhaube und die daran anschließende etwas steilere Windschutzscheibe.

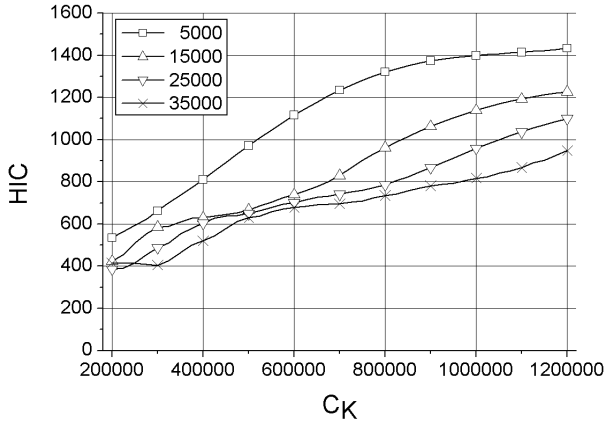


Abbildung 8.4: PKW G. Abhängigkeit des HIC von C_K und C_D. Die einzelnen Kurven entsprechen unterschiedlichen C_D Werten.

Man erkennt sehr deutlich, dass mit abnehmender Dämpfung und höheren Federkonstanten C_K der HIC Wert ansteigt. Überraschend allerdings ist, dass die Relativgeschwindigkeit v_{rel-n} abnimmt, das heißt, obwohl der Kopf mit einer geringeren Geschwindigkeit auf dem PKW landet resultiert dies in einer größeren Belastung. Dass die Relativgeschwindigkeit v_{rel-n} abnimmt erklärt sich durch die stärkere Beschleunigung des Fussgängers (und somit auch des

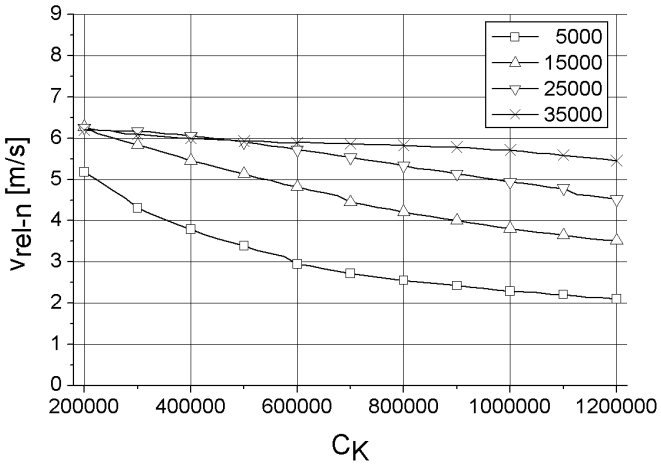


Abbildung 8.5: PKW G. Abhängigkeit von v_{rel-n} von C_K und C_D. Die einzelnen Kurven entsprechen unterschiedlichen C_D Werten.

Kopfs) bei einer härteren und schwächer gedämpften PKW Front. Der Kopf bewegt sich also schneller in Fahrtrichtung des PKW und trifft somit auch mit einer niedrigeren Geschwindigkeit auf. Allerdings ist aufgrund des stärkeren Kraftgesetzes der HIC Wert größer trotz der niedrigeren Aufprallgeschwindigkeit. Man erkennt hier schon sehr deutlich den diffiziellen Zusammenhang der einzelnen Parameter. In Abbildung 8.6 ist zur Verdeutlichung noch die absolute Geschwindigkeit v_{Abs} des Kopfs aufgetragen, die wie beschrieben bei härteren C_K und schwächerer Dämpfung ansteigt.

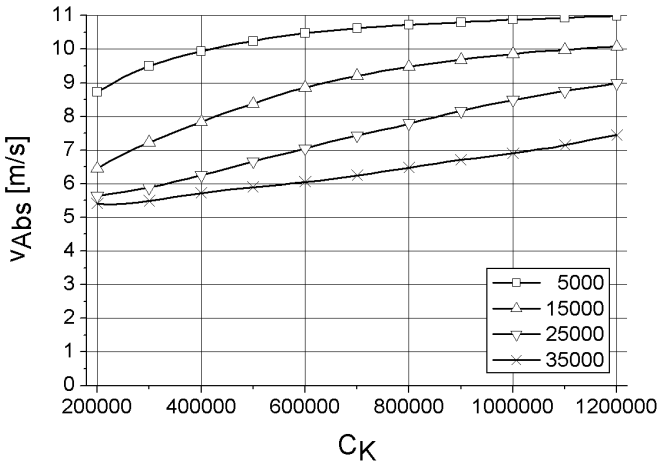


Abbildung 8.6: PKW G. Abhängigkeit der absoluten Kopfgeschwindigkeit v_{Abs} von C_K und C_D. Die einzelnen Kurven entsprechen unterschiedlichen C_D Werten.

Interessant ist noch anzumerken, dass sich mit steigendem C_K die Aufprallstelle des Kopfs immer mehr nach vorne verschiebt. Anfänglich ($C_K = 200000$) liegt die Position im Ansatz von der Motorhaube zur Windschutzscheibe und wandert bei $C_K = 1.2e^6$ im schwach gedämpften Fall ($C_D = 5000$) um $20cm$ und im stark gedämpften Fall ($C_D = 35000$) um

$12cm$ nach vorne.

Als nächstes folgt die Auswertung von PKW A. Die Charakteristik des PKW A liegt in der konstanten Neigung der gesamten PKW Front, die insgesamt steiler verläuft als bei PKW G. Im Gegensatz zu PKW G liegen die Aufschlagstellen des Kopfs für alle C_K und C_D immer auf der Windschutzscheibe.

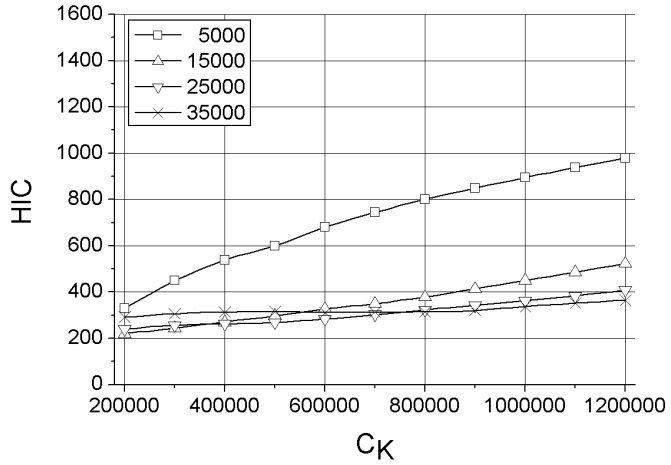


Abbildung 8.7: PKW A. Abhängigkeit des HIC von C_K und C_D. Die einzelnen Kurven entsprechen unterschiedlichen C_D Werten.

