Interner Bericht 2003/19

Dynamische Simulation von gelenkgekoppelten Starrkörpersystemen mit der Impulstechnik[1]

von
Alfred A. Schmitt
Institut für Betriebs- und Dialogsysteme
Fakultät für Informatik
Universität Karlsruhe
aschmitt@ira.uka.de

Inhaltsübersicht

[1] Gefördert von der DFG im Rahmen des SFB 588 „Humanoide Roboter“

Kurzfassung

Wir stellen hier eine völlig neue Methode zur dynamischen Simulation gelenkgekoppelter Starrkörpersysteme vor. Bei der Methode werden Kräfte konsequent durch Impulse ersetzt. Dadurch gelingt es, die Simulation sehr direkt und ohne das Lösen komplexer Systeme von Differentialgleichungen durchzuführen. Kern des Verfahrens ist ein neuer iterativer Algorithmus zur Bestimmung der Impulse, die dafür sorgen, dass die Gelenkbedingungen nicht verletzt werden. Das Verfahren ist vergleichsweise einfach zu implementieren, es ist sehr schnell und genau. Dies wird durch erste experimentell erhobene Daten und Genauigkeitsabschätzungen belegt. Mechanische Modelle müssen nur minimal vorverarbeitet werden, um sie simulieren zu können. Die impulsbasierte Methode ist außerdem sehr flexibel, wenn neue Gelenktypen eingeführt werden sollen. Auch Stoß und Reibung sind in unkomplizierter Weise integrierbar. Das Verfahren eignet sich besonders für die neue Generation von Virtual Reality Systemen, in denen sich die VR-Welten mechanisch und kinematisch korrekt verhalten sollen. Auch für die dynamische Simulation mechatronischer Systeme ist es sehr geeignet, da selbst sehr komplexe Modelle wie z. B. 6-beinige Laufmaschinen auf einem PC problemlos in Echtzeit simuliert werden können.

Abstract

We present a new method for the dynamic simulation of linked rigid body systems. Forces are systematically substituted by impulses. In this way we suceed to perform the simulation in a very direct way and without solving complex systems of differential equations. The basic principle of the method is a new iterative procedure to calculate impulses which conserve joint constraints. The implementation of the method is comparably simple and it is very fast and accurat. This is documented by a first presentation of experimental data and accuracy estimates. In order to start the simulation, mechanical models need only minimal preprocessing. The impulse-based method is also very flexible if new types of joints have to be introduced. Collision and friction are implemented in a direct manner. The method is especially well suited for the new generation of VR systems with correct dynamic and kinematic behaviour of VR-worlds. It is also well suited for the simulation of mechatronic systems since the simulation of complex models like six-legged walking machines is possible in real time on a PC.

1. Einleitung

In der dynamischen Simulation von Mehrkörpersystemen kann man grob zwischen den klassischen, auf Genauigkeit ausgerichteten Verfahren und den approximativen Methoden unterscheiden. Die letzteren Verfahren sind grobe Annäherungen an die korrekten mechanischen Verhältnisse und arbeiten z. B. mit Penalty-Methoden, um Gelenkbedingungen einzuhalten. Sie sind in der Computergraphik und in der Computeranimation populär, weil sie mit sehr geringem Aufwand implementiert werden können und darüber hinaus schnell sind. Außerdem liefern sie die gewünschten natürlich aussehenden Bewegungsabläufe.

Die klassischen, auf Genauigkeit ausgerichteten Verfahren stellen Systeme von Differentialgleichungen auf, die dann numerisch z. B. mit Runge-Kutta gelöst werden. Dabei kann man noch zwischen Verfahren mit verallgemeinerten bzw. reduzierten Koordinaten und solchen mit natürlichen (kartesischen) Koordinaten unterscheiden. Erstere versuchen, die Zahl der Variablen auf die Zahl der Freiheitsgrade des gesamten mechanischen Systems zu reduzieren, was zu genauen Integrationsergebnissen führt, aber in manchen Fällen extrem viel Vorverarbeitung und Handeingriff erfordern kann, z. B. bei vielen geschlossenen kinematischen Ketten. Außerdem können die einzelnen Gleichungen sehr umfangreich werden.

Demgegenüber sind die Verfahren mit natürlichen (kartesischen) Koordinaten, manchmal auch maximale Koordinaten genannt, darauf angewiesen, zunächst unbekannte Gelenkkräfte in die Differentialgleichungen einzubeziehen. Das gilt insbesondere für die weit verbreiteten Verfahren mit Lagrange-Multiplikatoren. Da bei den Verfahren mit explizit in Erscheinung tretenden Gelenkkräften in jedem Zeitschritt neben der Integration auch ein Gleichungssystem zu lösen ist, hängt die Simulationsgeschwindigkeit sehr davon ab, wie genau das Gleichungssystem gelöst wird. Mit iterativen Methoden können sehr schnell grobe Annäherungen erzielt werden.

Alle mit Differentialgleichungssystemen und ihren numerischen Lösungen arbeitenden dynamischen Simulationsverfahren haben ein Problem, wenn das mechanische Modell oft umgebaut werden muss und wenn ständig Körperkontakte, Kollisionen und Reibungseffekte mitverfolgt werden sollen: Änderungen im mechanischen System haben zur Folge, dass ein neues Gleichungssystem aufzustellen ist. Das führt z. B. zu der Konsequenz, dass eine 6-beinige Laufmaschine formal durch 2^6 verschiedene Differential-Gleichungssysteme beschrieben werden muss: Für jede Bodenkontakt-Konfiguration der 6 Beine ein eigenes System mit jeweils unterschiedlichen geschlossenen kinematischen Ketten. Weitere Schwierigkeiten stellen sich bei komplexeren Stoßvorgängen gelenkgekoppelter Systeme und bei Einbezug von Reibungseffekten ein.

Es ist also durchaus legitim, nach neuen Dynamik-Simulationsverfahren Ausschau zu halten, die das simulierte mechanische Modell während der Simulation problemlos umbauen und an neue Gegebenheiten blitzschnell anpassen können.

In der Technischen Mechanik sind die klassischen Methoden etabliert und durch mächtige Softwaresysteme (z. B. Adams, SDS, SD-Fast, Matlab-Simulink, Alaska, ODE usw.) unterstützt. Aus dieser Richtung besteht wohl weniger der Wunsch nach neuen Verfahren für die dynamische Simulation. Ganz anders sieht das allerdings in der Computergraphik aus. Bisherige Virtual Reality Systeme sind völlig unfähig, komplexere Szenarien mechanisch korrekt zu simulieren. Aus der Sicht vieler Anwendungen ist ein mechanisch korrektes Verhalten der Gegenstände und Mechanismen in den VR-Welten sehr erwünscht. Wir fordern daher, dass zukünftige VR-Systeme nicht nur die standardmäßigen Funktionalitäten,

sondern neue „Realitäten“ wie korrekte dynamische Simulationen und Manipulationen mit Starrkörpern anbieten können. Insbesondere sind wir an der Integration mechatronischer Systeme bis hin zu autonom agierenden Humanoiden Robotern interessiert. Unsere Anforderungen sind z. B. in Finkenzeller et al [2003] und Bender et al [2003] dargelegt und können durch ein Zitat aus der letztgenannten Arbeit umrissen werden:

„Bei der Simulation sind u. a. folgende Funktionalitäten abzudecken:
- Komplexe, dynamische Simulation für starre Körper, Umwelt, Roboterarme, Greifbewegungen und Objektmanipulation durch Roboterhände, Kollisions- und Stoßauflösung,
- Simulierte Sensorik und Aktuatorik, Servo-Antriebe,
- "Software in the Loop", d. h. die gesamte Software, die das Bewegungsverhalten und die Intelligenz des Humanoiden Roboters ausmacht und steuert, muss mit der Simulation zeitsynchron mitlaufen.“

Schwierig ist dabei nicht die VR-Technik, sondern die Mechanik-Simulation. Auch die Sensorik-Simulation und das Thema „Software in the Loop“ werfen bei komplexen mechatronischen Systemen Probleme auf, die wir in dieser Arbeit aber nicht diskutieren werden. Die typischerweise auf der Lösung von Differentialgleichungen beruhenden Verfahren für die dynamische Simulation müssen in erheblichem Umfang erweitert werden, wenn sie in der VR eingesetzt werden sollen. Dies war uns schon seit längerem bewusst und wir haben nach einem eleganten Algorithmus für die dynamische Simulation gesucht, der vor allem mit Blick auf die VR die Nachteile der klassischen Verfahren überwindet. Das in dieser Arbeit vorgestellte Verfahren der impulsbasierten Dynamiksimulation ist das Ergebnis dieser Suche.

2. Impulsbasierte dynamische Simulation von Massenpunktsystemen

Mit der von Schmitt et al [2000] entwickelten Massenpunkttechnik ist es möglich, Mehrkörpersysteme mit offenen und geschlossenen kinematischen Ketten und mit Antriebskräften wie servoangetriebenen Achsen zu simulieren. Die wichtigsten Eigenschaften dieses Verfahrens sind:

- Massenpunkte oder Punktmassen sind die atomaren Bausteine für Starrkörper.

- Einzelne Starrkörper werden durch eine Anzahl Massenpunkte modelliert, die definierte Abstände voneinander beibehalten müssen.

- Kräfte (bzw. Kraft-Impulse) greifen grundsätzlich nur an Massenpunkten an.

- Bei der Bewegungssimulation von der Zeit t bis zur Zeit $t + h$ (ein Zeitschritt) werden die Massenpunkte mit Newtons 2. Gesetz $F(t) = m \cdot \ddot{p}(t)$ bewegt, zunächst ohne Beachtung der Abstandsbedingungen. Hier ist $\ddot{p}(t)$ die Beschleunigung des Massenpunktes mit der Masse m und es gilt $F(t) = m \cdot g$ (Schwerkraft). Wenn die Abstandsbedingungen zum Zeitpunkt t alle erfüllt sind, so sind sie bei dieser Vorgehensweise zum Zeitpunkt $t + h$ mehr oder weniger verletzt. Es werden dann iterativ Impulse bestimmt, die zum Zeitpunkt t angewandt werden, und zwar so, dass mit diesen neuen Bewegungszuständen die Bewegungssimulation auch zum Zeitpunkt $t + h$ alle Abstandsbedingungen erfüllt.

Das Verfahren ersetzt also die kontinuierlichen inneren Kräfte, die dafür sorgen, dass die Massenpunkte in der Bewegung die vorgegebenen Abstandsbedingungen einhalten, durch iterativ bestimmte Impulse.

Mit einem sehr einfachen Beispiel kann das Prinzip demonstriert werden. In Abb. 2.1 ist an einem raumfesten Punkt p_0 mit einem masselosen Seil der Länge 1 ein ruhender Massenpunkt mit den zeitabhängigen Ortskoordinaten $p(t)$ mit Masse m aufgehängt. Wir haben also eine Abstandsbedingung $\left|p(t)-p_0\right|=1$ und für $p(t)$ die Differentialgleichung $m\cdot g = m\cdot \ddot{p}(t)$, also $\ddot{p}(t)=g$.

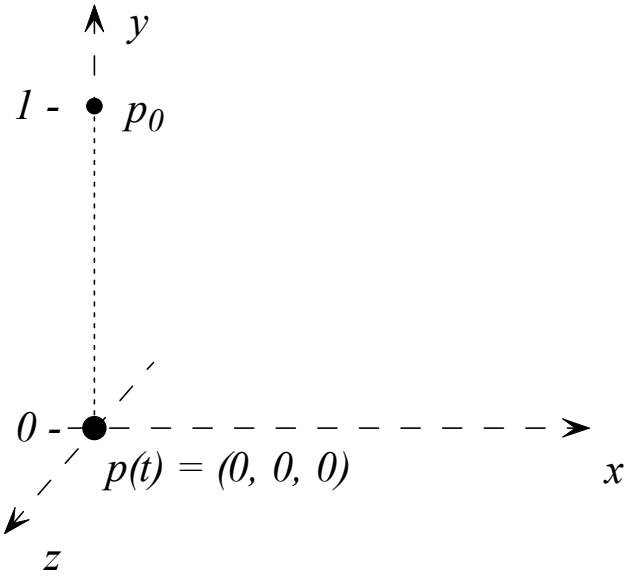

Abbildung 2.1: *Der Massenpunkt $p(t)$ ist mit einem masselosen Faden im Punkt p_0 aufgehängt.*

Der Schwerkraftvektor $g=(0,-9.81,0)$ zeigt in Richtung der negativen y-Achse. Die Anfangsbedingungen sollen sein $p(0)=(0,0,0)$ und $\dot{p}(0)=(0,0,0)$.

Der Zeitschritt von $t=0$ bis $t=h$ wird nun mit der Impulstechnik simuliert. Mit den Werten $p_y, \dot{p}_y$ jeweils der y-Koordinate haben wir:

	Zeit t	$p_y(t)$	$\dot{p}_y(t)$
(1) Konsistenter Zustand:	0	0	0
(2) Look-Ahead:	0+h	$-0.5\cdot g\cdot h^2$	$-g\cdot h$
(3) Korrekturimpuls:	0		$+0.5\cdot g\cdot h$
(4) Zustand	0	0	$+0.5\cdot g\cdot h$
(5) Vorwärtsschritt	0+h	0	$-0.5\cdot g\cdot h$
(6) Geschw.-Korr.	0+h	0	0

Kommentar: Bei (2) und (5) wird der Massenpunkt ohne Beachtung der Abstandsbedingung bewegt, also hier mit dem Fallgesetz. Der Korrekturimpuls (3), zum Zeitpunkt t=0 angewandt, verändert nur die Geschwindigkeit, nicht die Lage des Massenpunktes. Nach dem Schritt (5) liegt der Punkt wieder dort, wo er liegen muss, aber die Geschwindigkeit ist noch von Null verschieden, kann aber durch einen zum Zeitpunkt 0+h angewandten Impuls korrigiert werden. Impulse werden stets als paarweise entgegengerichtete Impulspaare angewandt. In obigem Beispiel greifen die Impulspaare an den Punkten $p(t)$ und p_0 an. Letzterer ist allerdings raumfixiert, kann sich also nicht bewegen und muss nicht weiter beachtet werden.

In diesem sehr einfachen Beispiel führt eine einzige Iteration zum Ziel, der Punkt p liegt wieder bei den korrekten Koordinatenwerten und die Geschwindigkeitskorrektur, die zum Zeitpunkt 0+h ebenfalls mit einem Impuls durchgeführt wird, ergibt den korrekten Bewegungszustand $\dot{p}(h) = (0,0,0)$.

Der Erfolg des Verfahrens beruht darauf, dass die kontinuierlich wirkenden inneren Kräfte durch Impulskräfte ersetzt werden. Impulse ändern nur die Geschwindigkeit und dadurch in der Zukunft auch den Ort des Massenpunktes. Da die Impulse grundsätzlich als entgegen gesetzte Impulspaare angewandt und durch keine Abstände verändert werden, denn die Abstandsbedingungen bleiben erhalten, sind sie energetisch neutral, leisten also keine Arbeit. Von diesen inneren Korrekturimpulsen können auch keinerlei translatorische oder rotatorische Effekte ausgehen. Die Diskussion weiterer technischer Details kann hier abgekürzt werden, weil das in dieser Arbeit vorgestellte Verfahren, welches die Impulstechnik auf gelenkgekoppelte Starrkörpersysteme erweitert, auf der gleichen grundsätzlichen Vorgehensweise beruht.

Das Massenpunktverfahren ist äußerst einfach zu implementieren. Die mechanischen Modelle bestehen in der Implementierung schlicht aus einer Liste von Massenpunkten $L = [\ldots, (p_i(t), \dot{p}_i(t), m_i), \ldots]$ und einer Liste von Abstandsbedingungen $G = [\ldots, (i, j, d_i), \ldots]$ und das Simulationsprogramm ist sehr kurz und übersichtlich. Es gibt bei dieser Vorgehensweise keine Trägheitstensoren, Drehvektoren usw.

Nachteilig ist jedoch, dass Starrkörper aus einer Anzahl von Massenpunkten bestehen müssen, also ein Quader z. B. aus 8 Massenpunkten mit bis zu 7 x 8 = 56 Abstandsbedingungen. Ein kompletter Roboterarm (7 Freiheitsgrade) mit Servo-Achsen benötigte 144 Massenpunkte zu seiner Modellierung (Schelling [2002]).

Erhebliche Verbesserungen sind zu erwarten, wenn es gelingt, die impulsbasierte Gelenkkorrektur auf Starrkörper zu übertragen, denn der oben erwähnte Roboterarm besteht dann nur noch aus 7 Starrkörpern. Dadurch wird also der ganze Korrekturaufwand überflüssig, der beim Massenpunktverfahren zu leisten ist, um die Massenpunkte einzelner Starrkörper auf Sollabstand zu halten.

3. Die dynamische Simulation einzelner, nicht gelenkgekoppelter Starrkörper

Viele der in diesem Abschnitt angesprochenen Formeln und Beziehungen werden seit langer Zeit in den Lehrbüchern der Mechanik und der Technischen Mechanik ausführlich

behandelt. Um auch Außenstehenden alle Details unserer Vorgehensweise offen zu legen, scheint es zweckmäßig, auch allgemein Bekanntes zu rekapitulieren.

Wir betrachten zunächst einen Starrkörper mit k fest im Raum liegenden Massenpunkten $p_i = (x_i, y_i, z_i)$ jeweils mit den Massen m_i (i=1...k). Die Gesamtmasse ist dann $m := \sum_{i=1}^{k} m_i$ und der Schwerpunkt $s := \frac{1}{m} \sum_{i=1}^{k} m_i \cdot p_i$.

Bezogen auf das absolut ruhende Weltkoordinatensystem (Inertialsystem) befindet sich der Körper in Nulllage, wenn $s = (0,0,0)$. Aus den folgenden Bestandteilen wird dann der Trägheitstensor J gebildet:

$$X^2 := \sum m_i \cdot x_i^2, \qquad Y^2 := \sum m_i \cdot y_i^2, \qquad Z^2 := \sum m_i \cdot z_i^2$$

$$XY := -\sum m_i \cdot x_i \cdot y_i, \quad YZ := -\sum m_i \cdot y_i \cdot z_i, \quad XZ := -\sum m_i \cdot z_i \cdot x_i$$

$$J = \begin{pmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & J_{23} \\ J_{31} & J_{32} & J_{33} \end{pmatrix} = \begin{pmatrix} Y^2 + Z^2 & XY & XZ \\ XY & Z^2 + X^2 & YZ \\ XZ & YZ & X^2 + Y^2 \end{pmatrix}$$

Wenn der Körper nicht aus einzelnen Massenpunkten, sondern aus kontinuierlichen Masseverteilungen besteht, so werden die Größen $X^2, Y^2, Z^2, XY, YZ, XZ$ mittels Integration bestimmt. (Man kann mit relativ einfachen Überlegungen nachweisen, dass für jeden Körper mit kontinuierlicher Masseverteilung ein bezüglich *J* äquivalenter Körper mit 6 Massenpunkten existiert. In der Dynamik sind also Starrkörper mit kontinuierlichen Masseverteilungen den aus diskreten Massenpunkten aufgebauten Starrkörpern äquivalent.)

