# Biomechanische Simulation von sportlichen Bewegungsabläufen mit Mehrkörpersystemen

Diplomarbeit
von
Florian Fritzsche

Hauptberichter: Prof. Dr. Hanns Ruder

Mitberichter: PD Dr. Stefan Luding

Institut für Astronomie und Astrophysik
der Eberhard-Karls-Universität Tübingen
Abteilung Theoretische Astrophysik
Arbeitsgruppe Biomechanik

März 2001

# Inhaltsverzeichnis

# Kapitel 1

# Einleitung

Bei der Analyse von sportlichen Bewegungsabläufen auf der Grundlage von Videobildanalysen werden innere Bewegungsgrößen, wie Gelenkmomente oder Gelenkbelastungen oder abstrakte Größen wie Drehimpulse berechnet, die für das Verstehen und die Optimierung einer Bewegung von Interesse sind. Vorteil der indirekten Messung mit Videokameras ist die Tatsache, dass der Sportler während seiner Bewegungsausführung nicht eingeschränkt ist.

In der biomechanischen Simulation kann der Mensch als ein Mehrkörpersystem (MKS) betrachtet werden, mehrere Starrkörper sind mit Gelenken untereinander verbunden. Das mathematische Modell beschreibt die Bewegungsgleichungen des MKS.

Das Ziel der vorliegeden Arbeit war, die Bewegung eines Flick-Flacks zu analysieren, um die dabei auftretenden Gelenkmomente zu bestimmen. Die aus einer Videobildanalyse stammenden kinematischen Größen wurden als Regelgrößen verwendet. Die Motivation für einen Regelansatz bestand darin, die Messwerte zu glätten und dem Modell, das die Realität nur unvollständig wiedergibt, die aufgenommene Flick-Flack Bewegung aufzuprägen.

Neben der Videobildanalyse werden in der Biomechanik weitere Messverfahren verwendete. Diese sind in Kapitel 2 zusammengestellt.

Es gibt zwei grundsätzliche Methoden der biomechanischen Analysen. Auf der einen Seite die Direkte Dynamik, auch als Vorwärtssimulation bezeichnet, bei der nach Vorgabe der ursächlichen Kräfte und Momente die Bewegungen berechnet werden. Komplementär dazu ist die Inverse Dynamik, bei der aus den kinematischen Bewegungsgrößen die Kräfte und Momente berechnet werden. Der in dieser Arbeit verwendete Ansatz der „Pseudoinversen Dynamik mit Regelalgorithmus“ liegt zwischen diesen beiden Methoden. Wie bei der Inversen Dynamik wird von der Bewegung ausgegangen, doch wirken die interessierenden Kräfte und Momente des Regelalgorithmus wieder auf die Mechanik zurück, wie bei der Vorwärtssimulation.

Die Analysemethoden werden in Kapitel 3 genauer erklärt, die Grundlagen zur verwendeten Regelung enthält Kapitel 4.

In Kapitel 5 werden die Arbeitsschritte im Einzelnen beschrieben. Kapitel 5.1 befasst sich mit dem Aufbau und der Durchführung der Videobildanalyse, in Kapitel 5.2 wird die Erstellung eines achtgliedrigen, ebenen Modells mit Hilfe des kommerziellen Software-Packets Simpack behandelt und der verwendete Regelalgorithmus, der an einen PID-Regler angelehnt ist und mit nichtlinearen Funktionen versehen ist, wird in Kapitel 5.3 beschrieben.

Im Kapitel 5.4 werden die Ergebnisse der Simulationen zusammengestellt und die Arbeit schließt mit dem Ausblick.

# Kapitel 2

# Biomechanische Messverfahren

In der Biomechanik gibt es verschiedene Messverfahren, die sich nach den gemessenen Größen einteilen lassen [1]. Siehe Tabelle 2.1:

| **Messverfahren** | **Größen** | **Einzelmessverfahren** |
|---|---|---|
| Dynamometrie | Kraft | Kraftmessplatten<br>Dynamometer |
| Kinemetrie | kinematische Größen:<br><br>Winkel<br>Körperkoordinaten<br>Beschleunigungen | Goniometer<br>Videobildanalyse<br>Beschleunigungssensoren |
| Anthropometrie | Geometrie:<br>Lage des Körperschwerpunkts<br>Massengeometrie:<br>Trägheitsmomente | mechanische Verfahren<br>statistische Modelle |
| Elektromyographie | Muskelpotentiale | Spannungsmessung<br>über Hautelektroden |

Tabelle 2.1: Messverfahren in der Biomechanik

Die Reaktionskraft von Geräten, beispielsweise die Kraft der Reckstange auf einen Turner, kann mit Hilfe von Kraftaufnehmern, bestehend aus Dehnungsmessstreifen, die an den Geräten angebracht sind, gemessen werden. Beispiele sind der Startblockdynamometer und der Reckdynamometer.