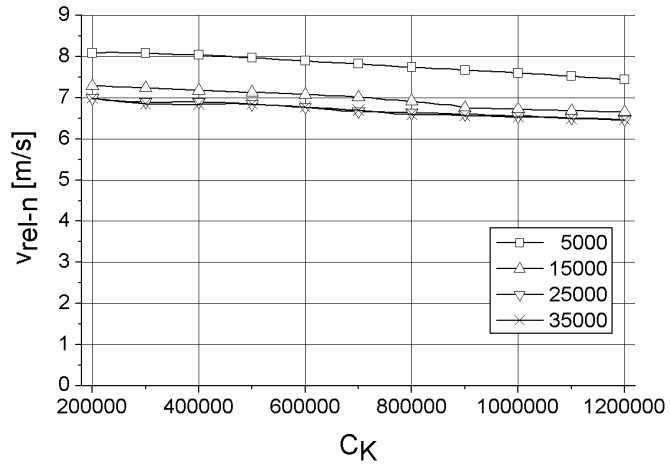


Abbildung 8.8: PKW A. Abhängigkeit von v_{rel-n} von C_K und C_D. Die einzelnen Kurven entsprechen unterschiedlichen C_D Werten.

Qualitativ ist wieder das gleiche Verhalten zu erkennen wie schon bei PKW G: Bei schwacher Dämpfung und hohen Kraftkonstanten steigt der HIC Wert, aber gleichzeitig beobachtet man wieder eine Abnahme der Relativgeschwindigkeit v_{rel-n}. Interessant ist, dass in diesem Fall die Relativgeschwindigkeiten v_{rel-n} im Moment des Aufpralls des Kopfs um $20 - 30\%$

höher liegen als bei PKW G. Der Vollständigkeit halber ist in Abbildung 8.9 die absolute Kopfgeschwindigkeit in Abhängigkeit von C_K und C_D für PKW A aufgetragen.

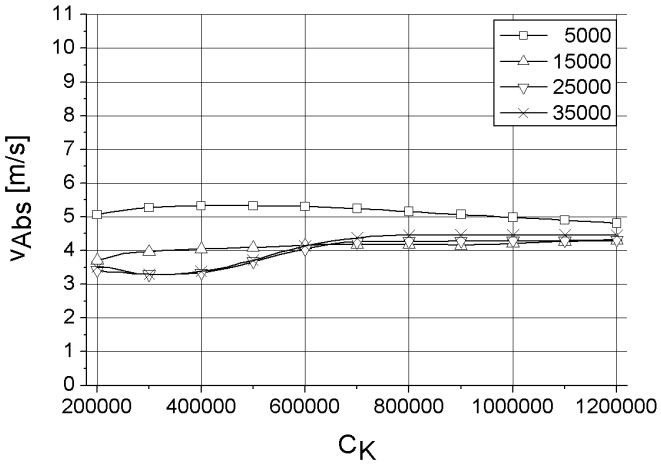


Abbildung 8.9: PKW A. Abhängigkeit der absoluten Kopfgeschwindigkeit v_{Abs} von C_K und C_D. Die einzelnen Kurven stehen für unterschiedliche C_D.

Auch bei PKW A wandern die Aufprallstellen bei höheren C_K zur PKW Front, allerdings nicht in dem gleichen Masse. Der größte Abstand der Aufprallstellen liegt hier bei $15cm$.

Im direkten Vergleich der beiden PKW Geometrien erkennt man sehr gut, dass PKW G ein durchweg höheres HIC besitzt als PKW A, wohlgemerkt unter den hier vorliegenden Anfangsbedingungen. Allerdings liegt die Aufprallstelle des Kopfs beim PKW G in den meisten Fällen auf der Motorhaube und nicht auf der Windschutzscheibe. Inwiefern sich hierdurch das Verletzungsrisiko ändert muß natürlich genauer untersucht werden. Ob die Aussage, dass PKW A ein geringeres Kopfverletzungsrisiko als PKW G besitzt, auch bei variierender PKW-Geschwindigkeit und Fußgängerposition Gültigkeit hat, soll die nächste Testreihe zeigen.

8.2.2 Variation PKW-Geschwindigkeit/Fußgängerposition

In der zweiten Testreihe wird die PKW-Geschwindigkeit v_{Car} und die Fußgängerposition α variiert. Für v_{Car} wurden die Werte $7, 10$ und $13m/s$ durchgerechnet. Die Fußgängerposition wurde in 1 Grad Schritten von 0 bis 360 Grad modifiziert, der Fußgänger wurde dabei um seine Hochachse rotiert. Für das Kraftgesetz wurden nun die konstanten Werte $C_K = 400000N/m^2$ und $C_D = 15000N/ms$ verwendet. Bei dieser Testreihe spielte die Implementierung des Schultergelenks eine wichtige Rolle. Denn gerade bei Unfällen mit 90 ± 10 Grad bzw. 270 ± 10 Grad ist die Art und Weise wie das Schultergelenk realisiert wird entscheidend dafür, ob der Kopf auf den PKW aufschlägt oder der Arm als *Polster* dazwischen liegt (Wohlgemerkt nur unter der Annahme, dass die Arme anfänglich gerade am Körper herabhängen). Diese Bereiche sind deshalb mit Vorsicht zu interpretieren, da sie schon bei leicht geänderten Anfangsbedingungen völlig andere Ergebnisse liefern. Außerhalb dieser Winkelbereiche ist dies weniger kritisch.

In Abbildung 8.10 sieht man wieder den HIC Wert sowie v_{rel-n} für PKW G. Aufgrund der Symmetrie der Simulation sollten die Ergebnisse bei α Grad und $360 - \alpha$ Grad gleich sein. Da das 3D-Modell des Menschen allerdings nicht perfekt spiegelsymmetrisch ist sind leichte Abweichungen zu erkennen.