Der Körper, in unserem Fall die Massenpunkte, können so um den Nullpunkt gedreht werden, dass alle Nebendiagonalelemente XY, XZ, YZ zu 0 werden. In diesem Fall, den wir generell anstreben, fallen die $x-, y-$ und $z-$Achse des Weltkoordinatensystems mit den Hauptträgheitsachsen zusammen und wir haben also die Parameter:

- Schwerpunkt $s = (0,0,0)$
- Hauptträgheitsachsen-Matrix $R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = (A_1, A_2, A_3)$
- Trägheitstensor $J = \begin{pmatrix} J_1 & 0 & 0 \\ 0 & J_2 & 0 \\ 0 & 0 & J_3 \end{pmatrix}$

Von jetzt an wollen wir diese Lage des Körpers als Nulllage bezeichnen. Soll der Starrkörper in allgemeine Lage gebracht werden, so ist s zu verändern und die

Hauptachsenvektoren A_1, A_2, A_3 sind in die neue Orientierung zu drehen. J ist der Trägheitstensor in Nulllage und muss ebenfalls transformiert werden, wenn der Körper gegenüber der Nulllage gedreht wird: $\tilde{J} := R \cdot J \cdot R^T$.

Die Spaltenvektoren $A_1, \ldots, A_3$ von R bilden auch nach beliebigen Transformationen im Weltkoordinatensystem ein Orthonormalsystem. Gleichzeitig ist R eine Rotationsmatrix, mit deren Hilfe Punkte des Körpers aus allgemeiner Lage in die korrespondierende Nulllage und umgekehrt transformiert werden können, und zwar wie folgt:

Ein Punkt p in Nulllage wird zu $p' := s + R \cdot p$ in allgemeiner Lage, wobei s der Schwerpunkt des Körpers (in Weltkoordinaten) ist, und p' wird zurücktransformiert in die Nulllage mittels $p = R^{-1}\left(p' - s\right)$. Da R orthonormal ist, gilt $R^{-1} = R^T$, so dass also das Invertieren solcher Matrizen einfach durch Transponieren realisiert werden kann.

Der Bewegungszustand eines Starrkörpers in allgemeiner Lage zum Zeitpunkt t ist gegeben durch:

(1) $s(t)$, $\dot{s}(t)$ (Schwerpunkt, Schwerpunktgeschwindigkeit)

(2) $R(t)$ (Hauptträgheitsachsen-Matrix oder Rotationsmatrix)

(3) $\omega(t)$ (Drehvektor)

Der Vektor ω (in Weltkoordinaten) zeigt in Richtung der Drehachse, die stets durch den Schwerpunkt verlaufen soll. Seine Länge $|\omega|$ ist die Rotationsgeschwindigkeit, wobei der Winkel im Bogenmaß (Radian) gemessen wird. Mit Hilfe von $\omega(t)$ und $\dot{s}(t)$ lassen sich Geschwindigkeiten von Punkten des Starrkörpers bestimmen:

(3.1) $$\dot{p}_i(t) = \dot{s}(t) + \omega(t) \times \left(p_i(t) - s(t)\right)$$

Da $p_i(t)$ in Weltkoordinaten angegeben ist, muss mit $p_i(t) - s(t)$ erst ein Vektor relativ zum Schwerpunkt berechnet werden, bevor dieser mit $\omega(t)$ multipliziert werden kann.

Starrkörperbewegungen auf ballistischen Bahnen

Wir haben nun alle Vorbereitungen getroffen, um uns der ersten Simulationsaufgabe zuzuwenden. Wir wollen bestimmen, wie sich ein Starrkörper aus einem bestimmten Bewegungszustand heraus im Raum bewegt, wenn außer der Schwerkraft $g \cdot m$ keine äußeren Kräfte einwirken. Wir bezeichnen dies als Bewegung von Starrkörpern auf ballistischen Bahnen.

Das Problem lösen wir mit den Newton-Euler-Formeln: Die Bewegung eines Starrkörpers auf ballistischen Bahnen ist zusammengesetzt aus der Bewegung des Schwerpunktes gemäß

(3.2) $F(t) = m \cdot g = m \cdot \ddot{s}(t)$ (2. Gesetz von Newton)

und der davon völlig unabhängigen rotatorischen Bewegung, die durch

(3.3) $J \cdot \dot{\omega} + \left(\omega \times \left(J \cdot \omega\right)\right) = 0$ (Euler)

beschrieben wird. Allerdings wird (3.2) grundsätzlich in Weltkoordinaten, (3.3) jedoch im körpereigenen Koordinatensystem – also quasi in Nulllage – gelöst.

Die Lösung von (3.2) ist trivial. Wenn wir einen Zeitschritt von $t = 0$ bis $t = h$ (h= Zeitschrittweite) betrachten und vom Bewegungszustand zum Zeitpunkt 0 ausgehend den Bewegungszustand zum Zeitpunkt h berechnen wollen, so ergeben sich die Formeln

$$\begin{aligned} &\ddot{s}(t) = g, \quad \dot{s}(h) = \dot{s}(0) + \int_0^h \ddot{s}(t)\,dt = \dot{s}(0) + g \cdot h, \\ &s(h) = s(0) + \int_0^h gt\,dt = s(0) + 0.5gh^2 \end{aligned} \tag{3.4}$$

Die Lösung von (3.3) ist in geschlossener Form nur in Spezialfällen bekannt. Wir lösen diese Differentialgleichung durch Taylor-Entwicklung an der Stelle $t = 0$:

Als erstes muss das $\omega(0)$, das wir im Bewegungszustand in Weltkoordinaten gespeichert haben, in das körpereigene Koordinatensystem transformiert werden mit

$$\underline{\omega}(0) := R^T(0) \cdot \omega(0).$$

Die in das körpereigene Koordinatensystem transformierten Größen kennzeichnen wir im Zweifelsfall durch Unterstreichen. Wir kennen $\underline{\omega}(0)$, jetzt im körpereigenen Koordinatensystem, also auch

$$\dot{\underline{\omega}}(0) = -J^{-1}\left(\underline{\omega}(0) \times \left(J \cdot \underline{\omega}(0)\right)\right)$$

und wegen $\dot{\underline{\omega}}(t) = -J^{-1}\left(\underline{\omega}(t) \times \left(J \cdot \underline{\omega}(t)\right)\right)$ durch fortgesetztes Differenzieren nach *t* auch

$$\ddot{\underline{\omega}}(0) = -J^{-1}\left(\left(\dot{\underline{\omega}}(0) \times J \cdot \underline{\omega}(0)\right) + \left(\underline{\omega}(0) \times J \cdot \dot{\underline{\omega}}(0)\right)\right)$$

usw. usf.

Mit der Notation $\omega 0 := \underline{\omega}(0), \omega 1 := \dot{\underline{\omega}}(0), \omega 2 := \ddot{\underline{\omega}}(0)$ usw. erhalten wir durch fortgesetztes Differenzieren die Ausdrücke

$$\begin{aligned} \omega 1 &= -J^{-1}\left(\omega 0 \times J\omega 0\right) \\ \omega 2 &= -J^{-1}\left(\omega 1 \times J\omega 0 + \omega 0 \times J\omega 1\right) \\ \omega 3 &= -J^{-1}\left(\omega 2 \times J\omega 0 + 2\omega 1 \times J\omega 1 + \omega 0 \times J\omega 2\right) \\ \omega 4 &= -J^{-1}\left(\omega 3 \times J\omega 0 + 3\omega 2 \times J\omega 1 + 3\omega 1 \times J\omega 2 + \omega 0 \times J\omega 3\right) \end{aligned}$$

Die Taylorentwicklung von $\underline{\omega}(t)$ an der Stelle $t = 0$ ist also

$$\underline{\omega}(t) = \omega 0 + \omega 1 \cdot t + \frac{1}{2!}\omega 2 \cdot t^2 + \frac{1}{3!}\omega 3 \cdot t^3 + \frac{1}{4!}\omega 4 \cdot t^4 + O\left(t^5\right)$$

Wir können also $\omega(t)$ mit beliebig vorgebbarer Genauigkeit bestimmen, wenn wir den oben bei $\omega 4$ abgebrochenen Prozess fortsetzen, was problemlos möglich ist. Wir erhalten also durch Einsetzen von $t = h$:

$$\underline{\omega}(h) = \omega 0 + \omega 1 \cdot h + \frac{1}{2!}\omega 2 \cdot h^2 + \frac{1}{3!}\omega 3 \cdot h^3 + \frac{1}{4!}\omega 4 \cdot h^4 + O\left(h^5\right)$$

Bei Schrittweiten von $h \le 0.01$ genügt die Entwicklung bis zum Glied der Ordnung 3, weil die restlichen Glieder bereits sehr klein sind und sich nach unseren Erfahrungen nur noch minimal auf die Simulationsergebnisse auswirken.

Ein kleines offenes Problem ist noch zu diskutieren: Wie berechnen wir J^{-1}, wenn nicht alle Elemente der Hauptdiagonale von $J \neq 0$ sind? Die Regel ist einfach: Wenn $J_i \neq 0$, so ist $1/J_i$ das entsprechende Diagonal-Element von J^{-1} . Ist $J_i = 0$, so setzen wir $J_i^{-1} = 0$. Diese Vorgehensweise führt zumindest bei der Eulerschen Differentialgleichung zu korrekten Lösungen. Denn in Gleichung (3.3) bedeutet $J_i = 0$, dass die Koordinaten-Komponente $\dot{\omega}_i$ durch den Faktor $J_i \cdot \dot{\omega}_i$ aus dem Differentialgleichungssystem ausgeblendet wird, also auch vollständig entfallen kann.

Die oben mit Taylor-Entwicklung berechnete Größe $\underline{\omega}(h)$ wird benötigt, wenn der Bewegungszustand von $t = 0$ zu $t = h$ fortgeschaltet werden soll – allerdings darf dabei die Rücktransformation in das Weltkoordinatensystem nicht vergessen werden. Rücktransformiert wird mit $\omega(h) := R(0) \cdot \underline{\omega}(h)$.

Als nächstes müssen wir untersuchen, wie sich ein Vektor v im Weltkoordinatensystem beim Übergang von $t = 0$ nach $t = h$ verändert, also von $v(0)$ nach $v(h)$. Auch hier muss zunächst in das körpereigene Koordinatensystem transformiert werden: $\underline{v}(0) = R^T(0) \cdot v(0)$. Es gilt nun $\dot{\underline{v}}(0) = \omega 0 \times \underline{v}(0)$ und generell $\dot{\underline{v}}(t) = \underline{\omega}(t) \times \underline{v}(t)$, also ergibt sich mit der Notation $v0 := \underline{v}(0)$, $v1 := \dot{\underline{v}}(0)$, $v2 := \ddot{\underline{v}}(0)$ usw. bei fortgesetztem Differenzieren die Entwicklung

$$v1 = \omega 0 \times v0$$

$$v2 = \omega 1 \times v0 + \omega 0 \times v1$$

$$v3 = \omega 2 \times v0 + 2 \cdot \omega 1 \times v1 + \omega 0 \times v2$$

$$v4 = \omega 3 \times v0 + 3 \cdot \omega 2 \times v1 + 3 \cdot \omega 1 \times v2 + \omega 0 \times v3$$

usw. usf. Die $\omega 0, \omega 1, \omega 2$ usw. haben wir oben bereits berechnet. Auch hier können wir nun mit der Taylor-Entwicklung die Lage von $\underline{v}(t)$ zum Zeitpunkt $t = h$, also $\underline{v}(h)$ bestimmen:

$$\underline{v}(h) = v0 + v1 \cdot h + \frac{1}{2!} v2 \cdot h^2 + \frac{1}{3!} v3 \cdot h^3 + \ldots\ldots$$

Wenn ω beim Term $\omega 4$ abgebrochen wird, so erhalten wir hier eine Entwicklung bis $v5$, also einen Restfehler von $O(h^6)$.

Wir können nun drei Funktionen definieren:

(3.5) $$Integrate0\left(k, \underline{v}(t), h\right) := \underline{v}(t+h)$$

Weil wir später zwischen verschiedenen Körpern unterscheiden müssen, führen wir gleich hier den Index k des betreffenden Körpers ein. $Integrate0$ bildet den Vektor $\underline{v}(t)$ im körpereigenen Koordinatensystem des Körpers mit Index k ab in $\underline{v}(t+h)$, ebenfalls noch im körpereigenen System. Die zweite Funktion leistet die gleiche Abbildung, aber für Vektoren im Weltsystem:

(3.6) $$Integratew\left(k, v(t), h\right) := v(t+h)$$

Mit der Rotationsmatrix $R_k(t)$ des Körpers k muss also gelten:

$$Integratew\left(k, v(t), h\right) := R_k(t) \cdot Integrate0\left(k, R_k^T(t) \cdot v(t), h\right)$$

Wir können eine weitere sehr nützliche Funktion *LookAhead* einführen, die Punkte $p(t)$ des Starrkörpers k in einem h-Zeitschritt (auf ballistischen Bahnen) vorwärts bewegt:

(3.7) $$LookAhead\left(k, p(t), h\right) := p(t+h)$$

Offensichtlich können wir sie wie folgt realisieren:

$$LookAhead\left(k, p(t), h\right) := s_k(t+h) + Integratew\left(k, p(t) - s_k(t), h\right)$$

Hierbei ist s_k der Schwerpunkt von Körper k, der sich wie in den Formeln (3.4) mitbewegt.

Wie man später sehen wird, ändert sich bei der iterativen Gelenkkorrektur das ω_0 der beteiligten Körper ständig, so dass es nur ganz selten möglich ist, auf die einmal berechneten Werte $\omega 0, \omega 1, \omega 2 \ldots$, zurückzugreifen.

In der gelenkgekoppelten Simulation ist an jedem Starrkörper mit Index k eine Menge von Punkten $p_{k1}, p_{k2}, \ldots$ fixiert für Gelenke und evtl. als Referenzpunkte für die graphische Darstellung. Diese müssen bei der Simulation mitbewegt werden. Es gibt auch die andere Vorgehensweise, diese Punkte in Nulllage zu speichern, dann sind sie unveränderlich und müssen in die Weltkoordinaten transformiert werden, wenn sie dort benötigt werden. In dieser Einführung in die impulsbasierte dynamische Simulation entscheiden wir uns für die erste Möglichkeit, also die Mitführung der Punkte in Weltlage. Wir könnten diese Punkte ganz einfach mit der oben eingeführten Funktion

$$p_{ki}(t+h) = LookAhead(k, p_{ki}(t), h)$$

vorwärts bewegen, was aber zu aufwändig wäre. Es gibt nämlich für jede beliebige Bewegung eines Starrkörpers von Lage A nach Lage B stets eine 3 x 3-Rotationsmatrix M_{rot}, die kombiniert mit der veränderten Schwerpunktlage eine einfache Umrechnung der Punkte von Lage A nach Lage B gestattet:

$$p_B = s_B + M_{rot}(p_A - s_A)$$

Diese Matrix M_{rot} wollen wir nun bestimmen für den Übergang vom Zeitpunkt $t = 0$ zum Zeitpunkt $t = h$.

Dazu genügt es, die drei Einheitsvektoren $e_1 = (1,0,0)$, $e_2 = (0,1,0)$ und $e_3 = (0,0,1)$ in Weltkoordinaten der Vorwärtssimulation zu unterziehen gemäß

$$v_x = Integratew(k, e_1, h)$$
$$v_y = Integratew(k, e_2, h)$$
$$v_z = Integratew(k, e_3, h)$$

Mit den v_x, v_y, v_z als Spaltenvektoren erhalten wir die gewünschte Rotationsmatrix

(3.8) $$M_{rot} = (v_x, v_y, v_z)$$

Denn ein beliebiger Vektor $v = (a,b,c)$ in Weltkoordinaten kann dargestellt werden als $v(0) = a \cdot e_1 + b \cdot e_2 + c \cdot e_3$, folglich transformiert als $v(h) = a \cdot v_x + b \cdot v_y + c \cdot v_z$ und damit $v(h) = M_{rot} \cdot v(0)$.

Die Rotationsmatrix M_{rot} kann nicht nur benutzt werden, um Punkte des Körpers k von Lage A nach Lage B zu transformieren, sondern auch um $R_k(0)$ auf die Zeit h umzurechnen. Dabei müssen nur die Spaltenvektoren von $R_k(0)$, die ja die Nulllageachsen des Körpers in Weltkoordinaten darstellen, transformiert werden:

(3.9) $$R_k(h) := M_{rot} \cdot R_k(0)$$

Nun haben wir schließlich alles zusammen, um den Bewegungszustand eines Körpers mit Index k, also $s_k(0), \dot{s}_k(0), \omega_k(0), R_k(0)$, auf die Zeit $t = h$ vorwärts zu rechnen, also auf $s_k(h), \dot{s}_k(h), \omega_k(h), R_k(h)$.

Wenn man M_{rot} so wie oben beschrieben berechnet, funktioniert das zwar einige Iterationsschritte ganz gut, aber irgendwann entgleist die Rechnung, weil sich in R_k kleine Abweichungen von der Orthonormalität einschleichen und diese verstärken sich explosions-

artig, wenn sie einmal vorhanden sind. Abhilfe besteht darin, M_{rot} nachzubearbeiten und strikt zu orthonormalisieren. Dafür haben wir ein sehr elegantes Verfahren gefunden:

Iterationsschritt zur Orthonormalisierung

(1) $v'_x := v_x / |v_x|, \quad v'_y := v_y / |v_y|, \quad v'_z := v_z / |v_z|$ (Normierung)

(2) $v_x = 0.5 * \left(v'_x + v'_y \times v'_z \right)$

$v_y := 0.5 * \left(v'_y + v'_z \times v'_x \right)$

$v_z = 0.5 * \left(v'_z + v'_x \times v'_y \right)$

Einmalige, nur in seltenen Fällen zweimalige Iteration führt zu einer (im Rahmen der Zahlengenauigkeit von ca. 13 Dezimalstellen) praktisch perfekten Orthonormalisierung. Man kann beweisen, dass der Orthonormalisierungsfehler durch einen Iterationsschritt quadriert wird, was bedeutet, dass z. B. bei einer Genauigkeit von 7 Dezimalstellen vor der Iteration nach der Iteration etwa 14 Dezimalstellen genau sind. Ein weiterer Vorteil des obigen Verfahrens besteht darin, dass es keine Bevorzugung einer Achse gibt.