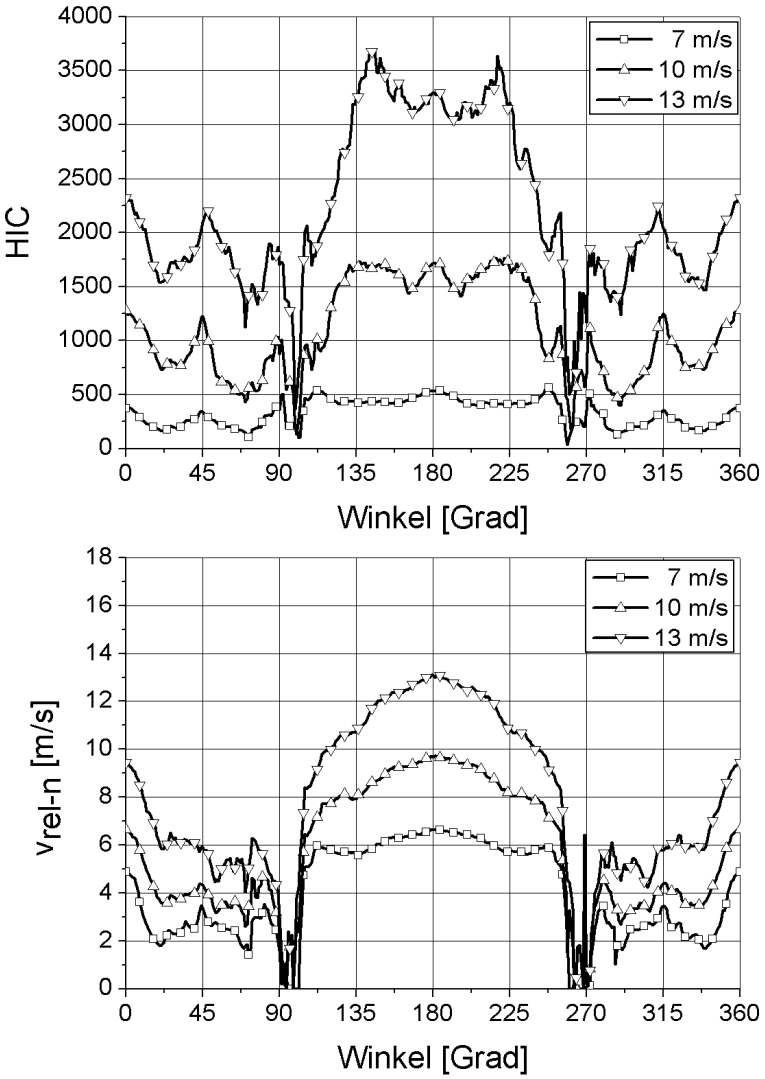


Abbildung 8.10: PKW G. Abhängigkeit von HIC und v_{rel_n} von v_{Car} und α. Die unterschiedlichen Kurven entsprechen verschiedenen Anfahrgeschwindigkeiten.

In dieser Testreihe sind zwei Dinge besonders hervorzuheben: Zum einen ist dies der extreme Unterschied beim HIC Wert für Anfahrten direkt von vorne und von hinten. Wird der Fußgänger beim Anprall direkt von vorne erwischt (Gesicht Richtung PKW) so ist das Verletzungsrisiko wesentlich höher als für Anfahrten von hinten. Bei hohen Geschwindigkeiten ist hier ein Unterschied um den Faktor 2 bis 3 zu beobachten. Hervorgerufen wird dies durch einen Peitscheneffekt, der vor allem aus den Gelenkanschlägen im Knie- und Hüftgelenk resultiert, die bei einem Anprall direkt von vorne wesentlich steifer sind als bei einem Anprall von hinten.

In Abbildung 8.11 sieht man die gleiche Testreihe für PKW A.

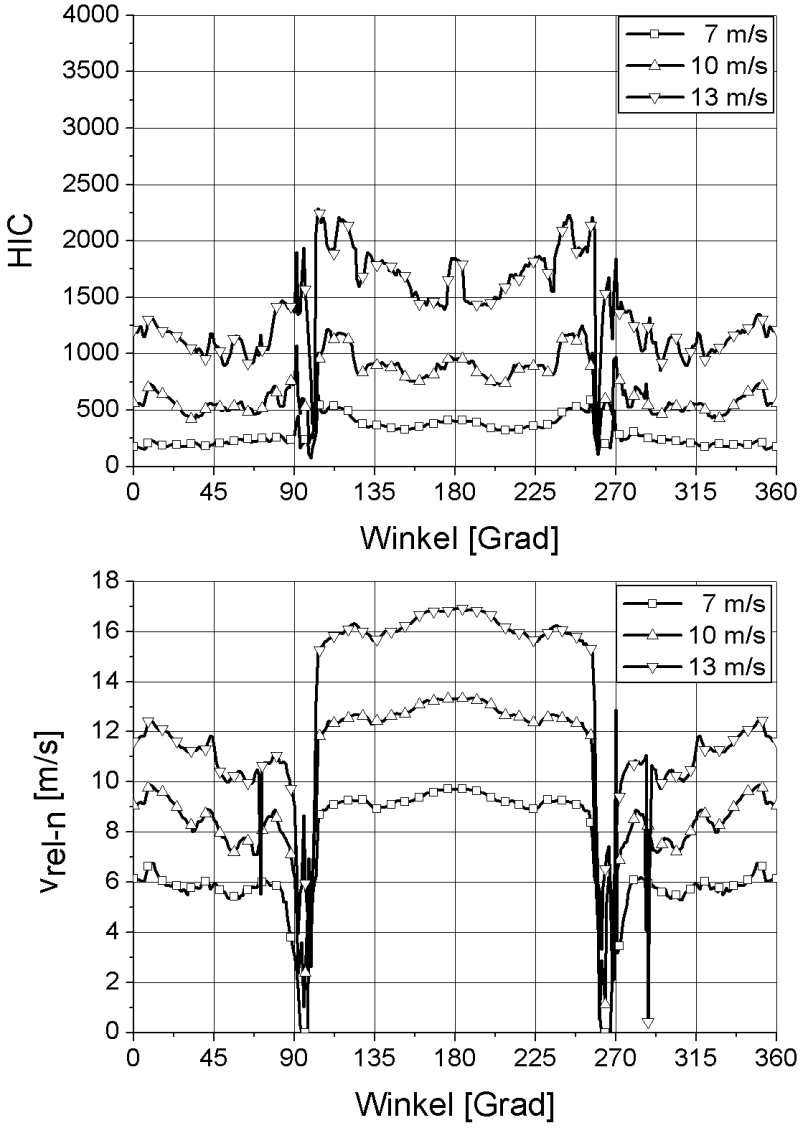


Abbildung 8.11: PKW A. Abhängigkeit von HIC und v_{rel_n} von v_{Car} und α. Die unterschiedlichen Kurven entsprechen verschiedenen Anfahrgeschwindigkeiten.

Hier bestätigt sich das zuvor bereits deutlich erkennbare geringere Verletzungsrisiko bei PKW A. Der Hauptgrund liegt in allen Fällen in der geringeren Kopfaufprallgeschwindigkeit kurz vor der Kollision. Dies liegt vor allem daran, dass der Weg den der Kopf bei PKW G zurücklegt bis es zum Kontakt kommt wesentlich größer ist als bei PKW A und er somit auch länger beschleunigt wird. Wohl gemerkt betrifft dies lediglich das Verletzungsrisiko des Kopfs. Im Rahmen dieser Arbeit kann aber nicht untersucht werden wie hoch das Risiko für andere Todesursachen, wie zum Beispiel durch Aortaruption, bei beiden PKW liegt.

In abschliessenden Bildersequenzen wird exemplarisch der Verlauf eines Unfalls mit den bei-

den PKW Geometrien gezeigt. Die Anfahrgeschwindigkeit des PKW betrug $10m/s$ und der Winkel α des Fußgängers zum PKW Betrug 130 Grad. Wie man den Grafiken zum HIC Wert entnehmen kann wären beide Unfälle mit hoher Wahrscheinlichkeit tödlich verlaufen, wobei PKW G ein um die Hälfte geringeren HIC Wert hat (PKW G: $HIC = 880$, PKW A: $HIC = 1650$). Jedes Einzelbild entspricht dabei einem Simulationsschritt von $5ms$ so dass man insgesamt eine Sequenz von $45ms$ sieht.

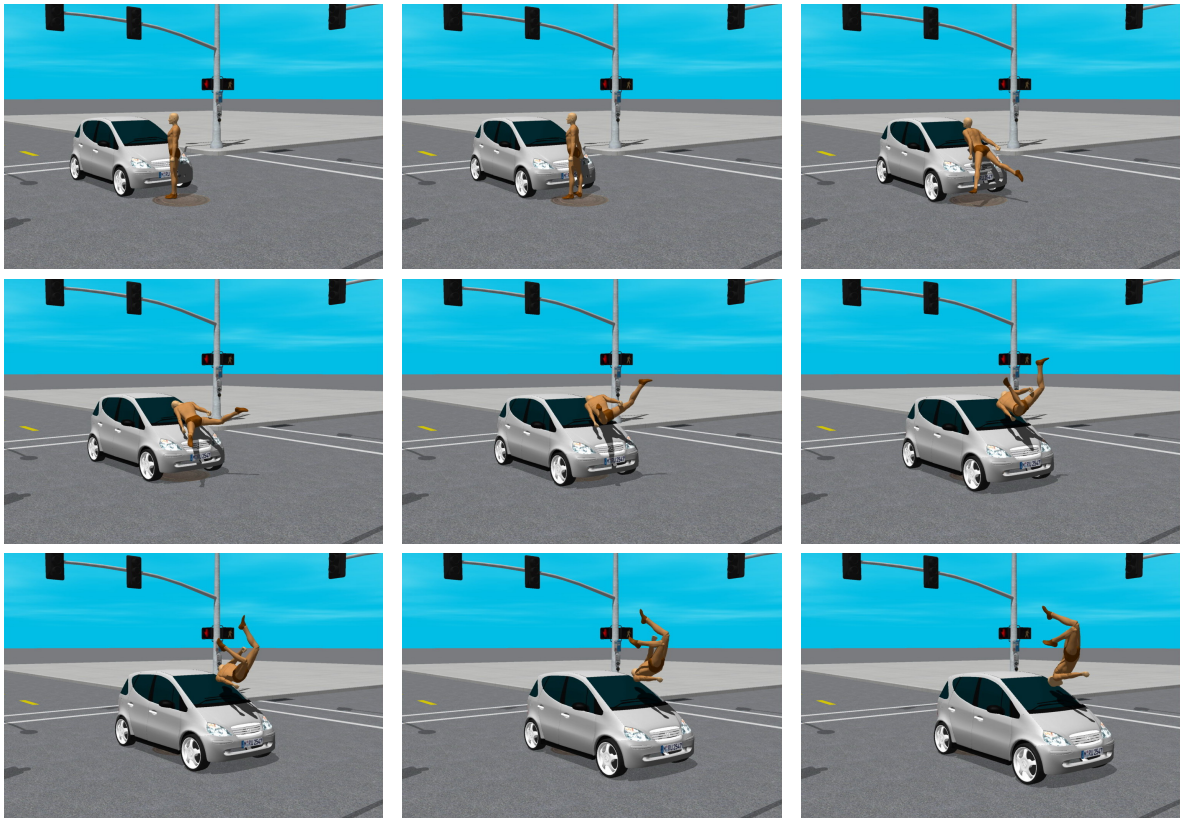


Abbildung 8.12: Bildsequenz zur Fußgänger-PKW Unfallsimulation für PKW A. Jedes Bild entspricht einem Zeitschritt von $5ms$. Die Anfahrgeschwindigkeit betrug $10m/s$. Der Fußgänger hatte einen Winkel α von 130 Grad zum PKW.