Abschließend für diesen Abschnitt kann festgestellt werden, dass wir die Bewegung von Starrkörpern auf ballistischen Bahnen mit beliebig hoher Genauigkeit simulieren können, weil wir keine Integration mit festliegender Restfehlerordnung benutzen, sondern die Taylor-Entwicklungen, wenn dies erforderlich sein sollte, dynamisch zu sehr hohen Ordnungen fortsetzen könnten, was dann allerdings Rechenzeit kostet. Dynamisch bedeutet hier, dass die Ordnung der Taylorpolynome als Parameter vorgebbar ist, ohne das Programm zu verändern.

4. Kraft- und Impulseinwirkungen auf Starrkörper

Wenn in dem Zeitintervall von 0 bis h eine konstante Kraft F auf einen Starrkörper einwirkt, und zwar an einem Punkt $p(t) = s(t) + v(t)$, $v(t)$ Relativvektor zum Schwerpunkt, so ergeben sich die Differentialgleichungen der Bewegung zu:

(4.1) $F + m \cdot g = m \cdot \ddot{s}(t)$

(4.2) $\tilde{J}(t) \cdot \dot{\omega}(t) + \left(\omega(t) \times (\tilde{J}(t) \cdot \omega(t)) \right) = v(t) \times F = Drehmoment$

Gleichung (4.2) ist jetzt in Weltkoordinaten gefasst, dabei muss J in Weltlage transformiert werden, was mit $\tilde{J}(t) := R(t) \cdot J \cdot R(t)^T$ erfolgt. Wenn man beide Gleichungen über die Zeit von 0 bis h integriert, h gegen 0 gehen lässt und dabei F durch F/h ersetzt, wobei F/h weiterhin die Dimension einer Kraft hat, so erhalten wir im einzelnen folgende Terme:

$$\lim_{h \to 0} \int_0^h (F/h)\, dt = I \qquad (I = \text{ Impuls})$$

$$\lim_{h\to 0}\int_0^h m\ddot{s}(t)\,dt = \lim_{h\to 0} m\left(\dot{s}(h)-\dot{s}(0)\right) = m\cdot\Delta\dot{s}$$

$$\lim_{h\to 0}\int_0^h m\cdot g\,dt \;= 0$$

$$\lim_{h\to 0}\int_0^h \tilde{J}(t)\cdot\dot{\omega}(t)\,dt = \tilde{J}(0)\cdot\lim_{h\to 0}\left(\omega(h)-\omega(0)\right) = \tilde{J}(0)\cdot\Delta\omega$$

$$\lim_{h\to 0}\int_0^h \left(\omega(t)\times\tilde{J}(t)\cdot\omega(t)\right)dt = 0$$

$$\lim_{h\to 0}\int_0^h v(t)\times\left(F/h\right)dt = v(0)\times I \qquad \text{(= Drehimpuls)}$$

Bei der Bildung der Grenzwerte ist zu beachten, dass alle nicht nach der Zeit abgeleiteten Größen beschränkt sind. Daher ergibt sich bei der dritten und fünften Zeile das Ergebnis 0.

In Impulsform lauten die Gleichungen (4.1) und (4.2) also

(4.3) $$I = m\cdot\Delta\dot{s}$$

(4.4) $$\tilde{J}(0)\cdot\Delta\omega = v(0)\times I$$

Der Impuls I wirkt in einem infinitesimal kurzen Zeitraum der Dauer ε mit der Kraft F/ε auf den Körper ein und verändert seinen Bewegungszustand gemäß

(4.5) $$\Delta\dot{s} = \frac{1}{m}\cdot I$$

(4.6) $$\Delta\omega = \tilde{J}(0)^{-1}\left(v(0)\times I\right)$$

Dabei ist also (4.5) eine sprunghafte Veränderung der Schwerpunktgeschwindigkeit gemäß $\dot{s}(0) := \dot{s}(0) + \Delta\dot{s}$ und (4.6) eine ebensolche Veränderung des Drehvektors

$$\omega(0) := \omega(0) + \tilde{J}^{-1}(0)\left(v(0)\times I\right).$$

$v(0)$ ist der Vektor, durch den der Angriffspunkt $p(0) = s(0) + v(0)$ – alles in Weltkoordinaten – definiert ist. Schließlich können wir noch feststellen, wie sich durch den Impuls I die Geschwindigkeit des Punktes $p(0)$ verändert. Vor der Impulseinwirkung haben wir $\dot{p}(0) = \dot{s}(0) + \omega(0)\times(v(0))$, und danach

(4.7) $$\dot{p}(0) = \dot{s}(0) + \Delta\dot{s} + (\omega(0)+\Delta\omega)\times v(0) =$$

$$= \dot{s}(0) + \frac{1}{m}\cdot I + (\omega(0) + \tilde{J}(0)^{-1}\left(v(0)\times I\right))\times v(0)$$

Bei unserer impulsbasierten Simulation werden wir bei der iterativen Korrektur von Gelenkbedingungen ganze Salven von Korrekturimpulsen auf die einzelnen Starrkörper

einwirken lassen. Die Zeit bleibt dabei stehen. Da die Größen $\Delta\dot{s}$ und $\Delta\omega$ offensichtlich rein additiv auf den Bewegungszustand einwirken, ist es völlig gleichgültig, in welcher Reihenfolge diese Impulswirkungen verbucht werden:

Additions-Subtraktions-Prinzip:

Solange die Zeit still steht, kann jede einmal stattgefundene Impulseinwirkung durch entsprechende Gegenimpulse wieder rückgängig gemacht werden.

Dieses Additions-Subtraktions-Prinzip wird z. B. bei der Simulation von Stoß- und Reibungsvorgängen eine wesentliche Rolle spielen. Es ist letztendlich wohl auch dafür verantwortlich, dass unser später beschriebenes iteratives Korrekturverfahren so problemlos funktioniert, denn welche Fehlerimpulse auch immer bei der Iteration auf einen Körper einwirken, sie können später wieder rückgängig gemacht werden.

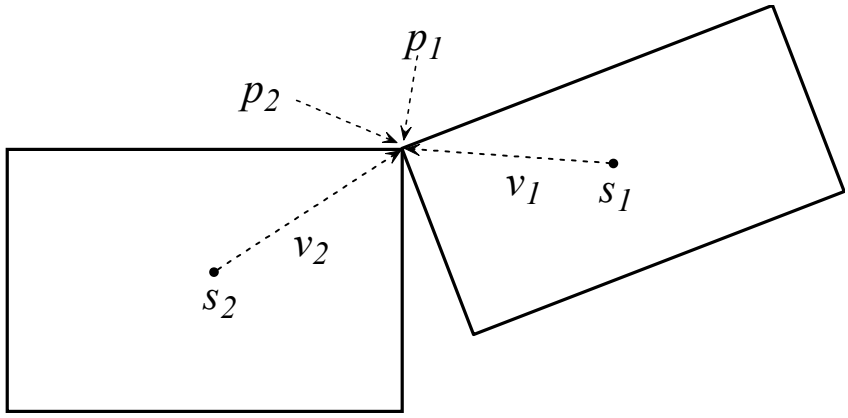

Abbildung 4.1: *Geschwindigkeitskorrektur in einem Kontaktpunktpaar*

Wir wollen nun das Problem der Geschwindigkeitskorrektur mit Impulsen lösen, man vergleiche Abb. 4.1. Zum Zeitpunkt $t = 0$ möge gelten

(a) $s_1(0) + v_1(0) = p_1(0) = p_2(0) = s_2(0) + v_2(0)$

(b) $\dot{p}_1(0) \neq \dot{p}_2(0)$

Zunächst sind also nur die Kontaktpunkte $p_1(0)$ und $p_2(0)$ der Körper 1 und 2 gleich, nicht aber ihre Geschwindigkeiten $\dot{p}_1(0)$ und $\dot{p}_2(0)$, und damit verschwindet auch nicht ihre Relativgeschwindigkeit $\Delta\dot{p} = \dot{p}_2(0) - \dot{p}_1(0)$.

Wir wollen nun durch Anwendung eines sich kompensierenden Impulspaares erreichen, dass auch ihre Geschwindigkeiten übereinstimmen, also $\dot{p}_1(0) = \dot{p}_2(0)$, und zwar ohne dass sich Ihre Lage ändert. Denn wenn sich z. B. an den Kontaktpunkten eine Gelenkverbindung zwischen den beiden Körpern befindet, so muss nicht nur der Ort der zusammengehörenden Gelenkpunkte gleich sein, sondern auch ihre Geschwindigkeit. Wegen der Formel (4.7) muss also für einen bisher noch unbekannten Impuls I gelten:

$$\dot{s}_1 + \frac{1}{m_1}\cdot I + \left(\omega_1 + \tilde{J}_1^{-1}\left(v_1 \times I\right)\right)\times v_1 \quad \square$$

(4.8)

$$\dot{s}_2 + \frac{1}{m_2}\cdot\left(-I\right) + \left(\omega_2 + \tilde{J}_2^{-1}\left(v_2 \times \left(-I\right)\right)\right)\times v_2$$

Wegen $\dot{p}_i = \dot{s}_i + \omega_i \times v_i$ für $i = 1, 2$ können wir umformen:

$$\left(\frac{1}{m_1} + \frac{1}{m_2}\right)\cdot I + \tilde{J}_1^{-1}\left(v_1 \times I\right)\times v_1 + \tilde{J}_2^{-1}\left(v_2 \times I\right)\times v_2 \quad =$$

(4.9)

$$\dot{s}_2 + \omega_2 \times v_2 - \dot{s}_1 - \omega_1 \times v_1 = \dot{p}_2\left(0\right) - \dot{p}_1\left(0\right) = \Delta\dot{p}$$

Hier sind $\dot{p}_1\left(0\right)$ und $\dot{p}_2\left(0\right)$ noch die Werte vor Anwendung von I .

Man beachte, dass in der Gleichung (4.8) der noch unbekannte Impuls I an Körper 1, also im Punkt p_1 mit positivem und an Körper 2 im Punkt p_2 mit negativem Vorzeichen angreift. Solche in einem Punkt oder auf einer geraden Linie in Richtung des I angreifenden Impulspaare nennen wir komplementär. Sie haben genau wie bei komplementären Kräften auf das MKS weder einen global translatorischen noch einen global rotatorischen Einfluss. Sie wirken also exakt so wie die an Gelenken auftretenden inneren Kräfte. Daher verändern solche Impulspaare auch nicht die Gesamtenergie, was allerdings voraussetzt, dass die Gelenkpunkte sich während der Simulation nicht voneinander entfernen dürfen.

Bevor wir die Gleichung (4.9) weiter bearbeiten, führen wir eine spezielle Matrix-Schreibweise für Vektoren $a = \left(a_1, a_2, a_3\right)$ ein:

$$a^* := \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix}$$

Es gilt nun $a \times b = a^* \cdot b$, d. h. das Kreuzprodukt kann jetzt durch eine Matrix-mal-Vektor-Operation ersetzt werden. Nun sind folgende Umformungen von (4.9) möglich:

$$\left(\tilde{J}_1^{-1}\cdot\left(v_1 \times I\right)\right)\times v_1 = -v_1 \times \left(\tilde{J}_1^{-1}\cdot\left(v_1 \times I\right)\right) =$$

$$-v_1^*\cdot\left(\tilde{J}_1^{-1}\cdot\left(v_1^*\cdot I\right)\right) = \left(-v_1^*\cdot\tilde{J}_1^{-1}\cdot v_1^*\right)\cdot I$$

Auch der Term $\frac{1}{m_1}\cdot I$ kann mit E als Einheitsmatrix in die Gestalt $\left(\frac{1}{m_1}\cdot E\right)\cdot I$ gebracht werden, so dass eine 3×3-Matrix als Faktor vor dem I auftritt. Also können wir eine Matrix

$$M\left(i, v_j, v_k\right) := \frac{1}{m_i}\cdot E + \left(-v_k^*\cdot\tilde{J}_i^{-1}\cdot v_j^*\right)$$

definieren, so dass die Gleichung (4.9) die folgende Gestalt erhält:

$$\left(M\left(1,v_1,v_1\right)+M\left(2,v_2,v_2\right)\right)\cdot I=\Delta\dot{p}$$

Die Unbekannte I erhalten wir also als Lösung eines linearen Gleichungssystems mit 3 Unbekannten. Solange die Zeit still steht, ändert sich

$$M:=M\left(1,v_1,v_1\right)+M\left(2,v_2,v_2\right)$$

nicht, da in ihr keine Bestandteile des Bewegungszustandes enthalten sind. Wir können also M^{-1} bestimmen, Singularitäten wurden bei M nie beobachtet, und bestimmen für beliebig vorgebbares $\Delta\dot{p}$ mit

$$I:=M^{-1}\cdot\Delta\dot{p}$$

Richtung und Stärke des komplementären Impulses, wenn an den beteiligten Kontaktpunkten die Differenzgeschwindigkeit $\Delta\dot{p}=\dot{p}_2\left(0\right)-\dot{p}_1\left(0\right)$ beseitigt werden soll. Mirtich [1996] nennt die Matrix M Kollisionsmatrix und zeigt, dass solche Matrizen nichtsingulär, symmetrisch und positiv definit sind.

Selbstverständlich müssen wir mit

$$\dot{s}_1:=\dot{s}_1+\frac{1}{m_1}\cdot I$$

$$\dot{s}_2:=\dot{s}_2+\frac{1}{m_2}\cdot\left(-I\right)$$

$$\omega_1:=\omega_1+\tilde{J}_1^{-1}\left(v_1\times I\right)$$

$$\omega_2:=\omega_2+\tilde{J}_2^{-1}\left(v_2\times\left(-I\right)\right)$$

den Bewegungszustand der beteiligten Körper aktualisieren, um die Geschwindigkeitsänderung Wirklichkeit werden zu lassen.

Man erkennt bei diesen Überlegungen einen großen Vorteil der impulsbasierten Methode, der sich später insbesondere auch bei Kollisionsauflösungen zeigen wird: Man kann die Zeit still stehen lassen und dennoch den Bewegungszustand des MKS im gewünschten Sinne manipulieren. Es wird sich zeigen, dass damit insbesondere Stöße, Kollisionen und Reibungsvorgänge auf elegante Weise beherrschbar werden.

5. Dynamische Simulation von gelenkgekoppelten Starrkörpersystemen

Wir haben nun alle Vorbereitungen getroffen, um unser neues Dynamik-Simulationsverfahren genau beschreiben zu können. Doch zunächst benötigen wir eine Spezifikation der statischen und dynamischen Parameter, die erforderlich sind, um ein gelenkgekoppeltes Mehrkörpersystem (MKS) vollständig zu beschreiben.

Definition:

In einem Mehrkörpersystem (MKS) von n *starren Körpern ist der Körper mit Index* $k \in \{1,\ldots,n\}$ *zum Zeitpunkt* t *definiert durch:*

(1) seine Gesamtmasse m_k *,*

(2) seinen Schwerpunkt $s_k(t) \in R^3$

(3) die Geschwindigkeit des Schwerpunktes $\dot{s}_k(t) \in R^3$

(4) den Drehvektor $\omega_k(t) \in R^3$

(5) seinen (konstanten) Trägheitstensor $J_k \in R^{3x3}$ *(im körpereigenen Koordinatensystem und in Diagonalform)*

(6) seine Rotationsmatrix $R_k(t) \in R^{3x3}$ *, die beschreibt, wo die Achsen des im Körper fixierten (also körpereigenen) Koordinatensystems im Weltkoordinatensystem liegen. (Es sind die drei Spaltenvektoren von* $R_k(t)$ *).*

(7) eine Liste von Referenzpunkten $p_{k1}(t), p_{k2}(t), \ldots \in R^3$*, welche mit dem Körper fix verbunden sind,*

Man beachte, dass in der impulsbasierten Simulation Beschleunigungen nicht mehr explizit in Erscheinung treten, weil deren Rolle von den Impulsen übernommen wird.

Bei Mehrkörpersystemen in der Schwerkraft müssen raumfeste Bodenkörper oder raumfeste Starrkörper vorgesehen werden. Für solche Körper sind natürlich die bewegungsrelevanten Daten irrelevant, weil sie grundsätzlich unbeweglich sind. Sie liefern in der Regel nur raumfeste Referenzpunkte und Hindernisse wie Boden- oder Wandpolygone.

Zusätzlich muss nun spezifiziert werden, wie die einzelnen Körper durch Gelenke miteinander verbunden sind. Für diese erste Beschreibung des Verfahrens beschränken wir uns auf Punkt-zu-Punkt-Gelenke („spherical joints“ oder „ball joints“) mit drei Freiheitsgraden. Später werden wir weitere Gelenktypen diskutieren.

Definition:

Eine Verbindung des Typs „Punkt-zu-Punkt-Gelenk“ wird beschrieben durch ein 5-Tupel

$$G = (0, k_1, i_1, k_2, i_2)$$

bestehend aus folgenden Bestandteilen:

0 *= Typ-Nr. des Gelenks,* 0 *= Punkt-zu-Punkt-Gelenk*
k_1 *= Index Körper 1*
i_1 *= Index des Referenzpunktes von Körper 1*
k_2 *= Index Körper 2*

i_2 = *Index des Referenzpunktes von Körper 2*

Durch ein solches Gelenk wird festgelegt, dass sich der Referenzpunkt $p_{k_1 i_1}$ *von Körper 1 stets am gleichen Ort befinden muss wie der Punkt* $p_{k_2 i_2}$ *von Körper 2.*

Für die Simulation bedeutet dies, dass immer $p_{k_1 i_1}(t) = p_{k_2 i_2}(t)$ angestrebt werden muss, d. h. die beteiligten Körper sind an diesen Referenzpunkten miteinander verkettet. Wir werden später Erweiterungen der Gelenktypen diskutieren, daher wurde oben schon vorsorglich die Typ-Nr. eingeführt, die dazu dienen wird, die unterschiedlichen Gelenktypen zu unterscheiden. Alle Gelenke eines MKS sind in einer linearen Liste

$$L := \{G_1, G_2, ...\}$$

vermerkt. Bei Komplettimplementierungen des Algorithmus wird die Liste L an ihrem Ende ständig verändert, weil es sich herausgestellt hat, dass Körperkontakt (ohne Gelenk) und auf einer Unterlage reibende, rollende bzw. ruhende Kontaktstellen temporär als Gelenk (mit bestimmten Eigenschaften) gehandhabt werden sollten, was die Struktur des Algorithmus sehr stark systematisiert.
Wir stellen nun den Basis-Algorithmus des Simulationsverfahrens vor. Wir benötigen allerdings noch folgenden wichtigen Begriff:

Definition:

Ein MKS befindet sich zu einem Zeitpunkt t in einem <u>Dmax-konsistenten Zustand</u>, wenn für alle Gelenke $(0, k_1, i_1, k_2, i_2) \in L$ *gilt*

$$\left| p_{k_1 i_1}(t) - p_{k_2 i_2}(t) \right| < \mathrm{Dmax},$$

wobei Dmax *eine kleine positive Konstante ist (z. B. Dmax = 0.0001 oder Dmax* $= 10^{-6}$*).*

Unser mit iterativer Korrektur von Gelenkbedingungen arbeitendes Verfahren kann die Toleranzabstände in Gelenken nur näherungsweise zu 0 machen. Der Abstandsparameter Dmax spielt daher eine entscheidende Rolle, wenn es um Genauigkeit, Geschwindigkeit und Energieerhaltung geht.