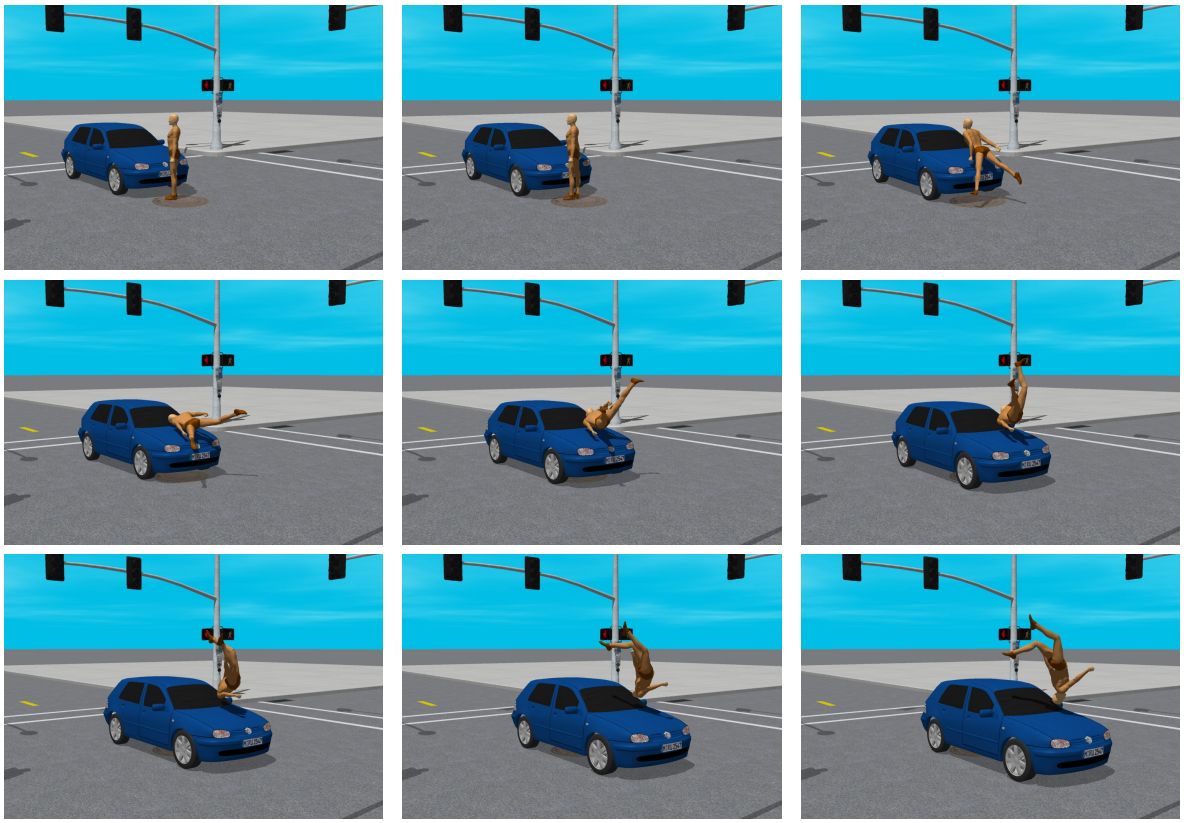


Abbildung 8.13: Bildsequenz zur Fußgänger-PKW Unfallsimulation für PKW G. Jedes Bild entspricht einem Zeitschritt von $5ms$. Die Anfahrgeschwindigkeit betrug $10m/s$. Der Fußgänger hatte einen Winkel α von 130 Grad zum PKW.

8.3 Zusammenfassung

In diesem Kapitel wurde eine einfache Sensitivitätsanalyse für ausgewählte Parameter bei Fußgänger-PKW Unfällen durchgeführt und damit die prinzipielle Machbarkeit von ausgiebigen Parameterstudien mit diesem Verfahren gezeigt. Bewußt wird hierbei aber auf irgendwelche globalen Schlußfolgerungen zur Sicherheit der einzelnen PKW verzichtet, denn hierzu müssen in weiteren Testreihen mit wesentlich größeren Parameterbereichen die oben gewonnenen Resultate bestätigt werden. Unter anderem ist auch noch zu klären inwiefern die anfängliche Verspannung der Muskeln kurz vor dem Unfall das Verletzungsrisiko verändert. Hierzu liegen generell noch so gut wie keine Erkenntnisse aus der Realität vor, es kann aber davon ausgegangen werden, dass Muskelverspannungen eine nicht zu vernachlässigende Rolle spielen. Dies könnte zum Beispiel durch einen weiteren Parameter realisiert werden, der die Stärke der Muskelverspannung verändert und dieser Parameter wird dann über einen großen Bereich variiert. Desweiteren ist wie bereits erwähnt in bestimmten Anfangsstellungen eine bessere Implementierung des Schultergelenks wünschenswert, da hierdurch der Ausgang der Unfälle von der Seite maßgeblich beeinflußt wird.

Kapitel 9

Abschluß/Ausblick

In den vorangegangenen Kapiteln wurde schrittweise die Realisierung eines schnellen und interaktiven Simulationsverfahrens für Mehrkörpersysteme beschrieben und dies speziell für biomechanische Simulationen mit Menschen. Dabei wurden spezielle Strukturen der Bewegungsgleichungen entwickelt und ferner dargestellt wie sich diese schnell integrieren lassen. Auch in der Kollisionserkennung wurden wichtige Neuerungen eingeführt ohne die keine interaktiven Geschwindigkeiten auf heutigen Rechnern möglich sind. Dabei spielt die Bestimmung der Ableitung des Zustandsvektors eine vernachlässigbare Rolle im Vergleich zum Aufwand des Integrationsverfahrens. Es kann also durchaus noch einiges mehr an Physik und Biomechanik in die Simulation gesteckt werden ohne die Ausführungsgeschwindigkeit stark zu reduzieren. Die Simulationsgeschwindigkeit ist hauptsächlich durch die Anzahl der verwendeten Körper bestimmt. Dabei wurde mit den zwei Anwendungsfällen der Vorwärtssimulation eines Reckturners und der Parameteranalyse von Fußgänger-PKW Unfällen demonstriert wie groß das Einsatzgebiet des Verfahrens ist.

Nun ist natürlich noch in vielen Bereichen der Simulation eine Verbesserung möglich und auch wünschenswert. Dies betrifft vor allem das Menschmodell. So ist besonders die Realisierung des Schultergelenks durch ein einfaches Kugelgelenk doch sehr vereinfachend. Gerade aber um die Vorhersagemöglichkeit bei seitlichen Fußgänger-PKW Unfällen zu verbessern, ist es unumgänglich hier ein besseres Gelenkmodell zu implementieren. Für bestimmte Simulationen (z.B. das Springen) ist ferner der Einfluß der Weichteile entscheidend. Mit einem einfachen Modell für Schwabbelmassen die mit Federn an die Starrkörper gekoppelt sind ließe sich so leicht das Modell erweitern.

Das größte Potenzial dieses Verfahrens liegt jedoch im Zusammenspiel mit einer Vorwärtssimulationen die entweder mit Kontrollalgorithmen, PID-Reglern oder auch per Hand Bewegungen synthetisiert und rekonstruiert. Aufgrund der Interaktivität ist es ferner ein universelles Werkzeug um schnell großflächige Parameteranalysen durchzuführen und um ein besseres *Gefühl* für bestimmte Bewegungen zu erlangen.

Anhang A

MATLAB Programm zum Stabpendel

Zur Simulation des Stabpendels wurde das Programm MATLAB [6] verwendet, mit dem man dieses System schnell aufstellen kann. MATLAB bietet verschiedenste Integrationsmethoden, um ein Differentialgleichungssystem zu integrieren. Zum Einsatz kamen die MATLAB eigenen Routinen ODE45 und ODE15s sowie das Verfahren RADAU5 von E. Hairer and G. Wanner. Das Programm besteht aus zwei Dateien: pendel2D.m ist das Hauptprogramm, hier werden die Variablen initialisiert, der initiale Zustandsvektor erzeugt und schließlich das Problem mit den Routinen ODE45, ODE15s und RADAU5 integriert. Das zweite Programm pendel2DForce.m berechnet die Ableitung des Zustandsvektors. Diese Funktion wird von den Routinen ODE45, ODE15s und RADAU5 aufgerufen.

```
% pendel2D.m
%
% this matlab program simulates the 2 dimensional movement under
% gravitational forces of a simple rigid pendulum with length 1.
% the mass of the pendulum is concentrated in a small sphere of
% radius 0.01. the pendulum is attached at one point with a stiff
% spring. to ensure numerical stability for the explicit integration
% scheme the stiff spring has a small damping term which is
% proportional to the velocity of the attachement point
%
clear;
tmax    = 4.0;        % integrate the system for 4 seconds
radius  = 0.01;       % radius of sphere
mass    = 1;          % mass of sphere
length  = 1;          % length of the pendulum
gravity = 9.81;       % gravitational constant
phi     = 20/180*pi;  % start angle of the pendulum
k       = -10000;     % force constant for the spring
d       = -0.01;      % damping constant for the spring

% rough estimates for the oscillation period of the pendulum
% and spring these values are not used in the simulation
tpendulum = 1/(2*pi*sqrt(length/gravity)*(1+(1/4)*sin(phi/2)^2))
```

```
tspring   = 1/(2*pi*sqrt(abs(k)/mass))

% the inertia of the pendulum is that of a sphere
% the rod of the pendulum is assumed to have zero mass
inertia = (2*mass/5)*radius*radius;

% start position for the pendulum
R = [-sin(phi)*(-1);   cos(phi)*(-1)];

% the initial state vector
%     velocity  position   angular momentum  rotation angle
Y0 = [0  0      R(1) R(2)  0                 phi];

% integrate system using the stiff solver ode15s
disp('ode15s')
tic
[t1,y1] = ode15s('pendel2Dforce', [0 tmax],Y0,
odeset('RelTol',1e-4,'Abstol',1e-4,'Stats','on'),
inertia,length,gravity,k,d);
toc

% integrate system using explicit runge kutta methods
disp('ode45')
tic
[t2,y2] = ode45('pendel2Dforce', [0 tmax],Y0,
odeset('RelTol',1e-4,'Abstol',1e-4,'Stats','on'),
inertia,length,gravity,k,d);
toc

% integrate system using radau5
disp('radau5')
options = odeset('RelTol',1e-4);
options = odeset(options,'AbsTol',1e-4);
options = odeset(options,'Stats','on');
options.Complex = 'off';
tic
[t3,y3] = radau5('pendel2Dforce', [0 tmax],Y0,
options,inertia,length,gravity,k,d);
toc

% plot angle versus time for all 3 methods
plot(t1,(180/pi)*y1(:,6),t2,(180/pi)*y2(:,6),t3,(180/pi)*y3(:,6));

legend('ode15s','ode45','radau5');
```

Die Funktion pendel2DForce berechnet die Ableitung des Zustandsvektors.

```
% pendel2Dforce.m
%
% this function returns the derivative of y
%
function dydt = f(t,y,flag,inertia,length,gravity,K,D)

% our state vector has 6 dimensions:
%   elements 1,2 are the x and y components of the
%     velocity of the center of mass
%   elements 3,4 are the position of the center of mass
%   element 5 is the angular momentum
%   element 6 the angle of the pendulum
dydt = zeros(6,1);

% compute the position and velocity of the point
% that is attached to the spring
a = [-sin(y(6))*length;  cos(y(6))*length];

% angular velocity ( w = I^(-1) * L )
w = (1/inertia) * y(5);

% absolute position and velocity of the attachement point
ap = y(3:4) + a(1:2);
av = y(1:2) + [-w*a(2); w*a(1)];

% acceleration exerted by the spring ( F = K*x + D*dx )
dydt(1:2) = K*ap + D*av;

% the change of position is velocity
dydt(3:4) = y(1:2);

% the change of angular momentum is torque ( T = Cross(r, L) )
dydt(5) = a(1)*dydt(2) - a(2)*dydt(1);

% change of angle is simply the angular velocity
dydt(6) = w;

% add gravitational acceleration
dydt(2) = dydt(2) - gravity;
```

Anhang B

Formeln zur Bestimmung der Masseeigenschaften

Die Auswertung der komplizierteren Flächenintegrale aus Kapitel 4.2 kann zum Beispiel mit Mathematica [7] durchgeführt werden. Folgendes Mathematica Programm realisiert dies:

```
(* ------------------------------------------------------- *)

(* Die Endpunkte des Dreiecks *)
a = {ax, ay, az};
b = {bx, by, bz};
c = {cx, cy, cz};

(* Berechnung durchfuehren fuer das folgende Vektorfeld *)
A = {x, 0, 0}

(* Normalenvektor *)
n = Cross[b - a, c - a];

(* Parametrisierung der Dreiecksflaeche *)
rx[t1_, t2_] := a + t1*(b - a) + t2*(c - a);

(* Substitution *)
F = ReplaceAll[A, {x -> rx[t1, t2][[1]],
   y -> rx[t1, t2][[2]],
   z -> rx[t1, t2][[3]]}];

(* Auswertung des Integrals *)
n.Integrate[F, { t1, 0, 1}, { t2, 0, 1 - t1}]

(* ------------------------------------------------------- *)
```

Für die in 4.2 gegebenen Vektorfelder werden jetzt die Integrale ausgewertet. Generell gilt es Integrale vom Typ

$$\frac{1}{2}\left((\boldsymbol{b}-\boldsymbol{a})\times(\boldsymbol{c}-\boldsymbol{a})\right)\int\limits_{\partial\Delta}\boldsymbol{A}dt_1dt_2$$

zu lösen. Im folgenden seien $\boldsymbol{a}$, $\boldsymbol{b}$ und $\boldsymbol{c}$ die Eckpunkte des Dreiecks und $\boldsymbol{n}$ ist gegeben durch $\boldsymbol{n}=(\boldsymbol{b}-\boldsymbol{a})\times(\boldsymbol{c}-\boldsymbol{a})$.