Wir beschreiben nun unseren Simulationsalgorithmus. Vorausgesetzt wird, dass sich das zu simulierende MKS zum Zeitpunkt t in einem Dmax-konsistenten Zustand befindet und dass *h* die vorgesehene Zeitschrittweite ist (P-Schritt = Programmschritt).

<u>P-Schritt 1:</u> *Prüfung der Gelenkbedingungen*

Solange noch ein Gelenk $G = (0, k_1, i_1, k_2, i_2) \in L$ gefunden werden kann mit:

$$p_{k_1 i_1}(t+h) := LookAhead\left(k_1, p_{k_1 i_1}(t), h\right),$$

$$p_{k_2 i_2}(t+h) := LookAhead\left(k_2, p_{k_2 i_2}(t), h\right) \quad und$$

$$\left| p_{k_1 i_1}(t+h) - p_{k_2 i_2}(t+h) \right| \geq \mathrm{Dmax}$$

so korrigiere das Gelenk mit einem „P-Schritt 2“. Wird ein solches Gelenk nicht mehr gefunden, so <u>terminiert</u> der P-Schritt 1.

P-Schritt 2: *Gelenkkorrektur (für Gelenke Typ = 0)*

$$delta := p_{k_2 i_2}(t+h) - p_{k_1 i_1}(t+h);$$

$$\Delta\dot{p} := delta / h;$$

$$v_1 := p_{k_1 i_1}(t) - s_{k_1}(t);$$

$$v_2 := p_{k_2 i_2}(t) - s_{k_2}(t);$$

$$M := M(k_1, v_1, v_1) + M(k_2, v_2, v_2);$$

$$I := M^{-1} \cdot \Delta\dot{p};$$

Aktualisierung des Bewegungszustands:

$$\dot{s}_{k_1}(t) := \dot{s}_{k_1}(t) + \frac{1}{m_{k_1}} \cdot I;$$

$$\dot{s}_{k_2}(t) := \dot{s}_{k_2}(t) + \frac{1}{m_{k_2}} \cdot (-I);$$

$$\omega_{k_1}(t) := \omega_{k_1}(t) + \tilde{J}_{k_1}^{-1}(v_1 \times I);$$

$$\omega_{k_2}(t) := \omega_{k_2}(t) + \tilde{J}_{k_2}^{-1}(v_2 \times (-I));$$

Wenn P-Schritt 1 abgeschlossen ist, also terminiert, so führe aus:

P-Schritt 3: *Fortschalten der Zeit von t nach t+* h *:*

Für jeden Körper k wird ausgeführt: (Vorwärtsbewegung auf ballistischen Bahnen)

(1) Aktualisierung des Drehvektors auf $\omega(t+h)$ (vgl. (3.5))

(2) Berechnung von M_{rot} (vgl. (3.8))

(3) Aktualisierung der körpereigenen Rotationsmatrix

$$R(t+h) = M_{rot} \cdot R(t) \quad \text{(vgl. (3.9))}$$

(4) Berechnung von $s(t+h)$ und $\dot{s}(t+h)$ (vgl. (3.4))

(5) Vorwärtsbewegung von Referenzpunkten p durch

$$p(t+h) = s(t+h) + M_{rot}(p(t) - s(t))$$

Damit ist der Grundalgorithmus vollständig spezifiziert. Wenn man ihn implementiert, so durchläuft man bei P-Schritt 1 mit einer einfachen Schleife alle Gelenke nacheinander und wiederholend und prüft, ob das jeweilige Gelenk den Dmax-Abstand nicht überschreitet. Das Iterationsverfahren von P-Schritt 1 zusammen mit P-Schritt 2 löst offensichtlich ein komplexes, nichtlineares Gleichungssystem. Es werden solche Impulse *I* gesucht, so dass zum Zeitpunkt $t+h$ alle Gelenkbedingungen wieder erfüllt sind. Nach P-Schritt 3 haben wir also wieder einen Dmax-konsistenten Systemzustand, jetzt für die Zeit $t+h$. Das Spiel kann von neuem beginnen.

Was den beschriebenen Algorithmus von allen uns bekannten Verfahren auszeichnet, ist erstens die konsequente Anwendung der Impulstechnik auch für Starrkörpersysteme mit Gelenken und zweitens die LookAhead-Technik. Dabei werden die Gelenkbedingungen in der Zukunft, also zur Zeit $t+h$ geprüft und es wird dann mit komplementären Impulspaaren, die zur Zeit t angewandt werden, versucht, die Gelenkbedingungen zur Zeit $t+h$ alle zu erfüllen. Wie wir das tun, ist sehr anschaulich zu erklären. Wir bestimmen mit $\Delta\dot{p} := delta / h$ diejenige Differenzgeschwindigkeit, die zum Zeitpunkt h zusätzlich zwischen den Gelenkpunkten herrschen müsste, um bei linearen Verhältnissen den Abstand der Gelenkpunkte zum Zeitpunkt $t+h$ zu Null zu machen. Dies kann natürlich nur eine Approximation sein, denn die Bewegung auf ballistischen Bahnen ist nicht linear und wenn ein Gelenk korrigiert wird, werden andere an den beteiligten Körpern befindlichen Gelenke wieder etwas gestört, d. h. die Abstandsbedingung weniger gut eingehalten. Daher kann man berechtigterweise die Frage stellen, ob sich bei der Iteration überhaupt Konvergenz einstellen kann.

Die Erfahrungen mit einer Vielzahl von praktisch durchgeführten Simulationen haben jedoch eindrucksvoll bestätigt, dass dieses Verfahren stets problemlos funktioniert, wenn die mechanischen Modelle keine Defekte aufweisen und die Zeitschrittweite *h* nicht zu groß gewählt ist.

Die Bahnkurve $p_{ki}(t)$ eines Referenzpunktes von Körper k verläuft zwar stetig, hat aber zu den Zeitpunkten, wo iterative Korrekturen stattfinden, also bei $t = 0, h, 2h$ usw. Knicke in der Kurve, also eine nichtstetige Ableitung. Das wird dadurch verursacht, dass die stetigen, an Gelenken einwirkenden inneren Kräfte bei uns durch Impulse an den Haltepunkten $t = 0, h, 2h\ldots$ substituiert werden. Wer allerdings stetige Lösungskurven benötigt, kann sich die leicht durch Interpolation höheren Grades (z. B. 5 und mehr) beschaffen. Die Ableitungen solcher Polynome liefern sowohl für die Geschwindigkeit als auch für die Beschleunigung stetig differenzierbare Werte. Die restlichen Unstetigkeiten, die durch den Übergang von einem Interpolationspolynom zum nächsten entstehen, sind dann nur noch sehr gering.

Wie bei jeder iterativen Vorgehensweise kann durch zu große Schrittweiten *h* oder durch kritische Situationen im Mechanik-Modell Divergenz eintreten. Die Iteration entgleist dann und muss abgebrochen werden, weil sich früher oder später Zahlenüberläufe ergeben. Wir können bei unserem Verfahren sehr früh feststellen, ob sich derartige Probleme einstellen, indem wir während der Iteration für jedes Gelenk $G_j \in L$ die Differenzen

$$diff_j := \left| p_{k_1 i_1}(t+h) - p_{k_2 i_2}(t+h) \right| \geq \mathrm{Dmax}$$

beobachten. Diese sollten monoton absinken. Wird auch nur ein Gelenk entdeckt, bei dem sich *diff* vergrößert, so brechen wir die Iteration ab, halbieren die Zeit-Schrittweite mit $h := h/2$ und setzen mit dieser neuen Schrittweite die Iteration fort. In kritischen Situationen kann sich dieser Vorgang mehrmals wiederholen. Wenn dann später wieder gute Konvergenz beobachtet wird, kann die Zeitschrittweite wieder vergrößert werden. Diese adaptive Schrittweitensteuerung funktioniert bei uns deshalb völlig problemlos, weil bei stillstehender Zeit das Additions-Subtraktions-Prinzip (siehe Abschnitt 4) für komplementäre Impulspaare gilt. Das bedeutet, dass wir beim Vorgang der Schrittweitenänderung keinerlei Korrekturen am Bewegungszustand vornehmen müssen, weil der iterative Korrek-

turalgorithmus so beschaffen ist, dass er vorher eingebrachte „Fehlimpulse“ automatisch wieder eliminiert. Dennoch empfiehlt es sich, vor der iterativen Gelenkkorrektur eine Kopie des Bewegungszustands abzuspeichern, um ihn gegebenenfalls bei Schrittweitenveränderungen wieder als Startkonfiguration übernehmen zu können.

6. Geschwindigkeitskorrektur

Wenn mit dem oben als P-Schritt 3 bezeichneten Fortschalten von Zeit t zur Zeit $t+h$ der neu erreichte Bewegungszustand geprüft wird, so erweist sich dieser als Dmax-konsistent. Im Allgemeinen ist allerdings nicht gesichert, dass die Referenzpunkte an Gelenken die gleichen Geschwindigkeiten aufweisen, wie es eigentlich sein müsste. Wie in Abschnitt 4 schon ausführlich diskutiert wurde, können wir diese Geschwindigkeitsdifferenzen durch Impulskorrekturen beseitigen:

P-Schritt 4: *Iterative Geschwindigkeitskorrektur*

(Unmittelbar nach Ausführen von P-Schritt 3 anzuwenden, falls gewünscht.)
Solange noch ein Gelenk $G=(0,k_1,i_1,k_2,i_2)\in L$ gefunden werden kann, für das gilt

$$\left|\dot{p}_{k_1 i_1}(t+h)-\dot{p}_{k_2 i_2}(t+h)\right|>\mathrm{Vmax} \qquad \left(z.B.\ \mathrm{Vmax}:=10^{-4}\,m/s\right),$$

so korrigiere die Differenzgeschwindigkeit wie folgt:

$$\Delta\dot{p}:=\dot{p}_{k_2 i_2}(t+h)-\dot{p}_{k_1 i_1}(t+h)$$

Sodann verfahren wir wie bei P-Schritt 2:

$$v_1:=p_{k_1 i_1}(t+h)-s_{k_1}(t+h);$$
$$v_2:=p_{k_2 i_2}(t+h)-s_{k_2}(t+h);$$
$$M:=M(k_1,v_1,v_1)+M(k_2,v_2,v_2);$$
$$I:=M^{-1}\cdot\Delta\dot{p};$$

Als nächstes aktualisieren wir mit dem komplementären Impulspaar *I, -I* den Bewegungszustand der beteiligten Körper. Allerdings werden hier alle Größen zur Zeit $t+h$ betrachtet, ohne Lookahead.

Sehr überraschend ist nun, dass die Geschwindigkeitskorrektur keinen Einfluss auf die Simulation hat. Ob mit oder ohne, man erhält stets – bei hinreichend kleinem Dmax – die gleichen Bahnkurven bzw. Punktorte. Die Geschwindigkeitskorrektur scheint also nur eine kosmetische Maßnahme zu sein, die aber deutlich Rechenzeit erfordert. Zunächst war uns völlig unklar, wie dies überhaupt möglich ist, bis wir uns an das Additions-Subtraktions-Prinzip (siehe Abschnitt 4) erinnerten. Die Impulse aus der Geschwindigkeitskorrektur werden also wieder „vergessen“ bzw. eliminiert durch die regulären Gelenkkorrekturimpulse.

Allerdings muss hier auch angemerkt werden, dass mit dem Verfahren der Geschwindigkeitskorrektur dynamische Probleme behandelt werden können, die mit den klassischen Dynamik-Verfahren nicht lösbar sind. Will man z. B. in gelenkgekoppelten Mehrkörpersystemen Stöße und Kollisionen simulieren, so geht das nur mit dem Verfahren der

Geschwindigkeitskorrektur. Näheres dazu wird in Abschnitt 11 ausgeführt. Man vergleiche auch Kap. 6 in Wittenburg [1977].

7. Bestimmungen von Korrekturimpulsen mit einem Gesamt-Gleichungssystem

In Abschnitt 4 konnten wir zeigen, dass die noch unbekannten Impulse während der Iteration durch ein lineares Gleichungssystem (zunächst nur der Ordnung 3) berechnet werden können. Wir nutzen dies in P-Schritt 2 und in P-Schritt 4 (Geschwindigkeitskorrektur). Analysiert man das Problem der Berechnung der Korrekturimpulse genauer, so kann man ein wesentlich größeres Gesamt-Gleichungssystem formulieren. Für jedes Gelenk G_j muss ein Impuls I_j bestimmt werden, so dass alle Geschwindigkeitsdifferenzen $\Delta\dot{p}_j$ abgebaut werden. Das ergibt ein großes lineares Gleichungssystem der Struktur

$$\text{(7.1)} \qquad \begin{pmatrix} M_{11} & M_{12} & \cdots & M_{1m} \\ M_{21} & M_{22} & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots \\ M_{m1} & M_{m2} & \cdots & M_{mm} \end{pmatrix} \begin{pmatrix} I_1 \\ I_2 \\ \vdots \\ I_m \end{pmatrix} = \begin{pmatrix} \Delta\dot{p}_1 \\ \Delta\dot{p}_2 \\ \vdots \\ \Delta\dot{p}_m \end{pmatrix}.$$

m ist die Zahl der Gelenke, also $L = \{G_1, G_2, \ldots, G_m\}$. Die 3x3-Matrizen M_{ii} in der Hauptdiagonale kennen wir bereits, denn mit $G_i = \left(0, k_1, i_1, k_2, i_2\right)$ gilt

$$M_{ii} = M\left(k_1, v_{k_1 i_1}, v_{k_1 i_1}\right) + M\left(k_2, v_{k_2 i_2}, v_{k_2 i_2}\right).$$

Eine Zeile

$$\text{(7.2)} \qquad \sum_{i=1}^{m} M_{ji} \cdot I_i = \Delta\dot{p}_j$$

des Gleichungssystems steht für ein Gelenk $G_j = \left(0, k_1, i_1, k_2, i_2\right)$ und verknüpft alle I_i, die sowohl auf Körper k_1 als auch auf Körper k_2 einwirken. Dabei gilt: I_i wirkt auf Körper k ein, wenn k einer der Körper des Gelenks $G_i = \left(0, k_1, i_1, k_2, i_2\right)$ ist. Wenn ein Körper also z. B. nur zwei Gelenke hat, so wirken auch nur die zu den Gelenken gehörenden Korrekturimpulse auf den Körper ein. Das bedeutet, dass in der Regel nur wenige der M_{ji} in (7.2) von der 3x3-Nullmatrix verschieden sind, was zur Folge hat, dass die große Gesamtmatrix $\overline{\overline{M}} = \left(M_{ij}\right)$ dünn besetzt ist. Wir verzichten hier darauf, die Formel zur Bestimmung der Gesamtmatrix $\overline{\overline{M}}$ anzugeben, wollen aber über Erfahrungen mit diesem großen Gleichungssystem berichten.

Auch $\overline{\overline{M}}$ ist während einer Gelenkkorrekturphase konstant. Man kann also, um sämtliche *I´*s auf einen Schlag zu bestimmen, das lineare Gleichungssystem (7.1) lösen. Für die Geschwindigkeitskorrektur ist dies völlig ausreichend. Für die Gelenkkorrektur jedoch nicht,

weil das Korrekturverfahren wegen der LookAhead-Technik ein approximierender Prozess ist. Daher muss das Gleichungssystem

$$\bar{\bar{M}} \cdot \left(I_j \right) = \left(\Delta \dot{p}_j \right)$$

mehrmals gelöst werden, jeweils mit neu bestimmter rechter Seite $\left(\Delta \dot{p}_j \right)$, denn auf diese Relativgeschwindigkeiten wirken über die Funktion Lookahead zusätzliche Störeinflüsse, die im Gleichungssystem nicht einkalkuliert sind. Es handelt sich um rotatorische Einflüsse und Nutationseffekte, die von der ballistischen Bewegung in nicht vorhersehbarer Weise auf den Iterationsprozeß zur Bestimmung der Korrekturimpulse einwirken. Da das Gleichungssystem bei komplexeren Mechanikmodellen also mehrmals gelöst werden muss, bei still stehender Zeit $\bar{\bar{M}}$ aber konstant ist, kommt auch die Invertierung von $\bar{\bar{M}}$ in Frage, um mit

$$\left(I_j \right) = \bar{\bar{M}}^{-1} \cdot \left(\Delta \dot{p}_j \right)$$

zu operieren und dadurch die Unbekannten I_j schneller zu bestimmen.

Schließlich soll hier noch auf die Arbeit von Baraff [1996] hingewiesen werden, der zeigte, dass dünn besetzte Matrizen ähnlich zu $\bar{\bar{M}}$ spezielle sehr schnelle Lösungsverfahren erlauben. Er betrachtete das Problem, Lagrange-Multiplikatoren zu bestimmen. Auch beim Einsatz iterativer Lösungsverfahren wie z. B. Gauß-Seidel für das Gleichungssystem (7.1) ergeben sich bei geeigneter Implementierung durch die dünne Besetzung erhebliche Beschleunigungen.

Wir haben das Gesamtmatrix-Verfahren in verschiedenen Varianten implementiert und grobe Zeitmessungen durchgeführt. Bei einfacheren passiven Modellen wie dem Dreifachpendel und einer geschlossenen kinematische Ketten mit 8 Gliedern (Modell 8-Loop, siehe Abschnitt 9.) wurden für einen Iterationsschritt nur 1 bis 3 Lösungen von (7.1) benötigt. Bei komplexeren Modellen wie einer 6-beinige Laufmaschine brachte die Gleichungssystem-Methode keinen Gewinn. $\bar{\bar{M}}$ war bei diesem Modell sogar singulär und musste iterativ (mit Gauß-Seidel) gelöst werden.

Insgesamt konnten wir feststellen, dass die simple Iterationstechnik (P-Schritt 1 und 2) mit dem Gleichungssystem-Verfahren voll konkurrenzfähig und sehr viel einfacher zu implementieren ist. Sie ist auch erheblich flexibler, wenn unterschiedliche Gelenktypen zu verarbeiten sind. Nach dem derzeitigen allerdings noch unsicheren Kenntnisstand darf man also von den Methoden mit Gesamt-Gleichungssystem keine Wunderdinge erwarten.

8. Fehleranalyse und Verfahren höherer Ordnung

Um eine Fehlerabschätzung für die impulsbasierte Simulationsmethode zu gewinnen, wollen wir die Differentialgleichung

$$\ddot{p}(t) = \frac{1}{m} F(t) \tag{8.1}$$

für einen von der Kraft *F(t)* bewegten Punkt mit Masse *m* lösen. *F(t)* sei bekannt und für $t \geq 0$ als Potenzreihe

$$F(t) = \sum_{i=0}^{\infty} a_i t^i = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + \ldots$$

gegeben. Wir lösen nun zunächst (8.1) mit der impulsbasierten Methode für das Zeitintervall $t \in [0, h]$. Für den anzuwendenden Impuls erhalten wir

$$I(h) = \int_0^h F(t)dt = a_0 h + \frac{a_1}{2} h^2 + \frac{a_2}{3} h^3 + \ldots.$$

Ein für ein Zeitintervall $t \in [0, h]$ berechneter Impuls ist ein Mittelwert, der zur Zeit $t = h/2$, also in der Mitte des Zeitintervalls, angewandt werden muss, um zum Zeitpunkt h eine gute Näherung an die korrekte Lösung von (8.1) zu erzielen. Also ergibt sich bei dieser Vorgehensweise

(8.2)
$$\begin{aligned} p_{(1)}(h) &= p(0) + h\dot{p}(0) + \frac{h}{2m} I(h) = \\ &= p(0) + h\dot{p}(0) + \frac{1}{m}\left(\frac{a_0}{2} h^2 + \frac{a_1}{4} h^3 + \frac{a_2}{6} h^4 + \ldots\right). \end{aligned}$$

Wenn wir (8.1) direkt integrieren, so erhalten wir die exakte Lösung

(8.3)
$$p(h) = p(0) + h\dot{p}(0) + \frac{1}{m}\left(\frac{a_0}{2} h^2 + \frac{a_1}{6} h^3 + \frac{a_2}{12} h^4 + \ldots\right).$$

Vergleicht man nun die Koeffizienten zu den Potenzen von *h*, so stellt man fest, dass diese bis h^2 übereinstimmen und dass sich erst bei h^3 eine Differenz ergibt ($\frac{a_1}{4}$ gegenüber $\frac{a_1}{6}$).

Der Einschritt-Fehler ist also von der Ordnung $O(h^3)$, der über eine feste Zeitspanne bei variablem *h* auflaufende Gesamtfehler von der Ordnung $O(h^2)$. Diese Fehlerordnung konnte bei zahllosen Testsimulationen mit den unterschiedlichsten Mechanik-Modellen beobachtet werden.

Bei den Gelenkkorrekturen (vgl. P-Schritt 1 und 2) wenden wir die Korrekturimpulse scheinbar zum falschen Zeitpunkt an, nämlich am Anfang des *h*-Intervalls. Das ist kein Widerspruch zu der obigen Rechnung, denn die Korrekturimpulse werden unter der Prämisse berechnet, dass sie zum Zeitpunkt *t=0* angewendet werden. Sie sind nicht das Ergebnis einer Integration, sondern sollen zum Zeitpunkt *h* die Gelenkbedingungen wiederherstellen.

Die obige Methode der Fehlerabschätzung kann genutzt werden, um impulsbasierte Iterationsverfahren höherer Ordnung zu gewinnen. Mit

$$p_{(1)}(h) = p(0) + h\dot{p}(0) + \frac{1}{m}\left(\frac{a_0}{2} h^2 + \frac{a_1}{4} h^3 + \frac{a_2}{6} h^4 + \ldots\right)$$

bezeichnen wir das oben bereits berechnete Integrationsergebnis für *p(h)* mit einer Impulsanwendung in der Mitte des *h*-Intervalles. Ein genaueres Ergebnis erzielt man, indem man zuerst für das Teilintervall [0,h/2] einen Impuls I_1 berechnet, für das Teilintervall [h/2,h] den Impuls I_2 und diese nacheinander zu den Zeitpunkten h/4 (Zeitpunkt für I_1) und 3h/4 (Zeitpunkt für I_2) auf den Massenpunkt einwirken lässt. Bei dieser zweischrittigen Methode erhält man

$$I_1(h) = \int_0^{h/2} F(t)dt = \frac{a_0}{2}h + \frac{a_1}{8}h^2 + \frac{a_2}{24}h^3 + \ldots$$

sowie

$$I_2(h) = \int_{h/2}^{h} F(t)dt = \frac{a_0}{2}h + \frac{3a_1}{8}h^2 + \frac{7a_2}{24}h^3 + \ldots,$$

also

$$p_{(2)}(h) = p(0) + h\dot{p}(0) + \frac{1}{m}(\frac{3h}{4}I_1(h) + \frac{h}{4}I_2(h))$$

$$= p(0) + h\dot{p}(0) + \frac{1}{m}(\frac{a_0}{2}h^2 + \frac{3a_1}{16}h^3 + \frac{5a_2}{48}h^4 + \ldots).$$

Ein noch wesentlich genaueres Ergebnis für p(h) kann nun so berechnet werden:

$$p_3(h) = (-1/3)p_1(h) + (4/3)p_2(h)$$

$$= p(0) + h\dot{p}(0) + \frac{1}{m}(\frac{a_0}{2}h^2 + \frac{a_1}{6}h^3 + \frac{a_2}{12}h^4 + O(h^5)).$$

Wie der Vergleich mit (8.3) bestätigt, liegt hier Übereinstimmung mit der exakten Lösung bis zu Termen der Ordnung h^4 vor. Die oben benutzten Faktoren (-1/3) und (4/3) wurden mit einem Gleichungssystem bestimmt. Wie sich dieses Ergebnis umsetzen lässt, um die Fehlerordnung der impulsbasierten Simulationsmethode deutlich zu erhöhen, konnte noch nicht geklärt werden, denn diese Fehlerrechnung ist zunächst als fiktiv zu betrachten. Sie bezieht sich nur auf das simple Mechanikmodell des von Kräften bewegten Massenpunktes.

9. Ergebnisse von Testsimulationen

Wir wollen nun einige Simulationsergebnisse diskutieren. Als erstes betrachten wir die Simulation eines 1-m-Pendels mit einem Ausschlag von 10 Grad gegen die Senkrechte, siehe Abb. 9.1.

Für das mathematische Pendel mit definierter Schwingweite ist die Schwingungszeit bekannt. Der physikalisch korrekter Wert ist T_0=2.00989262729860 Sek. und kann mit der aus der theoretischen Physik bekannten Formel

$$T_0 = 2\pi\sqrt{\frac{R}{g}}\left\{1+\left(\frac{1}{2}\right)^2 \sin^2\left(\frac{\varphi}{2}\right)+\left(\frac{1\cdot 3}{2\cdot 4}\right)^2 \sin^4\left(\frac{\varphi}{2}\right)+\ldots\right\}$$

berechnet werden, wobei *R* die Pendellänge und φ der Ausschlagwinkel ist. Simuliert wurde mit *g*=9.81 und *Dmax*=10^{-9}.

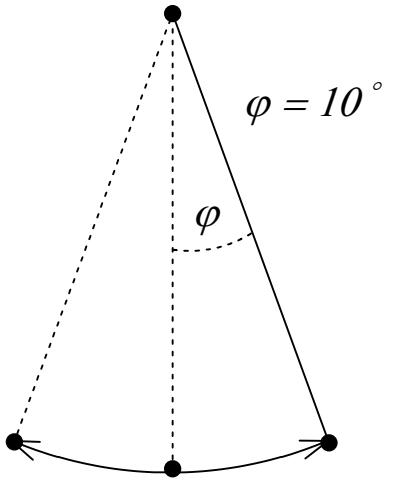

Abbildung 9.1: *Mathematisches 1-m-Pendel mit 10 Grad Ausschlag.*

h	T	$T-T_0$
0.04	2.0085445249	0.00134810239666
0.02	2.0095560286	0.00033659873943
0.01	2.0098089140	0.00008371325490
0.005	2.0098717016	0.00002092568899
0.0025	2.0098873957	0.00000523163119
0.00125	2.0098913212	0.00000130613721
0.000625	2.0098923000	0.00000032733840
0.0003125	2.0098925467	0.00000008061132
0.00015625	2.0098926095	0.00000001779746

Tabelle 9.1: *Impulsbasierte Simulation des 1-m-Pendels.* h *= Zeitschrittweite,* T *= Schwingungszeit des simulierten Pendels,* $T-T_0$ *=Abweichung vom physikalisch korrekten Wert. Die Schwingungszeiten werden durch Interpolation der Zeit für Nulldurchgänge des Pendels bestimmt.*

Aus der Tabelle (9.1) kann folgendes abgelesen werden: Von Zeile zu Zeile wird die Schrittweite h halbiert, die Schwingungszeit T vergrößert sich dadurch minimal. Der Fehler $T-T_0$, also die Differenz zum physikalisch korrekten Wert, reduziert sich dabei jeweils um den Faktor 1/4. Dies entspricht dem in Abschnitt 8 theoretisch ermittelten Fehlerverhalten. Bei dem häufig als Standard-Schrittweite benutzten Wert von h=0.01 ist der Zeitfehler für eine volle Schwingung also nur noch ca. 83 μs.

Das in Abschnitt 8 entwickelte Verfahren höherer Ordnung mit der Formel $p_3(h) = (-1/3)p_1(h) + (4/3)p_2(h)$ kann mit Hilfe der Tabelle überprüft werden:

Nimmt man statt $p_1(h)$ die Schwingungszeit für h=0.01, für $p_2(h)$ diejenige für 0.005, so ergibt sich 2.0098926307999996 = -(1/3)*2.0098089140+(4/3)*2.0098717016, und das ist erheblich genauer als der Wert für $p_2(h)$. Die Differenz zum physikalisch korrekten Wert T_0 ist nur noch $3.5 \cdot 10^{-9}$, während diese bei der oben tabellierten kleinsten Schrittweite von *h* = 0.00015625 immerhin noch $1.7 \cdot 10^{-8}$ beträgt. Dies deutet darauf hin, dass das in Abschnitt 8 gewonnene Verfahren höherer Ordnung in der Tat einen erheblichen Genauigkeitsgewinn verspricht. Ob sich diese pauschal hochgerechneten Ergebnisse auch bei der echten impulsbasierten Simulation zweiter Ordnung einstellen werden, ist noch offen und bleibt zukünftigen Untersuchungen vorbehalten.

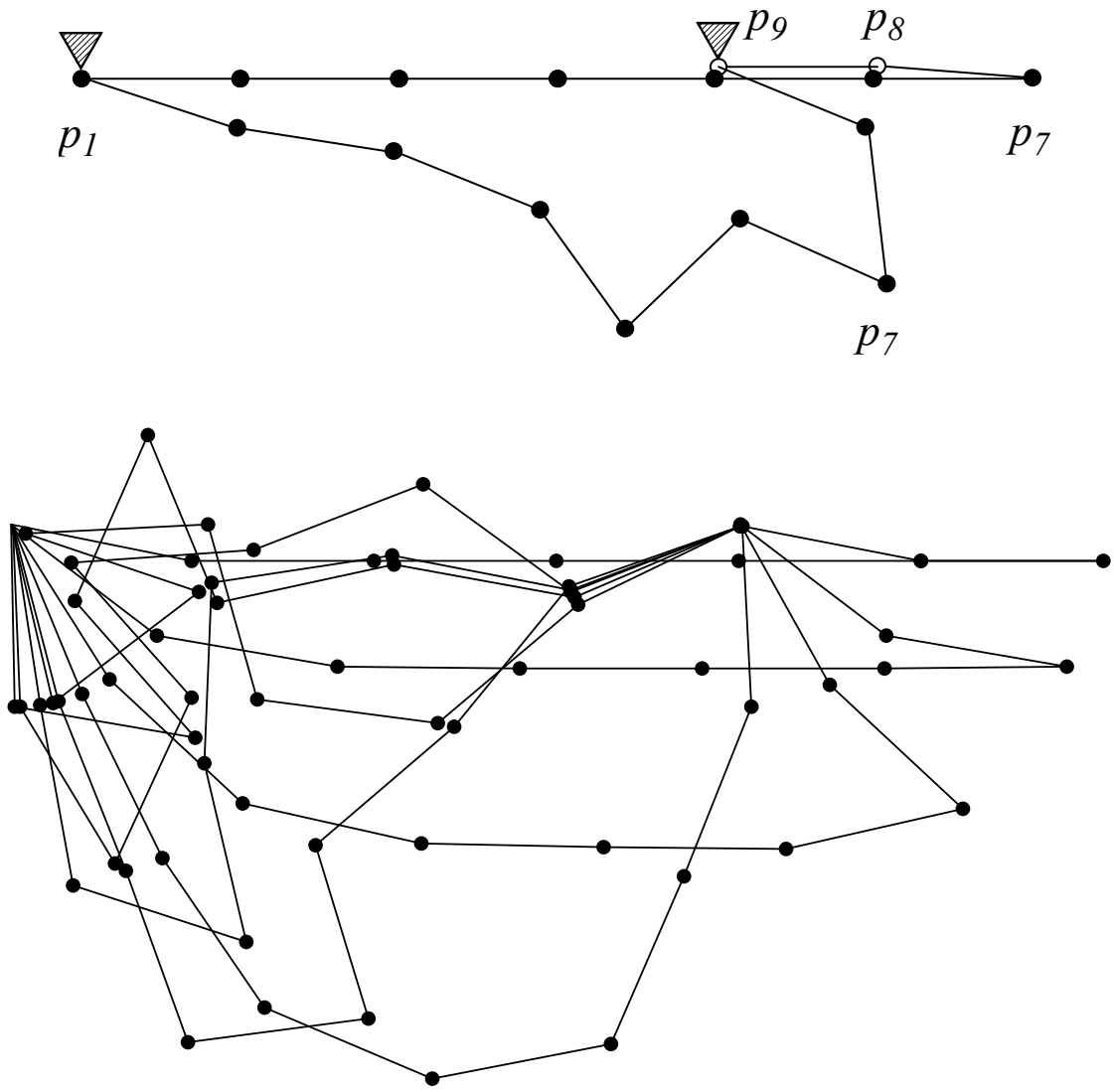

Abbildung 9.2:: *Modell 8-Loop. Oben: Die miteinander verketteten Massenpunkte* $p_1 \ldots p_9$ *bilden eine Kette, wobei* p_1 *und* p_9 *Aufhängepunkte sind. Formal handelt es sich um 8 gelenkverbundene Starrkörper bzw. Stangen. Unten: Wenn die Kette aus der waagerechten Ruhelage freigegeben wird, bewegt sie sich stark chaotisch. Aufgezeichnet sind die Konfigurationen für die Zeiten 0.2, 0.4,...2.0 Sek.*

Als nächstes diskutieren wir Simulationsergebnisse für das Modell 8-Loop. Es besteht aus 9 verketteten Massenpunkten, die an beiden Enden aufgehängt sind, siehe Abb. 9.2.

Zur Zeit 0 befinden sich alle Kettenglieder auf Höhe der Aufhängepunkte in Ruhe, das Modell wird dann durch die Schwerkraft bewegt. Kollisionen zwischen Massenpunkten werden nicht beachtet. Es resultiert eine ziemlich chaotische Bewegung. Die Simulationszeit beträgt 10 Sekunden. Während dieser Zeit zählen wir die Zahl der Korrekturschritte pro Zeitintervall *h* bzw. pro Sekunde und bilden Mittelwerte, die in den Tabellen 9.2 und 9.3 angegeben werden.

	Dmax →					
h ↓	0.001	0.0001	0.00001	0.000001	0.0000001	0.00000001
0.02	48.39	92.73	158.40	254.68	364.99	443.07
0.01	24.63	60.68	113.08	199.96	299.45	370.53
0.005	13.14	39.88	81.36	144.80	223.30	336.23
0.0025	9.51	20.33	51.38	101.46	171.46	286.42
0.00125	9.01	10.93	28.47	68.49	126.56	215.85
0.000625	9.00	9.13	15.56	44.85	95.81	161.56
0.0003125	9.00	9.00	10.00	24.86	59.24	112.94
0.00015625	9.00	9.00	9.03	13.19	36.72	76.30

Tabelle 9.2: *Mittlere Zahl von Korrekturschritten, also P-Schritt 1, in einem h-Zeitintervall. Modell 8-Loop, Simulationszeit 10 Sek.*

Man erkennt deutlich, dass bei gleich bleibender Zeitschrittweite h umso mehr Iterationsschritte benötigt werden, je kleiner Dmax wird. Da das Modell 9 Gelenke hat, muss man die Zahlen durch 9 dividieren, um die mittlere Zahl von Korrekturen für ein Gelenk zu erhalten.

	Dmax →					
h ↓	0.001	0.0001	0.00001	0.000001	0.0000001	0.00000001
0.02	2419	4636	7919	12734	18249	22153
0.01	2462	6068	11307	19995	29944	37053
0.005	2627	7975	16271	28959	44659	67245
0.0025	3805	8132	20550	40582	68585	114566
0.00125	7209	8747	22777	54795	101249	172677
0.000625	14400	14601	24896	71761	153293	258496
0.0003125	28799	28802	32001	79539	189572	361410
0.00015625	57597	57597	57791	84443	235024	488322

Tabelle 9.3: *Mittlere Zahl von Korrekturschritten, also P-Schritt 1, in einer Sekunde Simulationszeit. Modell 8-Loop, Simulationszeit 10 Sek.*

In der Tabelle 9.2 erkennt man unten links, dass hier vielfach mit der Minimalzahl von 9 Korrekturschritten simuliert wurde, was natürlich darauf zurückzuführen ist, dass h sehr klein und Dmax groß ist. In diesen Fällen wird also nur ein Korrekturschritt pro Gelenk benötigt.

Aus der Tabelle 9.3 kann man ablesen, wie sich die Simulationsgeschwindigkeit abhängig von den wichtigen Parametern Dmax und h ändert. Dazu muss man wissen, dass auf einem PC mit ca. 1.7 GHz pro Sekunde mehr als 670000 Korrekturschritte ausgeführt werden. Aus der Tabelle liest man ab, dass bei h=0.01 und Dmax= 0.00001 pro Sekunde im Mittel 11307 Korrekturschritte erforderlich sind. Rein rechnerisch müsste die Simulation also ca. 60 mal schneller als Realzeit ablaufen. Gemessen wurde 64-fache Realzeit.

Tabelle 9.4 präsentiert erfasste Daten zum Energiefading. Sie macht indirekt auch Aussagen über die Genauigkeit der Simulation. Ist das Energiefading deutlich, so ist auch ein entsprechender Fehler in den Simulationsdaten enthalten. Aus der Tabelle liest man ab, dass sich nur bei großen Werten von Dmax und h deutliche Abweichungen von der Energiesumme 0 ergeben.