Für das Volumen V erhält man

$$V = n_x\cdot\tfrac{1}{6}\cdot(a_x+b_x+c_x) \qquad \boldsymbol{A}=(x,0,0)$$

Die x,y,z Komponenten des Schwerpunkts sind

$$\begin{aligned}
R_x &= n_x\cdot\tfrac{1}{24}\cdot({a_x}^2+{b_x}^2+{c_x}^2+a_xb_x+b_xc_x) && \boldsymbol{A}=(\tfrac{1}{2}x^2,0,0)\\
R_y &= n_y\cdot\tfrac{1}{24}\cdot({a_y}^2+{b_y}^2+{c_y}^2+a_yb_y+b_yc_y) && \boldsymbol{A}=(0,\tfrac{1}{2}y^2,0)\\
R_z &= n_z\cdot\tfrac{1}{24}\cdot({a_z}^2+{b_z}^2+{c_z}^2+a_zb_z+b_zc_z) && \boldsymbol{A}=(0,0,\tfrac{1}{2}z^2)
\end{aligned}$$

Und für die 6 Komponenten des Trägheitstensors erhält man

$$\begin{aligned}
I_{xx} &= n_y\cdot\tfrac{1}{60}(\quad a_ya_ya_y+a_ya_yb_y+a_ya_yc_y+a_yb_yb_y+a_yb_yc_y+\\
&\quad a_yc_yc_y+b_yb_yb_y+b_yb_yc_y+b_yc_yc_y+c_yc_yc_y)+\\
&\quad n_z\cdot\tfrac{1}{60}(\quad a_za_za_z+a_za_zb_z+a_za_zc_z+a_zb_zb_z+a_zb_zc_z+\\
&\quad a_zc_zc_z+b_zb_zb_z+b_zb_zc_z+b_zc_zc_z+c_zc_zc_z)\\
I_{yy} &= n_x\cdot\tfrac{1}{60}(\quad a_xa_xa_x+a_xa_xb_x+a_xa_xc_x+a_xb_xb_x+a_xb_xc_x+\\
&\quad a_xc_xc_x+b_xb_xb_x+b_xb_xc_x+b_xc_xc_x+c_xc_xc_x)+\\
&\quad n_z\cdot\tfrac{1}{60}(\quad a_za_za_z+a_za_zb_z+a_za_zc_z+a_zb_zb_z+a_zb_zc_z+\\
&\quad a_zc_zc_z+b_zb_zb_z+b_zb_zc_z+b_zc_zc_z+c_zc_zc_z)\\
I_{zz} &= n_x\cdot\tfrac{1}{60}(\quad a_xa_xa_x+a_xa_xb_x+a_xa_xc_x+a_xb_xb_x+a_xb_xc_x+\\
&\quad a_xc_xc_x+b_xb_xb_x+b_xb_xc_x+b_xc_xc_x+c_xc_xc_x)+\\
&\quad n_y\cdot\tfrac{1}{60}(\quad a_ya_ya_y+a_ya_yb_y+a_ya_yc_y+a_yb_yb_y+a_yb_yc_y+\\
&\quad a_yc_yc_y+b_yb_yb_y+b_yb_yc_y+b_yc_yc_y+c_yc_yc_y)\\
I_{xy} &= n_x\cdot\tfrac{1}{120}(\quad 3a_xa_xa_y+a_xa_xb_y+a_xa_xc_y+2a_xa_yb_x+2a_xa_yc_x+\\
&\quad 2a_xb_xb_y+a_xb_xc_y+a_xb_yc_x+2a_xc_xc_y+a_yb_xb_x+\\
&\quad a_yb_xc_x+a_yc_xc_x+3b_xb_xb_y+b_xb_xc_y+2b_xb_yc_x+\\
&\quad 2b_xc_xc_y+b_yc_xc_x+3c_xc_xc_y)\\
I_{xz} &= n_x\cdot\tfrac{1}{120}(\quad 3a_xa_xa_z+a_xa_xb_z+a_xa_xc_z+2a_xa_zb_x+2a_xa_zc_x+\\
&\quad 2a_xb_xb_z+a_xb_xc_z+a_xb_zc_x+2a_xc_xc_z+a_zb_xb_x+\\
&\quad a_zb_xc_x+a_zc_xc_x+3b_xb_xb_z+b_xb_xc_z+2b_xb_zc_x+\\
&\quad 2b_xc_xc_z+b_zc_xc_x+3c_xc_xc_z)\\
I_{yz} &= n_y\cdot\tfrac{1}{120}(\quad 3a_ya_ya_z+a_ya_yb_z+a_ya_yc_z+2a_ya_zb_y+2a_ya_zc_y+\\
&\quad 2a_yb_yb_z+a_yb_yc_z+a_yb_zc_y+2a_yc_yc_z+a_zb_yb_y+\\
&\quad a_zb_yc_y+a_zc_yc_y+3b_yb_yb_z+b_yb_yc_z+2b_yb_zc_y+\\
&\quad 2b_yc_yc_z+b_zc_yc_y+3c_yc_yc_z)
\end{aligned}$$

Literaturverzeichnis

[1] INTEC GmbH, www.simpack.de, 2

[2] TNO Automotive, Madymo, www.automotive.tno.nl, 2

[3] MSC Software Corporation, 2300 Traverwood Drive, Ann Arbor, Michigan 48105 USA, www.adams.com, 2

[4] Oana Schüszler, Computersimulationen von realen Kraftfahrzeug Fußgänger Unfällen, Dissertation, Universität Tübingen, 1998, 2

[5] Valentin Keppler, Modellbildung und Simulation biomechanischer Menschmodelle mit Interaktion in einer virtuellen Umgebung, Dissertation, Universität Tübingen, 2003, 2, 39, 40, 88

[6] MATLAB, The MathWorks Inc., 3 Apple Hill Drive, Natick MA 01760-2098, USA, www.matlab.com, 9, 99

[7] Mathematica, Wolfram Research, Inc., 100 Trade Center Drive, Champaign, IL 61820-7237, USA, www.wolfram.com, 103

[8] Lawrence F. Shampine / Mark W. Reichelt, The Matlab ODE Suite, SIAM Journal on Scientific Computing, Volume 18 (Januar 1997), Nummer 1, Seiten 1 bis 22, 9

[9] Ernst Hairer / Gerhard Wanner, Solving Ordinary Differential Equations II, Springer Verlag, ISBN 3-540-60452-9, 7, 14, 15, 29

[10] P. N. Brown / George D. Byrne / Alan C. Hindmarsh, VODE: A Variable Coefficient ODE Solver, SIAM Journal on Scientific and Statistical Computing, Volume 10 (1989), Seiten 1038 bis 1051, sowie LLNL Report UCRL-98412, Juni 1988, 7, 14

[11] Scott D. Cohen / Alan C. Hindmarsh, CVODE, A solver for stiff and nonstiff initial value problems, LLNL Report UCRL-MA-118618, October 1994, 7, 14

[12] K.Strehmel / R.Weiner, Numerik gewöhnlicher Differentialgleichungen, Teubner 1995, ISBN 3-519-02097-1, 10

[13] Jørgen Bjørnstrup, Estimation of Human Body Segment Parameters, Historical Background, October 1995 33, 55

[14] Jørgen Bjørnstrup, Estimation of Human Body Segment Parameters, Statistical Analysis of Results from Prior Investigations, June 1996, 33

[15] Chandler / E.F.Clauser / C.E.MCConville, Investigation of inertial properties of the human body, AMRL Technical Report, NASA Wright-Patterson Air Force Base, 74, 1975, 33, 55