	Dmax →					
$h\downarrow$	0.001	0.0001	0.00001	0.000001	0.0000001	0.00000001
0.02	8.34888268	3.91859475	1.18293503	1.76430215	1.96143197	2.28035856
0.01	0.73559002	0.18926225	0.20830578	0.25642428	0.23072030	0.25728820
0.005	0.38693763	0.08356390	0.04945572	0.05274739	0.05417435	0.05432397
0.0025	0.71407485	0.05918295	0.01288742	0.01401945	0.01286931	0.01426934
0.00125	0.18258168	0.01396812	0.00373649	0.00337147	0.00339436	0.00314362
0.000625	0.06504896	0.02756842	0.00183589	0.00088356	0.00085766	0.00089008
0.0003125	0.01170068	0.00561310	0.00132198	0.00034599	0.00022715	0.00020310
0.00015625	0.00203239	0.00203239	0.00138729	0.00016536	0.00006400	0.00005337

Tabelle 9.4: *Energiefading bei Modell 8-Loop während der Simulationszeit von 10 Sek. Abgedruckt sind die jeweiligen Mittelwerte von* $\left|E_{kin} + E_{pot}\right|$*, also der Absolutbeträge der kinetischen und potentiellen Energie des simulierten Modells. Diese sollten im Idealfall gleich 0 sein, weil es sich um ein passiv schwingendes Modell handelt.*

Wir haben z. B. das obige Modell 8-Loop auch mit einem weit verbreiteten kommerziellen System für die dynamische Simulation simuliert, bei *h*=0.01. Die dabei jeweils nach 1, 2, 3,... Sekunden Simulationszeit punktuell erfassten Energiefehler sind in Tabelle 9.5 wiedergegeben. Alle Zahlenwerte sind im m-kg-sek-System dimensioniert. Der Tabelle entnimmt man, dass die impulsbasierte Methode, hier mit Dmax=0.00001 simuliert, die Energie wesentlich besser konserviert als die kommerzielle Methode auf der Basis von Differentialgleichungen. Wir vermuten, dass das hochgradig chaotische Verhalten des Modells 8-Loop mit vielen plötzlichen Kraftsprüngen für diese Differenzen verantwortlich ist. Möglicherweise wurde bei dem kommerziellen System nicht mit einer optimalen Parametereinstellung bzw. mit einer ungeeigneten Integrationsmethode gearbeitet. Bei Mechanikmodellen mit nicht so sprunghaften Änderungen der Beschleunigungen kann man von den Verfahren mit Differentialgleichungen erheblich besseren Energieerhalt erwarten.

Simulationszeit Sek.	impulsbasiert	kommerzielles Dynamik-System
1.00	-0.03867	2.50409
2.00	-0.03359	3.05036
3.00	-0.02942	7.99727
4.00	-0.07474	7.67494
5.00	-0.01186	9.34529
6.00	-0.01703	6.92466
7.00	-0.11441	8.24925
8.00	0.00264	10.35616
9.00	-0.04581	8.52872
10.00	-0.05355	10.28015

Tabelle 9.5: *Vergleich des Energiefehlers $E_{kin} + E_{pot}$ zwischen der impulsbasierten und der Simulation mit einem weit verbreiteten kommerziellen System. Die Simulation ist bezüglich der Energieerhaltung perfekt, wenn der Energiefehler gleich 0 ist.*

10. Erweiterung des Gelenktyp-Repertoires

Einer der größten Vorteile der impulsbasierten Methode ist es, dass man spezifische Gelenkeigenschaften nicht in Differentialgleichungen oder algebraische Bedingungsgleichungen („constraints“) einbauen muss. Ein neu zu schaffender elementarer Gelenktyp erfordert natürlich zunächst ein Tupel-Format

$$G = \left(Typ, \{Parameter\}\right),$$

um solche Gelenke in die Gelenkliste L einfügen zu können. „*Typ*“ ist eine neue, noch nicht vergebene Typzahl. Darüber hinaus muss der neue Gelenktyp bei P-Schritt 1 und P-Schritt 2 durch ein entsprechendes Unterprogramm verarbeitet, also in das iterative, impulsbasierte Korrekturverfahren eingebunden werden.
In einigen Fällen lassen sich neue Gelenktypen bausteinartig aus den bereits vorhandenen Gelenktypen aufbauen. Das gilt z. B. für *Drehachsen* zwischen zwei Körpern, die offensichtlich durch zwei Punkt-zu-Punkt-Verbindungen, also durch zwei Gelenke des Typs 0 realisiert werden können. Man könnte für Drehachsen auch einen eigenen Gelenktyp einführen, also z. B.

$$G = \left(Typ_{Drehachse}, k_1, i_1, i_2, k_2, i_3, i_4\right),$$

wobei i_1, i_2 zwei Achspunkte in Körper k_1 und i_3, i_4 die korrespondierenden Achspunkte in Körper k_2 indizieren. Da ein solches Gelenk bezüglich der Rechenzeit nur kleine Verbesserungen gegenüber der Lösung mit zwei Punkt-zu-Punkt-Gelenken erlaubt, ist der Vorteil nicht erheblich.
Viele Kombinationsmöglichkeiten eröffnet auch die einfache *Schienenführung* eines Referenzpunktes. Dabei werden im Körper k_1 zwei Punkte p_{11}, p_{12} als Endpunkte der Geraden Schiene definiert. Diese Gerade darf vom Referenzpunkt p_2 des Körpers k_2 nicht verlassen werden, d. h. der Punkt des zweiten Körpers, den wir Gleitpunkt nennen wollen, darf

sich frei auf der Schiene bewegen, hat aber an den Enden der Schiene jeweils einen Anschlag, den er nicht überwinden kann.

Die Schienenführung wird wie folgt in das iterative Gelenk-Korrekturverfahren integriert: Mit LookAhead wird die Lage der Schiene und des Gleitpunktes zur Zeit h bestimmt. Der minimale Abstandsvektor *delta* des Gleitpunkts von der Schiene – Endanschläge evtl. inbegriffen – wird der weiteren Korrektur zugrunde gelegt, vgl. P-Schritt 2 für die weitere Verarbeitung von *delta*. Angriffspunkt des korrigierenden Impulses I ist der Gleitpunkt zur Zeit 0, und zwar für beide Körper, weil der Punkt höchstens den Abstand Dmax von der Schiene hat. Die Schienenführung ist ein elementarer Gelenktyp mit z. B. der Tupelkodierung $G = (1, k_1, i_1, i_2, k_2, i_3)$. Mit einer Schiene, aber zwei Gleitpunkten im zweiten Körper erhalten wir ein *Schiebe-Drehelement*. Dies kann also durch zwei Schienenführungen mit jeweils identischer Schiene und verschiedenen Gleitpunkten im 2. Körper realisiert werden. Mit einer weiteren parallelen Schienenführung erhalten wir schließlich eine *Schlittenkonstruktion*. Spätestens bei der Schlittenkonstuktion erscheint es ratsam zu sein, diese nicht aus zwei Schienenführungen zusammenzusetzen, sondern einen eigenen Gelenktyp einzuführen, denn die Rechenzeiteinsparungen dürften deutlicher ausfallen.

Allgemein ist es möglich, Gleitpunkte auf Raumkurven und Raumflächen zu fixieren und damit auch Schraubbewegungen usw. zu realisieren.

Ein weiteres wichtiges konstruktives Element ist die *Federkraft mit oder ohne Dämpfung*. Dieser Gelenktyp benötigt als Parameter zwei Referenzpunkte: p_1 in Körper 1 und p_2 in Körper 2. Außerdem einen Abstand $Dist$ dieser beiden Punkte, bei dem die Federkraft 0 ist, eine Federkonstante K_f und evtl. eine Dämpfungkonstante K_d, in Tupelkodierung also z. B.

$$G = (\#_{DämpfFeder}, k_1, i_1, k_2, i_2, Dist, K_f, K_d).$$

Sowohl die lineare Federkraft als auch die von der Relativgeschwindigkeit zwischen p_1 und p_2 abhängende Dämpfungskraft können nur als Impulse auf die beteiligten Körper einwirken, weil uns in unserer Theorie kontinuierliche Kräfte nicht zur Verfügung stehen.

Mit $\Delta p(t) := p_2(t) - p_1(t), \quad \Delta_1 p(t) = \Delta p(t) / |\Delta p(t)|$ (letzteres ist der auf Länge 1 normierte Vektor $\Delta p(t)$) erhalten wir die zur Zeit t wirkende Federkraft

$$K(t) := K_f \cdot (|\Delta p(t)| - Dist) \cdot \Delta_1 p(t).$$

Also sollte der zur Zeit $t = 0$ anzuwendende Federimpuls mit

$$I := \int_{-h/2}^{+h/2} K(t)\,dt \tag{9.1}$$

bestimmt werden. Für dieses Integral ist

$$I \approx K_f \cdot (|\Delta p(0)| - Dist) \cdot \Delta_1 p(0) \cdot h \tag{9.2}$$

eine passable Näherung, die zu relativ genauen Simulationsergebnissen führt, Details dazu später. Da alle in *I* eingehenden Größen zum Zeitpunkt $t = 0$ bereits feststehen und unveränderlich sind, wird der Impuls I gleich am Anfang der iterativen Gelenkkorrekturen angewandt, und zwar $+I$ auf den Punkt $p_1(0)$ von Körper 1 und $-I$ auf den Punkt $p_2(0)$ von Körper 2. Federkräfte mit oder ohne Dämpfung sind also bei der oben vorgeschlagenen approximativen Verarbeitung mit sehr wenig Rechenaufwand implementierbar, so dass auch umfangreiche Mehrkörpersysteme mit vielen Feder-Elementen in Realzeit simuliert werden können.
Selbstverständlich könnte man das Impulsintegral (9.1) mit numerischen Methoden sehr genau bestimmen, wobei aber die Werte von $p_1(h)$ und $p_2(h)$ wegen der iterativen Gelenkkorrekturen noch nicht endgültig feststehen. Wollte man auch dieses Teilproblem sehr korrekt lösen, so müsste man auch den Federkraftimpuls in die iterative Korrektur einbeziehen.
Da dies eine für das Iterationsverfahren auch aus theoretischer Sicht sehr wichtige Verfahrenweise ist, soll sie genauer erläutert werden:
Man berechnet bei der iterativen Gelenkkorrektur bei stillstehender Zeit $t = 0$ jeweils auch Federkraftimpulse $I_1, I_2, \ldots$. Bei der Iteration werden die Werte $p_i(h) = LookAhead(k_i, p_i(0), h)$ immer genauer, also auch die I_i . Daher wird I_1 voll auf die beteiligten Körper angewandt, sodann $I_2 - I_1$ usw., allgemein $I_i - I_{i-1}$. Am Ende der Iteration wurde dabei also genau der Federkraftimpuls angewandt, der dem Ergebnis der Iterationskorrektur entspricht und dieser Impuls ist voll in die Gelenkkorrekturen eingegangen. Das entspricht also – abgesehen von der Diskretisierung durch die Impulstechnik – genau dem Einbezug von Federkräften in Differentialgleichungssysteme.

Dieser relativ hohe Aufwand scheint in der Regel nicht notwendig zu sein. Um die mit der Näherung (9.2) erzielbare Simulationsgenauigkeit zu prüfen, haben wir ein einfaches Masse-Feder-System simuliert, das genau der Differentialgleichung $\ddot{y}(t) = -y(t)$ mit den Anfangswerten $y(0) = 1, \quad \dot{y}(0) = 0$ entspricht. Lösung ist die Funktion $y(t) = \cos(t)$ mit der Schwingungszeit $T = 2\pi$. Abhängig von der Schrittweite h ergaben sich in der Simulation die in Tabelle 10.1 aufgelisteten Ergebnisse.

Zeitschrittweite h	%-Abweichung von $T = 2\pi$	Mittelwert $\lvert y(t) - \cos(t) \rvert$
0.04	-0.0733752740	0.0003663015
0.02	-0.0183389285	0.0000917266
0.01	-0.0045842580	0.0000229515
0.005	-0.0011459594	0.0000057404
0.0025	-0.0002864587	0.0000014354
0.00125	-0.0000716143	0.0000003589
0.000625	-0.0000179040	0.0000000897

Tabelle 10.1: Impulsbasierte Simulation eines Masse-Feder-Systems mit der Bewegungsgleichung $\ddot{y}(t) = -y(t)$ und der exakten Lösung $y(t) = \cos(t)$. Der Fehler in der Schwingungszeit T in % ist in der

zweiten Spalte, die mittlere Abweichung der Simulation von der exakten Lösung - gemittelt über einen Zeitraum von 17.2 Sekunden - befindet sich in der dritten Spalte.

Wie wir in Abschnitt 9 gesehen haben, kann man erst ab einer Schrittweite von $h = 0.01$ mit guten Simulationsergebnissen rechnen. Die oben dokumentierte Abweichung von 0.0045842580 % bei einer Gesamt-Zeit von 17.2 Sekunden möchten wir als überraschend genaues Ergebnis bezeichnen. Auch die absolute mittlere Abweichung von der exakten Lösung von 0.0000229515 ist ebenfalls sehr gering. Löst man $\ddot{y}(t) = -y(t)$ mit der einfachen Integrationsmethode von Euler, so ergibt sich bei T eine Abweichung von 0.03666314 % und eine mittleren Abweichung von 0.0291477436411. Zum Vergleich noch die Werte für den Fall, dass man mit Runge-Kutta vierter Ordnung löst: 0.00000078 % und Abweichung von 0.0000000004590. Die sanften Änderungen der Beschleunigung bei dem Schwingungsvorgang führen dazu, dass das Runge-Kutta-Verfahren sehr gute Ergebnisse liefert.

Auch hier können wir die Genauigkeit der impulsbasierten Methode erheblich verbessern, wenn wir zum Verfahren zweiter Ordnung übergehen. Bezüglich der %-Abweichung würde sich dann bei h = 0.01 ergeben:

0.0000001401 = (4/3.0)*(-0.0011459594)-(1/3.0)*(-0.0045842580),

also sogar ein leicht besserer Wert als für Runge-Kutta. Beim korrigierten Mittelwert ergäbe sich 0.00000000336 = (4/3.0)*(0.0000057404)-(1/3.0)*(0.0000229515). Ob sich diese pauschal hochgerechneten Ergebnisse auch bei der echten impulsbasierten Simulation zweiter Ordnung ergeben werden, konnte noch nicht festgestellt werden.

Soll die Federkraft zusätzlich gedämpft werden, so haben wir als gegen die Bewegung gerichtete Dämpfungskraft

$$K(t) = K_d \cdot \frac{d}{dt}\left|\Delta p(t)\right| \cdot \Delta_1 p(t)$$

einzubeziehen, also mit der Näherung $\Delta_1 p(0)$ für die relativ konstante Größe $\Delta_1 p(t)$

$$I \approx K_d \cdot \Delta_1 p(0) \cdot \int_{-h/2}^{+h/2} \frac{d}{dt}\left|\Delta p(t)\right| dt$$

$$I \approx K_d \cdot \Delta_1 p(0)\left(\left|\Delta p(h/2)\right| - \left|\Delta p(-h/2)\right|\right)$$

Auch hier könnte man wie bei der Federkraft das Impulsintegral genau bestimmen, müsste dann aber, weil mit dem Zeitparameter $t = +h/2$ ein Vorgriff auf noch nicht bekannte Größen erfolgt, in die iterative Korrektur einsteigen, was wir aus Aufwandsgründen (Implementierungsaufwand und Rechenzeit) vorerst vermeiden wollen. Daher schieben wir das Zeitfenster $[-h/2, +h/2]$, in dem integriert wird, etwas zurück auf $[-h, 0]$ und erhalten dann die Näherung

$$I \approx K_r\left(\left|\Delta p(0)\right| - \left|\Delta p(-h)\right|\right) \cdot \Delta_1 p(0).$$

Dieser Impuls ist nun einmalig anzuwenden auf $p_1(0)$ und $-I$ auf $p_2(0)$. „Einmalig“ bedeutet, dass dieser Gelenktyp (bei der hier gewählten approximativen Implementierung) in der iterativen Gelenkkorrektur für jedes Zeitintervall nur einmal einzubeziehen ist.

Speziell in der Robotik und Mechatronik sind auch Antriebe erforderlich, also bei Achsen Drehmomente und bei Schlittenführungen translatorische Kräfte. Drehmomente und translatorische Kräfte müssen, wie oben beim Feder-Dämpfungs-Element bereits demonstriert, stets durch Integration zu Impulsen aufintegriert werden. Es reicht hier aus, wenn wir beschreiben, wie Achsen angetrieben werden.

Eine *servoangetriebene Achse* ist durch folgende Parameter gekennzeichnet:

Eine Drehachse mit zwei Punkt-zu-Punkt Gelenken zwischen Körper 1 und Körper 2,

abseits der Achse je einen Messpunkt $Pm_1(t), Pm_2(t)$ in den beteiligten Körpern; bei der Winkelstellung $\varphi_{ist} = 0$ sollten diese beiden Messpunkte zusammenfallen. (Referenzpunkte, Messpunkte und ähnliche an Körpern fixierten Punkte können sich beliebig außerhalb der eigentlichen Körpermasse befinden. Sie sind fest mit dem Körper verbunden und machen alle Körperbewegungen exakt mit, siehe P-Schritt 3.)

Wichtige Parameter sind auch der bereits erwähnte Winkel $\varphi_{ist}(t)$ und evtl. $\dot{\varphi}_{ist}(t)$. Als äußere Parameter kann man z. B. einführen:

(1) $\varphi_{soll}(t)$, also der Winkel, der durch Drehmomentwirkung eingestellt werden soll,

(2) Tmax, das durch die Servomotoren maximal zur Verfügung stehende Drehmoment,

(3) $C_F =$ Koeffizient für geschwindigkeitsabhängige Bremsreibung, entweder konstant oder als Parameter veränderbar und dann als Bremse und Feststellbremse nutzbar.

Wie man diesen Ausführungen entnimmt, kann man zwar simple Standardantriebe modellieren, genauere Nachbildungen in der Simulation erfordern jedoch viele Varianten. Außerdem kommt jetzt auch massiv die Regelung und das dabei angewandte Verfahren ins Spiel.

Wir wollen hier nur insofern ins Detail gehen, als wir zeigen wollen, wie ein Drehimpuls auf die mit einer Achse verbundenen Körper einwirkt, siehe Abb. 10.1.

Das Impulspaar *I, -I* greift an den Punkten p_1 und p_2 an, die für die Drehung beide als sowohl in Körper 1 als auch in Körper 2 liegend aufgefasst werden. Es handelt sich dabei allerdings nur um temporär eingeführte Hilfspunkte.

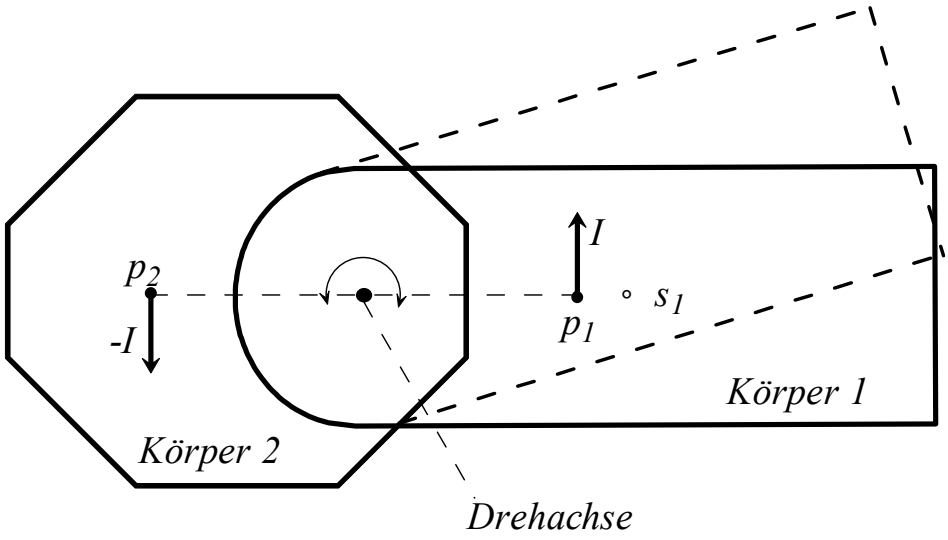

Abbildung 10.1: Der Drehimpuls I greift an der Drehachse an und wirkt auf Körper 1 ein, und zwar im Punkt p_1 mit I und im Punkt p_2 mit $-I$. Nicht eingezeichnet ist das genau entgegengesetzt wirkende Drehimpulspaar, welches auf Körper 2 einwirkt.

Auf den Schwerpunkt s_1 hat das an Körper 1 angreifende Impulspaar keine Wirkung, weil ihre vektorielle Summe gleich 0 ist. Der Drehimpuls ergibt sich zu

$$D = (p_1 - s_1) \times I + (p_2 - s_1) \times (-I) = (p_1 - p_2) \times I$$

Wenn wir $|p_1 - p_2| = 1$ unterstellen, also Hebelarmlänge 0.5 bei 2 x wirkender Impulskraft, so gilt $|D| = |I|$ und D hat die Richtung der Drehachse. Das daraus resultierende $\Delta\omega_1$ ist also schließlich

$$\Delta\omega_1 = \tilde{J}_1^{-1} \cdot Achse_1 \cdot |I|.$$

Hier ist $Achse_1$ der normierte Achsvektor mit dem bezüglich der Richtung von I korrekten Vorzeichen. Analog gilt für den zweiten Körper

$$\Delta\omega_2 = \tilde{J}_2^{-1} \cdot Achse_1 \cdot (-|I|).$$

denn die Drehimpulse müssen sich kompensieren.

Als Fazit können wir feststellen, dass das Ausüben eines Drehmoments an einer Achse auf entsprechend starke Drehimpulse zurückgeführt wird, die sehr einfach berechnet werden können und nur die Drehvektoren ω der beteiligten Körper verändern. Damit ist die Grundlage gelegt für die Konstruktion angetriebener Achsen.

<u>*Lineare Kraftwirkungen*</u> zwischen zwei Punkten p_1, p_2, die in den Körpern 1 und 2 fixiert sind, müssen ebenfalls auf Impulseinwirkungen zurückgeführt werden. Mit der beim Feder-Dämpfungselement benutzten Notation muss der Impuls durch Integration gemäß

$$I := \int_{-h/2}^{+h/2} \Delta_1 p(t) \cdot K(t)\, dt$$

ermittelt werden, wobei zu jedem Zeitpunkt $\Delta_1 p(t)$ die Kraftrichtung und $K(t)$ ihre skalare Stärke ist. Wenn $K(t)$ bekannt ist, lassen sich wie bereits beim Feder-Dämpfungselement geschildert sehr genaue oder approximative Werte für *I* bestimmen. Der Rest ist dann einfach: I greift an p_1 an, $-I$ an p_2.

11. Kollision und Reibung in der impulsbasierten Simulation

Die Integration korrekter Verfahren für Kollision und Reibung in die dynamische Simulation von MKS ist nicht einfach und erfordert normalerweise sehr komplexe Erweiterungen des Grundalgorithmus. Wir wollen hier vor allem die grundsätzliche Vorgehensweise erläutern, wie Kollision und Reibung in die impulsbasierte Simulation integriert werden können. Vollständige Implementierungen und Leistungstests konnten noch nicht durchgeführt werden, so dass die hier vorgestellten Theorien als vorläufig zu betrachten sind. In unseren existierenden Implementierungen verwenden wir derzeit nur die allgemein üblichen, einfach zu implementierenden approximativen Verfahren.

Das Thema Kollision und Reibung hat in den letzten Jahren verstärkt Beachtung gefunden, weil die Probleme nicht so gut verstanden werden, dass von allseits befriedigenden Verfahren vor allem auch für Implementierungen in VR-Systemen gesprochen werden könnte. Einen schönen Überblick über aktuelle Ansätze und Lösungen gibt Stewart [2000]. Auch das schon fast klassische Buch von Pfeiffer und Glocker [1996] ist hier zu nennen sowie die in Richtung VR orientierten Entwicklungen von Sauer und Schömer [1998].

Für uns ist vor allem die Methode von Mirtich und Canny [1995] relevant, weil sie die Bedeutung der impulsbasierten Vorgehensweise betonen. Sie bringen allerdings nur einzelne Starrkörper zur Kollision und simulieren gelenkgekoppelte Systeme mit konventionellen Methoden, während wir in der Lage sind, Kollisionen von komplexen gelenkgekoppelten MKS vollständig impulsbasiert zu simulieren. Ihr Ansatz, der vermutlich auf eine Arbeit von Keller [1986] zurückgeht, dient uns allerdings nicht direkt als Grundlage. Wir zeigen hier vielmehr, wie man mit konsequenter Ausnutzung der Vorteile der impulsbasierten Simulationsmethode auch Kollision und Reibung sehr befriedigend simulieren kann, wobei speziell die bereits früher ausführlich erläuterten Verfahren der iterativen Geschwindigkeits- und Gelenkkorrektur eine Schlüsselrolle spielen werden.

Wir halten uns konsequent an Poissons Stoßgesetz sowie an Coulombs Reibungsgesetz und außerdem an das Starrkörper-Prinzip: Unsere Körper haben keine Elastizitäten, Stoßverformungen und ähnliches, was verschiedentlich eingeführt wurde, um mit der Wirklichkeit besser überein zu stimmen. Aus Gründen der Systematik wollen wir also annehmen, dass sich Starrkörper an den Stoßstelle überhaupt nicht verformen.

Um nun Kollision und Coulombsche Reibung in unsere dynamische Simulation zu integrieren, sind folgende Vorgänge abzudecken:

- Kollision mit Abprall,
- Kollision mit Übergang in gleitende oder stockende Reibung,
- Gleitreibung, rollende oder stockende Reibung an Berührungspunkten,
- Ablösung eines Berührungspunktes durch Auseinanderbewegung der beteiligten Körper.

Um diese Vorgänge homogen und systematisch in das Verfahren der impulsbasierten Simulation zu integrieren, führen wir folgende zusätzliche Gelenktypen ein:

a) Stoßpunktepaar $G = (\#S, k_1, p_1, k_2, p_2)$

b) Berührungspunktepaar $G = (\#R, k_1, p_1, k_2, p_2)$

Diese Gelenktypen sind nicht permanent, sondern erscheinen nur temporär in der Gelenkliste *L*. Stoßpunktepaare werden genau nur so lange in die Gelenkliste L aufgenommen, bis das Kollisionsereignis abgearbeitet bzw. aufgelöst wird, das zwischen den Punkten p_1 von Körper k_1 und p_2 von Körper k_2 stattfindet, Berührungspunktepaare nur so lange, wie sich die beteiligten Körper berühren. Kollisionsereignisse werden während der Simulation von der mitlaufenden Kollisionserkennung entdeckt. Ein Berührungspunktepaar ist zwischen je zwei Punkten p_1 und p_2 verschiedener Körper eingerichtet, die sich berühren. Ein solches Gelenk hat zwei Aufgaben: Erstens muss sichergestellt werden, dass sich die beteiligten Körper nicht durchdringen und zweitens müssen Coulombsche Reibungskräfte in die Simulation eingebracht werden.

Berührungspunktepaare müssen auch in die Geschwindigkeitskorrektur einbezogen werden. Das geschieht dadurch, dass die Geschwindigkeitsnormale (bezogen auf die Kontaktebene) zu 0 gemacht wird, ohne die Tangentialgeschwindigkeit zu beachten. Reibung hat auf die Geschwindigkeitskorrektur keinen Einfluss, weil die Zeit dabei still steht bzw. weil sich bei der Geschwindigkeitskorrektur nichts bewegt.

„Time-Stepping"-Algorithmus

Mit dem Begriff des Time-Stepping bezeichnet man bei der kollisions- und reibungsbehafteten dynamischen Simulation die Vorgehensweise, den jeweils nächsten Zeitschritt an die markanten Ereignisse wie Stoß und Kontaktpunktablösung anzupassen. Stewart (2000) diskutiert u. a. das Time-Stepping im Überblick, die Dissertation von Tzitzouris (2001) beschreibt eine konkrete Implementierung. Die beschriebenen Algorithmen sind außerordentlich komplex und kompliziert, sie sind völlig ungeeignet etwa für Implementierungen in VR-Systemen. Denn neben dem Kollisionsproblem müssen auch häufig sogenannte nichtlineare Komplementaritätsprobleme gelöst werden, um die mechanischen Bedingungen bei Stoß und Reibung einzuhalten. Bei unserem vollständig impulsbasierten Ansatz entfallen die Komplementaritätsprobleme, weil wir die entsprechenden Probleme auf etwas andere Weise lösen, nämlich mit dem sehr mächtigen Algorithmus der iterativen Gelenkkorrektur.

Unser *Time-Stepping-Algorithmus* läuft wie folgt ab:

Während der dynamischen Simulation mit der Standard-Zeitschrittweite h läuft die Kollisionserkennung mit. Falls in einem LookAhead-Zeitfenster $t_0 \to t_0 + h$ Kollisionsereignisse erkannt werden und das zeitlich früheste zum Zeitpunkt $t_0 + h_1 < t_0 + h$ stattfindet, so wird dieses Zeitfenster auf die Zeitschrittweite h_1 verkürzt. Wie bereits festgestellt wurde, sind solche Verkürzungen und auch Verlängerungen von Zeitschrittfenstern bei uns problemlos möglich. In diesem verkürzten Zeitfenster finden nach Definition keine Kollisionen mehr statt, wir erhalten also zum Zeitpunkt $t_0 + h_1$, dem ersten Kollisionszeitpunkt, einen konsistenten Systemzustand, der durch Anwendung der routinemäßig durchgeführten Gelenkkorrekturen und der anschließenden Zeitfortschaltung erreicht wird.

Im Inneren von Zeitschritten werden also entweder Kollisionen entdeckt, was zur Verkürzung von Zeitschritten führt, oder der Zeitschritt wird ohne Kollisionen durchgeführt. Dabei

werden die aktuell in L befindlichen Gelenke abgearbeitet, also neben den Standard-Gelenktypen auch die aktuell in *L* befindlichen Berührungspunktepaare. Am Ende der iterativen Gelenkkorrekturen wird überprüft, ob sich unter den Berührungspunktepaaren welche befinden, bei denen eine Punktablösung, also ein Voneinanderentfernen der beteiligten Körper, erfolgt ist. Solche Gelenke werden in der Liste *L* gelöscht.

Die Kollisionsauflösung erfolgt also bei dieser Vorgehensweise stets genau zu dem Zeitpunkt, wo normalerweise die Geschwindigkeitskorrektur durchgeführt wird. Für sie muss wie oben beschrieben temporär ein Stoßpunktepaar $G = (\#S, k_1, p_1, k_2, p_2)$ eingerichtet und in die Gelenk-Liste *L* eingetragen werden. Sodann wird der Stoß abgearbeitet, Details dazu werden weiter unten erläutert. Falls sich dabei herausstellt, dass sich die Kontaktpunkte p_1 und p_2 nach dem Stoß nicht genügend voneinander entfernen, wird das Stoßpunktepaar in der Liste *L* in ein Berührungspunktepaar $G = (\#R, k_1, p_1, k_2, p_2)$ umgewandelt und verbleibt in *L*, während es ansonsten aus L entfernt wird.

Stoßereignisse mit Reibung

Als erstes betrachten wir den Stoßvorgang wie in Abb. 11.1 dargestellt. Zum Zeitpunkt t_1

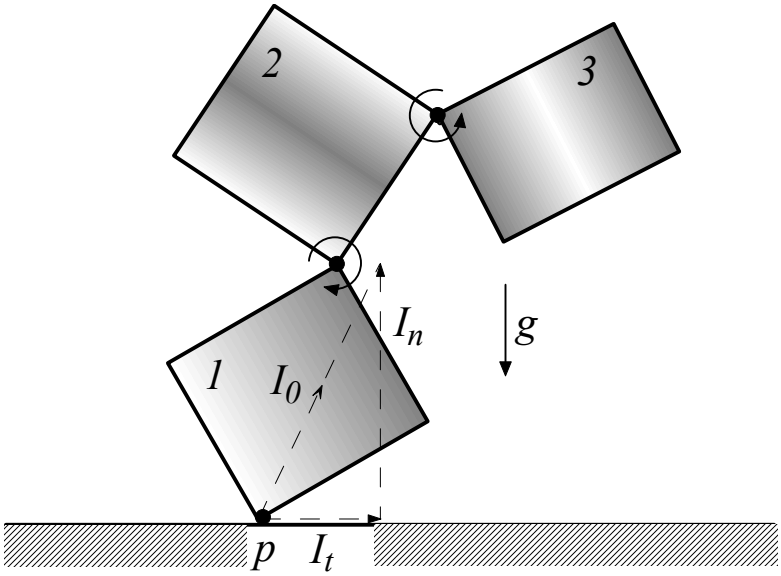

Abbildung 11.1: *Ein gelenkgekoppeltes System bestehend aus drei gelenkgekoppelten Starrkörpern fällt auf eine starre Bodenplatte.*

möge der Punkt p von Körper 1 die Bodenplatte berühren und damit ein Stoßereignis einleiten. Wie oben bereits beschrieben erkennen wir solche Stoßereignisse während der dynamischen Simulation, indem wir im LookAhead-Zeitfenster $t_0 \to t_0 + h$ mit den in der Literatur vielfach beschriebenen Methoden eine Kollisionserkennung durchführen und, falls eine Kollision entdeckt wird, die Schrittweite h dann so verkürzen, dass das MKS genau zum Zeitpunkt t_1 der ersten Kollision im Intervall $[t_0, t_0 + h]$ zum Halten kommt, also in einem Dmax-konsistenten Zustand. Dies ist also ein Zeitschrittverfahren, welches stets genau an die Kollisionsereignisse angepasste *h*-Werte benutzt.

Das Stoßereignis wird nun wie folgt abgehandelt: Wir führen als erstes die in Abschnitt 6 beschriebene Geschwindigkeitskorrektur durch und bringen p vorübergehend zur Ruhe, also $\dot{p}(t_0) = 0$, weil der Boden (Körper 0) als unbeweglich betrachtet wird[2]. Alle bei p angewandten Korrekturimpulse werden aufsummiert, dabei möge sich der Gesamtimpuls I_0 ergeben. Dies ist also genau derjenige Impuls, der Körper 1 an p bezüglich der Bodenfläche zur Ruhe bringt, wir nennen ihn daher auch Bremsimpuls. Man beachte, dass sämtliche durch Gelenke mit Körper 1 direkt oder indirekt verbundenen Teilkörper des MKS einen Einfluss auf I_0 ausüben.

I_0 wird nun in eine auf der Kontaktfläche senkrecht, also in Normalenrichtung, stehende Komponente I_n und eine tangentiale Komponente I_t zerlegt, siehe auch Abb. 11.1:

(10.1) $$I_0 = I_n + I_t .$$

Um so vorgehen zu können, muss eine Normalenrichtung berechenbar sein, was wir in dieser einführenden Arbeit der Einfachheit halber stets voraussetzen wollen.

Poissons Stoßgesetzt besagt nun:

Wenn der normal zur Oberfläche wirkende Impuls bis zum Verschwinden der Relativgeschwindigkeit in Normalenrichtung (bis zum Ende der Kompressionsphase) mit I_n bezeichnet wird, so berechnet man den nach der Kompressionsphase auszuübenden Abstoßungsimpuls I_{reflex} gemäß

(10.2) $$I_{reflex} = (1-e)I_n ,$$

wobei e eine von Materialeigenschaften abhängige Konstante $0 \le e \le 1$ ist.

Für den vollplastischen Stoß ohne jeden Abprall gilt also $e = 1$, was eine erhebliche Energieabsorption bewirken kann. Bei $e = 0$ liegt ein vollelastischer Stoß ohne jede Energieabsorption vor. Damit ist also geklärt, wie mit I_n zu verfahren ist.

Auch beim reinen Stoßereignis müssen wir die Coulombsche Reibung berücksichtigen, wenn korrekte Simulationsergebnisse erzielt werden sollen. Denn wenn wir die oben identifizierte tangentiale Komponente I_t wieder unverändert mit negativem Vorzeichen auf p einwirken lassen, um die Tangentialgeschwindigkeit wieder herzustellen, die unmittelbar vor dem Stoß herrschte, so wäre die Bodenfläche glatt wie Eis. Würden wir nur I_{reflex} anwenden, ohne I_t weiter zu beachten, so hätten wir die Relativgeschwindigkeit von p bezogen auf die Bodenfläche zum Verschwinden gebracht, was einem sehr stumpfen Boden entsprechen würde.

Bei Coulombs Reibungsgesetz sind zwei Fälle zu unterscheiden: gleitende („sliding“) und stockende („sticking“) Reibung. In Impulsform übersetzt ergibt sich: Wenn I_n und I_t so wie oben definiert sind, so liegt stockende Reibung vor, wenn

[2] Wenn beide kollidierende Körper beweglich sind, so wird die Relativgeschwindigkeit der Berührungspunkte zu Null gebracht.

(10.3) $$|I_t| \le \mu \cdot |I_n|$$

gilt. Hierbei ist μ der bekannte von Materialeigenschaften abhängige Reibungskoeffizient. In diesem Fall sind die Normalenkräfte so groß, dass die Tangentialkräfte nicht mehr ausreichen, ein Gleiten zu bewirken. Das bedeutet, dass die Tangentialbewegung gestoppt werden muss, d. h. I_t wird nicht mehr in den Bewegungszustand eingerechnet. Interessanterweise interessiert also nur der Winkel, den der Gesamt-Bremsimpuls I_0 mit der Tangentialebene bildet, nicht die absolute Stärke, um die Entscheidung zwischen stockender bzw. gleitender Reibung zu fällen.