[16] C.E.Clauser / J.T.MCConville / J.W.Young, Weight, volume and center of mass of segments of the human body, AMRL Technical Report, NASA Wright Patterson Air Force Base, 69-70, 1969, 33

[17] NASA Reference Publication 1024. The internal properties of the body and it's segments, NASA, 1978, 33

[18] Erik B.Dam / Martin Koch / Martin Lillholm, Quaternions, Interpolation and Animation, 26

[19] David Baraff, Linear-Time Dynamics using Lagrange Multipliers, SIGGRAPH 1996, COMPUTER GRAPHICS Proceedings,

[20] Alexander Sporrer, Grundlagen und Modellierung eines biomechanischen Mehrkörpersystems des Mensch zur Computersimulation von Bewegungsabläufen bei rechtsmedizinischen Fragestellungen, Dissertation, Univ. München, 2000, 39

[21] Eric Larsen / Stefan Gottschalk / Ming C. Lin / Dinesh Manocha, PQP - A Proximity Query Package, Fast Proximity Queries with Swept Sphere Volumes, University of North Carolina, Chapel Hill, USA, Technical report TR99-018, www.cs.unc.edu/~geom/SSV/, 40

[22] Kollisionserkennungssysteme an der University of North Carolina, Chapel Hill, USA, www.cs.unc.edu/~geom/collide/index.shtml,

[23] S.Gottschalk / Ming C.Lin / D.Manocha, OBBTree: A Hierarchical Structure for Rapid Interference Detection, Department of Computer Science, University of North Carolina, Chapel Hill, USA, ACM Siggraph, 1996, 40

[24] Gabriel Zachmann / Jan Klein, Time-Critical Collision Detection Using an Average-Case Approach, VMV 2003, Nov 2003, München, 41

[25] Stephan A.Ehmann / Ming C.Lin, Accurate and Fast Proximity Queries betweeen Polyhedra using Convex Surface Decomposition, Department of Computer Science, University of North Carolina, Chapel Hill, USA, EUROGRAPHICS 2001, Volume 20(2001), Number 3, 40

[26] Stephan A.Ehmann / Ming C.Lin, SWIFT: Accelerated Proximity Queries using Multi-Level Voronoi Marching, Department of Computer Science, Univ. of North Carolina, Chapel Hill, USA, www.cs.unc.edu/~geom/SWIFT++/, 40

[27] Jon Cohen / Ming C.Lin / Dinesh Manocha, I-COLLIDE: An interactive and exact Collision Detection Library for large Environments composed of Convex Polyhedra, Department of Computer Science, University of North Carolina, Chapel Hill, USA, www.cs.unc.edu/~geom/I_COLLIDE/, 40

[28] Thomas C.Hudson / Ming C.Lin / Jonathan Cohen / Stefan Gottschalk / Dinesh Manocha, V-COLLIDE: Accelerated Collision Detection for VRML, Department of Computer Science, University of North Carolina, Chapel Hill, USA, www.cs.unc.edu/~geom/V_COLLIDE/index.html, 41

[29] Young J.Kim / Ming C.Lin / Dinesh Manocha, DEEP: Dual-space Expansion for Estimating Penetration depth between convex polytopes, Department of Computer Science, University of North Carolina, Chapel Hill, USA, www.cs.unc.edu/~geom/DEEP/, 41

[30] Brian Mirtich, V-Clip: Fast and Robust Polyhedral Collision Detection, ACM Transactions on Graphics, Juli 97, www.merl.com/projects/vclip, 41, 43, 45

[31] Elmer G.Gilbert / Daniel W.Johnson / S.Sathiya Keerthi, A Fast procedure for computing the distance between complex objects in three-dimensional space, IEEE Journal of Robotics and Automation, 4(2):193-203, April 1988, 41, 43

[32] Stephen Cameron, Enhancing GJK: Computing minimum penetration distances between convex polyhedra, International Conference on Robotics and Automation, IEEE, April 1997, 41

[33] Ming C.Lin/Dinesh Manocha/John Canny, Fast contact determination in dynamic environments, IEEE Conference on Robotics and Automation, p: 602-609, 1994, 41

[34] Elmar Schömer / Christian Thiel, Efficient collision detection for moving polyhedra, 11th Annual ACM Symposium on Computational Geometry, p: 51-60, 41

[35] Jessica K.Hodgins / Wayne L.Wooten / David C.Brogan / James F.O'Brien, Animating Human Athletics, College of Computing, Georgia Institute of Technology Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, September 1995, www.cc.gatech.edu/gvu/animation/Animation.html, 69

[36] Andrew Witkin / Michael Kass, Spacetime Constraints, Computer Graphics, 22:159-168, 1988, Proceedings Siggraph 1988, 70

[37] Anthony C.Fang / Nancy S.Pollard, Efficient Synthesis of Physically Valid Human Motion, Department of Computer Science, Brown University, Proceedings of Siggraph 2003, Volume 22, Issue 3, pg. 417 - 426, July 2003, 70

[38] Michael Günther, Computersimulationen zur Synthetisierung des muskulär erzeugten menschlichen Gehens unter Verwendung eines biomechanischen Mehrkörpermodells, Dissertation, Universität Tübingen, 1997, 56

[39] I.A. Kapandji, Funktionelle Anatomie der Gelenke, Hippokrates Verlag, 3. Auflage, 2001, ISBN 3-7773-1941-4, 57, 62, 63, 64

[40] Felix E. Zajac, Muscle and tendon: properties, models, scaling, and application to biomechanics and motor control, Critical reviews in biomedical engineering, Vol. 17 (4), p: 359-411, 1989, 58, 59

[41] Karin Gruber / Hanns Ruder / J.Denoth / K.Schneider, A comparative study of impact dynamics: wobbling mass model versus rigid body models, Journal of Biomechanics, 31 (1998) pages 439-444, 72

[42] Dirk Haberkamp, Biomechanische Bewegungsanalyse des Kippaufschwungs am Hochreck von Turnern unterschiedlichen Leistungsniveaus, Examensarbeit 2003, Universität Koblenz, 70, 72, 78

[43] Thomas Göth, Biomechanische Bewegungsanalyse des Kippaufschwungs am Hochreck von Turnern unterschiedlichen Leistungsniveaus, Examensarbeit 2003, Universität Koblenz, 70, 71, 75

[44] SIMI Motion, SIMI Reality Motion Systems GmbH, www.simi.com, 71

[45] J. Versace, A Review of the Severity Index, SAE Paper No. 710881, Proceedings of the Fifteenth Stapp Car Crash Conference, New York, Society of Automotive Engineers, 1971, 86