Wenn keine stockende Reibung vorliegt, so ist zusätzlich zu $-I_t$ die der Tangentialgeschwindigkeit entgegen gesetzte Reibungskraft

(10.4) $$I_r = -\mu |I_n| v_{1t}$$

einzubeziehen, hier gleich in Impulsform und mit dem Einheitsvektor in Richtung der Tangentialgeschwindigkeit v_{1t}, welche uns aber nur approximativ bekannt ist. Bedingt durch Nutationseffekte und andere Einflüsse können wir nicht einfach $-I_t$ als Richtung von v_{1t} unterstellen, denn der durch die Reibung einwirkende Gegenimpuls I_r kann diese Richtung verändern. (Man beachte, dass der Index 1 bei v_{1t} hier und im Folgenden bedeutet, dass der Geschwindigkeitsvektor auf die Länge 1 normiert ist.)

Wir haben also die Gleichung (10.4) zu lösen, um die noch unbekannte tangentiale Abprallrichtung v_{1t} zu bestimmen. Das Problem ist deshalb nicht trivial, weil v_{1t} auf I_r einwirkt und dieses wieder auf v_{1t} und diese Wechselwirkung durch das dynamische Verhalten des gesamten MKS vermittelt wird.

Keller [1986] und später Mirtisch und Canny [1995] bewältigen diese Schwierigkeit durch das Lösen einer speziellen Differentialgleichung. Dabei wird allerdings nur die Kollision zwischen einzelnen, nicht gelenkgekoppelten Starrkörpern behandelt.

Wir können bei konsequenter Anwendung der impulsbasierten Methode wie folgt vorgehen: Wir wenden auf den Punkt *p* zunächst I_{reflex} und $-I_t$ an. Die resultierende Tangential-Richtung der Geschwindigkeit von *p* liefert eine erste Approximation für v_{1t}, die wir mit $v^{(1)}$ bezeichnen. Wenn μ gleich 0 ist, so sind wir fertig, die Bodenebene (Körper 0) ist sehr glatt. Andernfalls wenden wir zusätzlich den aus $v^{(1)}$ formal resultierenden Reibungsgegenimpuls an, also $I_r^{(1)} := -\mu |I_n| v^{(1)}$. Das liefert ein genaueres $v^{(2)}$. Damit berechnen wir $I_r^{(2)} := -\mu |I_n| v^{(2)}$, einen wieder genaueren Reibungsimpuls, und lassen den Impuls $I_r^{(2)} - I_r^{(1)}$ am Punkt p auf Körper 1 einwirken. Durch die Subtraktion von $I_r^{(1)}$ ist dieser völlig eliminiert. Wir erhalten ein noch etwas genaueres $v^{(3)}$ usw. usf. Dieses iterative Verfahren sollte sehr schnell konvergieren, weil $\left| I_r^{(i+1)} - I_r^{(i)} \right|$ schnell sehr klein werden

sollte. Letztendlich lösen wir hier das Problem, ein der gleitenden Coulomb-Reibung entsprechendes bremsendes $I_r^{(\infty)}$ zu finden, so dass

$$I_r^{(\infty)} = -\mu |I_n| v^{(\infty)} \tag{10.5}$$

und so, dass $I_r^{(\infty)}$ und $v^{(\infty)}$ genau die gleiche Richtung haben. Wenn wir bei der Iteration zur Bestimmung von $I_r^{(\infty)}$ stets die Geschwindigkeitskorrektur im ganzen MKS mitlaufen lassen, wobei allerdings die Kontaktstelle p nicht in die Geschwindigkeitskorrektur einbezogen wird, so erfordert dies nur geringen Korrekturaufwand, weil sich dabei das System in einem nahezu perfekt korrigierten Geschwindigkeitszustand befindet[3]. Im Ergebnis erhalten wir also ein Verfahren zur Auflösung von Kollisionen mit Reibung, das gegenüber Mirtich und Canny [1995] eine wesentliche Verallgemeinerung darstellt, weil das gesamte gelenkgekoppelte MKS auf den Stoßvorgang einwirkt.

Damit ist nun erläutert, wie ein Stoßereignis in der impulsbasierten dynamischen Simulation verarbeitet wird. Wir berechnen nach der ersten Geschwindigkeitskorrektur die Impulse I_{reflex} *und* $-I_t$, lassen sie einwirken, und bestimmen dann $I_r^{(\infty)}$, wobei die Einwirkung während der Iteration erfolgt. Das MKS bewegt sich dabei nicht und wird wieder in einen geschwindigkeitskorrigierten Zustand überführt.

Das Verfahren ist insbesondere auch geeignet, mehrere gleichzeitige Stoßereignisse zu simulieren, da die Geschwindigkeitskorrekturen stets über das gesamte MKS ausgedehnt werden müssen. Das bedeutet wiederum, dass man auch das gröbere und damit ungenauere Verfahren anwenden kann, alle in einem LookAhead-Zeitfenster $t_0 \to t_0 + h$ entdeckten Kollisionen simultan zum Zeitpunkt $t_0 + h$ aufzulösen.

Reibung zwischen Kontaktpunkten

Wir wollen nun erläutern, wie ein Berührungspunktepaar $G = (\#R, k_1, p_1, k_2, p_2)$ in der iterativen Gelenkkorrektur simuliert wird. Das Verfahren ist nicht mehr ganz so einfach wie bei den bisher besprochenen Gelenkkorrekturen. Auch hier wollen wir ein einsichtiges Beispiel diskutieren. Dazu legen wir einen Starrkörper auf die schiefe Ebene und nehmen an, dass dieser an 2 Auflagepunkten p und p_2 aufliegt, wobei wir nur die beim Punkt p anfallenden Korrekturaktivitäten zum Zeitpunkt t diskutieren, siehe Abb. 11.2.

Im ersten Durchlauf der iterativen Gelenkkorrektur gehen wir wie folgt vor:

$$p(t+h) = LookAhead(1, p(t), h);$$

Wenn $p(t+h)$ außerhalb des Körpers 0 zu liegen kommt, ist nichts zu tun, auch die Reibungskräfte verschwinden. Liegt $p(t+h)$ jedoch im Inneren des Körpers 0, so berechnen

[3] Eine mögliche Komplikation, dass sich nämlich der Geschwindigkeitsvektor von *p* bei der Geschwindigkeitskorrektur in Richtung der Bodenplatte drehen könnte, wollen wir hier nur erwähnen, aber nicht diskutieren.

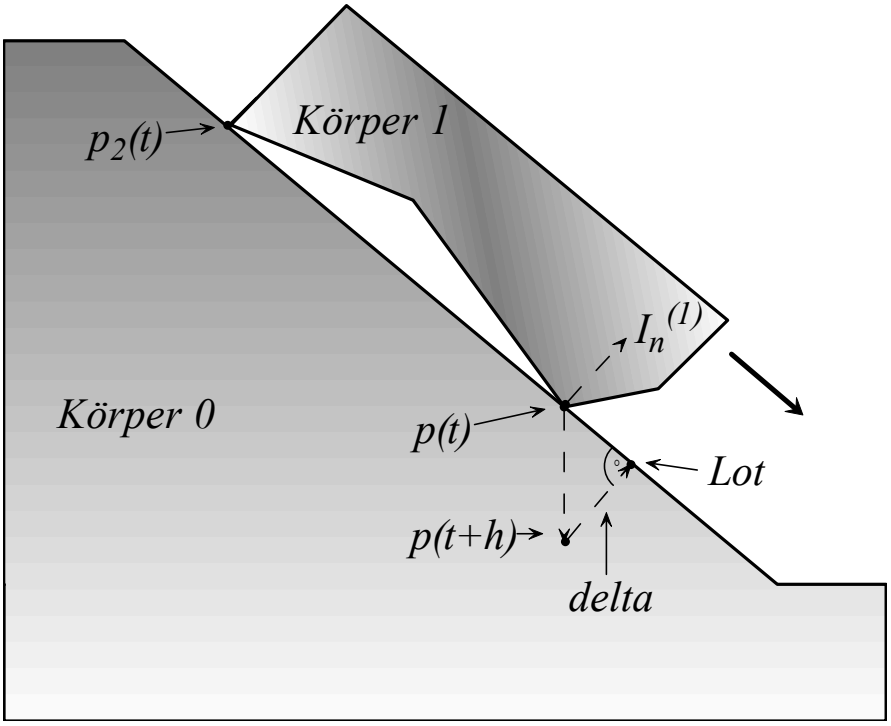

Abbildung 11.2: *Gleitender bzw. ruhender Körper auf der schiefen Ebene*

wir einen Korrekturimpuls, der das Eindringen verhindert:

$Lot =$ Lotpunkt von $p(t+h)$ bezüglich der Oberfläche von Körper 0;

$delta := Lot - p(t+h)$;

$\Delta\dot{p} := delta / h$

$I_n^{(1)} := M^{-1} \cdot \Delta\dot{p}$ (siehe P-Schritt 2 in Abschnitt 5)

Dieser erste Korrekturimpuls wirkt auf $p(t)$ ein und sorgt dafür, dass $p(t+h)$ nicht mehr im Inneren des Körpers 0, sondern auf dessen Oberfläche zu liegen kommen sollte. Wir merken uns

$$I_{nsum} := I_n^{(1)}$$

und bestimmen den aktuellen, durch Coulomb-Reibung zur Verfügung stehenden Reibungsimpuls

$$I_{r\max}^{(1)} := -\mu \cdot \left| I_{nsum} \right| \cdot v_{1t}^{(1)}$$

Hierbei ist $v_{1t}^{(1)}$ der auf Länge 1 normierte Relativ-Geschwindigkeitsvektor von $p(t)$ nach Anwendung des Impulses $I_n^{(1)}$. Hier und im Folgenden muss $v_{1t}^{(i)}$ wie folgt berechnet werden: $v_t^{(i)} = (p(t+h) - p(t))/h$, sodann mit $v_{1t}^{(i)} = v_t^{(i)} / \left| v_t^{(i)} \right|$ normiert werden.

In unserem Beispiel ist der Körper 0 als raumfest angenommen, so dass Relativgeschwindigkeit und Absolutgeschwindigkeit übereinstimmen.

Nun wird schließlich auch der durch die Reibungskraft beigesteuerte Impuls $I_{r\max}^{(1)}$ auf $p(t)$ angewandt, allerdings nur so stark, dass $p(t)$ nur abgebremst wird bis maximal zur Relativgeschwindigkeit 0, also bis zum Ruhezustand. Den dabei angewandten Bremsimpuls $I_{rist}^{(1)} = \beta \cdot I_{r\max}^{(1)}$, wobei $0 \le \beta \le 1$ den anwendbaren Anteil von $I_{r\max}^{(1)}$ skaliert, merken wir uns für dieses Gelenk:

$$I_{ralt} := I_{rist}^{(1)}.$$

Bei den weiteren Korrekturdurchgängen, z. B. beim i-ten Durchgang, wird so vorgegangen:

Bestimme analog wie oben $I_n^{(i)}$, wende ihn auf $p(t)$ an und summiere auf:

$I_{nsum} := I_{nsum} + I_n^{(i)}$, sodann wird die aktuelle Richtung des normierten Geschwindigkeitsvektors $v_{1t}^{(i)}$ bestimmt, dann $I_{r\max}^{(i)} := -\mu \cdot \left| I_{nsum} \right| \cdot v_{1t}^{(i)}$, der durch die Reibung maximal zur Verfügung stehende Impuls entgegen der Bewegungsrichtung.

Anschließend wird bestimmt $I_{rist}^{(i)} = \beta \cdot I_{r\max}^{(i)} - I_{ralt}$, wobei $0 \le \beta \le 1$ wieder dafür sorgt, dass *p* durch einen zu großen Reibungsimpuls höchstens zur Ruhe kommt, aber nicht rückwärts bewegt wird. Hier entscheidet sich also, ob stockende oder gleitende Reibung vorliegt. Schließlich wird $I_{rist}^{(i)}$ auf *p* angewandt und bis zum nächsten Korrekturdurchgang durch dieses Gelenk mit $I_{ralt} := I_{rist}^{(i)}$ gespeichert.

Im Ergebnis wird im Rahmen der iterativen Korrektur durch dieses Vorgehen erreicht, dass der Kontaktpunkt mit normal zur Oberfläche wirkenden Gegenimpulsen daran gehindert wird, in das Innere des Körpers 0 einzudringen. Die durch diese Impulse bewirkten Druckkräfte werden in I_{nsum} aufsummiert und direkt zur Bestimmung der jeweils maximal verfügbaren Reibungskraft benutzt. $I_n^{(i)}$ sollte für *i = 2, 3, 4,...* schnell kleine Werte annehmen und die Richtung von $I_{r\max}^{(i)}$ sollte sich schnell stabilisieren. Die normale Gelenkkorrektur enthält also hier den iterativen Vorgang, durch den letztendlich erreicht wird, dass der tatsächlich angewandte Reibungsbremsimpuls genau entgegengesetzt der Bewegungsrichtung wirkt, denn es gilt nach Abschluss der Gelenkkorrekturen, also im eingeschwungenen Zustand des iterativen Verfahrens der Gelenkkorrekturen:

$$I_{rist}^{(\infty)} = \beta \cdot I_{r\max}^{(\infty)} = -\beta \cdot \mu \cdot \left| I_{nsum}^{(\infty)} \right| \cdot v_{1t}^{(\infty)}.$$

Bei der geschilderten Vorgehensweise werden alle Effekte simuliert, die man bei einem auf der schiefen Ebene abrutschenden Körper beobachten kann: Übergänge zwischen stockender und gleitender Reibung. Außerdem muss der Reibungskoeffizient μ nicht konstant sein, wir können ihn mühelos von der Gleitgeschwindigkeit abhängig machen, indem stets

ein solches μ in die Formel für $I_{r\max}^{(i)}$ eingesetzt wird, das der aktuellen Tangentialgeschwindigkeit $v_t^{(i)}$ entspricht.

Obwohl die korrekte Implementierung des temporären Gelenktyps „Berührungspunktepaar“ oben nur grob skizziert wurde und eine Reihe von Feinheiten nicht angesprochen wurden, die in die Implementierung zusätzlich integriert werden müssen, wobei vor allem gekrümmte Kontaktflächen und Rollkontakt zu nennen sind, sollte klar geworden sein, dass Kollision und Reibung mit hoher Genauigkeit in das impulsbasierte Verfahren integriert werden können.

12. Schlussanmerkungen

Das hier vorgestellte Simulationsverfahren ist noch sehr jung, denn es wurde erst ab Januar 2003 entwickelt, als Verallgemeinerung der in Abschnitt 2 kurz vorgestellten impulsbasierten Simulation von Massenpunktsystemen. Obwohl eine umfassende Evaluation noch aussteht, sind die bisher bekannten Fakten jedoch sehr ermutigend. Die hervorstechendsten Eigenschaften sind die Einfachheit des Verfahrens sowie die hohe Simulationsgeschwindigkeit, letztere wird vor allem für Anwendungen in der Virtual Reality gefordert. Auch die natürliche Art, in der Kollisionsauflösung und Coulombsche Reibung integrierbar sind, ist ein großer Vorteil, insbesondere auch für den Einsatz in VR-Systemen. Auf den ersten Blick könnte man vermuten, dass das Ersetzen kontinuierlicher Kräfte durch Impulse nur zu ungenauen Simulationsergebnissen führen kann. Umso größer war unsere Überraschung, als wir durch Vergleichstests mit kommerziellen Dynamik-Systemen feststellen konnten, dass die impulsbasierte Simulation mit diesen voll konkurrenzfähig ist. Schließlich gelang es über verhältnismäßig einfache Fehlerabschätzungen (siehe Abschnitt 8), impulsbasierte Verfahren höherer Ordnung zu finden, die alle Restzweifel an evtl. mangelnder Genauigkeit ausgeräumt haben.

Danksagung

Für seinen fachkundigen Rat und seine kritische Anteilnahme bin ich Herrn Kollegen Wittenburg zu großem Dank verpflichtet. Auch Herrn Bender und Herrn Stelzner möchte ich an dieser Stelle für manche fruchtbare Diskussion und Zuarbeit danken. Wie immer haben mich Frau Szameitat und Herr Lindner bei der Erstellung der Reinschrift kompetent unterstützt.

13. Literatur

Baraff D. (1996)
Linear-time dynamics using lagrange multipliers,
Computer Graphics (SIGGRAPH 96 Proceedings), pages 137-146.

Bender J., M. Baas, A. Schmitt (2003)
Ein neues Verfahren für die mechanische Simulation in VR-Systemen und in der Robotik,
Proc. 17. Symposium Simulationstechnik ASIM 2003, Magdeburg, 111-116.

Finkenzeller, D., M. Baas, S. Thüring, S. Yigit, A. Schmitt (2003)
VISUM: A VR System for the Interactive and Dynamics Simulation of Mechatronic Systems,
Proc. Virtual Concepts 2003, Biarritz, 155-162.

Keller, J. B. (1986)
Impact with Friction,
J. of Applied Mechanics, 53, March 1986.

Mirtich, B. (1996)
Impulse-based Dynamic Simulation of Rigid Body Systems,
Dissertation, University of California at Berkeley.

Mirtich, B., J. Canny (1995)
Impulse-based Simulation of Rigid Bodies,
Symposion on Interactive 3D Graphics, Monterey, Cal.

Pfeiffer, F., Ch. Glocker (1996)
Multibody Dynamics with Unilateral Contacts,
Wiley Series in Nonlinear Science, New York.

Sauer J., E. Schömer (1998)
A constraint-based approach to rigid body dynamics for virtual reality applications,
ACM Symposium on Virtual Reality Software and Technology, VRST'98, S. 153-161.

Schelling, F. (2002)
Anwendung von Massenpunktverfahren für die Echtzeitsimulation eines humanoiden Roboterarms,
Diplomarbeit, Institut für Betriebs- und Dialogsysteme, Universität Karlsruhe.

Schmitt, A., S. Thüring (2000)
Ein vereinfachtes numerisches Verfahren für die mechanische Simulation in Virtual-Reality-Systemen,
Interner Bericht 2000-26, Fakultät für Informatik, Universität Karlsruhe.
(http://www.ubka.uni-karlsruhe.de/vvv/ira/2000/26/26.pdf)

Stewart, D. E. (2000)
Rigid-Body Dynamics with Friction and Impact,
SIAM Review, Vol. 42, No. 1, pp 3-39.

Tzitzouris, James A. (2001)
Numerical Resolution of Frictional Multi-Rigid-Body Systems via Fully Implicit Time-Stepping and Nonlinear Complementarity
Dissertation, The John Hopkins University, Baltimore.

J. Wittenburg (1977)
Dynamics of Systems of Rigid Bodies,
B. G. Teubner, Stuttgart 1977.

.-.-.-.