Winkel lassen sich mit Goniometern bestimmen. Ein Prinzip beruht darauf, dass zwei Hebelarme über eine Achse verbunden sind, an der sich ein Potentiometer befindet. Für die Winkelstellung des Goniometers wird die abgegriffene Spannung als Maß verwendet. Befestigt man die Hebelarme an zwei gelenkig verbundenen Körpersegmenten, so lassen sich einachsige Gelenkbewegungen recht genau vermessen. Die Goniometerachse soll dabei möglichst gut mit der Gelenkachse übereinstimmen.

Ein weiteres Messverfahren ist die Elektromyographie (EMG). Die elektrischen Impulse der Nervenfasern, welche die motorischen Endplatten der Muskelfasern innervieren, werden über Hautelektroden registriert. Es liegt somit eine grobe Information über die Zahl der rekrutierten Muskelfasern vor. Damit erhält man Anhaltspunkte für die Stimulation eines Muskels, es lassen sich also Aussagen über die Beteiligung oder Nichtbeteiligung von Muskeln bei einer Bewegung treffen und Erkenntnisse über die Koordination des Muskelspiels gewinnen. Die Bestimmung der Muskelkraft ist nicht möglich, da keine Korrelation zwischen EMG-Signal und Muskelaktivität besteht.

## 2.1 Kraftmessplatte

Kraftmessplatten werden in der Orthopädie bei der Ganganalyse und in der Biomechanik zur Messung von Bodenreaktionskräften eingesetzt. Auf dem piezoelektrischen Effekt basierende Kraftmessplatten bestehen aus einer rechteckigen Deckplatte, die an ihren vier Ecken unterstützt ist. Dort sind triachsiale Kraftsensoren angebracht. Aus den Sensorwerten lassen sich die resultierende Kraft, der Kraftangriffspunkt und das freie Moment durch diesen Punkt um die vertikale Achse berechnen [13]. Die Eigenfrequenz der Deckplatte begrenzt den Frequenzbereich der Messung. Verzerrungsfreie Ergebnisse lassen sich noch für eine Samplefrequenz von 300 Hz erreichen, was für sportliche Bewegungen, abgesehen von starken stoßartigen Belastungen, ausreicht. Der Fehler liegt bei diesem Verfahren bei einem Prozent. Da die Person bei der Untersuchung die Platte treffen muss, kann das bewusste Zielen den Bewegungsablauf behindern.

## 2.2 Beschleunigungsmessung

Die Beschleunigungsmessung beruht auf dem Prinzip, dass mittels einer kleinen seismischen Masse bei der Beschleunigung eine Feder verformt wird. Die Verformung wird von einem Dehnungsmessstreifen oder einem Piezokristall in ein analoges Signal umgewandelt. Solche Geräte heißen auch Beschleunigungssensoren. Es gibt ein- und dreiachsiale Geräte mit unterschiedlichen Messbereichen im Handel. Die kleinsten wiegen unter 10 g und haben ein Volumen von unter 1 $cm^3$. Der Vorteil der Beschleunigungssensoren gegenüber Verfahren, welche die Orte messen, liegt darin, dass man Größen erhält, die den zweimal differenzierten Koordinaten entsprechen, welche für die Bewegungsanalyse benötigt werden (siehe Kapitel 3). Problematisch ist allerdings die Befestigung der Sensoren auf der Haut. Da die Haut nicht starr mit dem Skelettsystem verbunden ist, kommt es zu Amplitudenänderungen und Phasenverschiebungen, bei harten Stößen treten zusätzliche Schwingungen auf. Die gemessenen Beschleunigungen beziehen sich auf ein körperfestes System und müssen alle in ein ruhendes Koordinatensystem transformiert werden. Da zusätzlich immer die Erdbeschleunigung wirkt, hängt der Messwert von der Orientierung zur Richtung der Gewichtskraft ab. Es ist deshalb erforderlich, die räumliche Orientierung der Messelemente zusätzlich zu kennen.

## 2.3 Videobildanalyse

Das Ziel der Videobildanalyse eines Bewegungsablaufs ist die räumliche Bestimmung definierter Bezugspunkte auf dem Körper, die durch Marker gekennzeichnet werden. Sie können farbig oder reflektierend sein oder ein Infrarot-/Ultraschallsignal emittieren. Da sie auf der Haut angebracht werden, tritt die im Abschnitt 2.2 beschriebene Befestigungsproblematik auf: Die Marker schwingen bei Stößen.

Die Aufnahmen werden üblicherweise mit Digitalkameras gemacht. Durch Abbildungsverfahren, zu deren Kalibrierung Eichobjekte aufgenommen werden müssen, kann die Bearbeitung am Bildschirm erfolgen, wofür Software zur Verfügung steht. Nach Auswertung der vorgenommenen Messung liegt dann der zeitliche Ortsverlauf der Marker in tabellarischer Form vor.

Diese Methode hat den großen Vorteil, dass der Sportler in seiner Aktion nicht behindert wird. Fehler entstehen, wenn die Abbildungsvoraussetzungen nicht erfüllt sind, z.B. durch eine geringe Abbildungsqualität der Objektive am Bildrand, die von Linsenverzerrungen herrühren. So dürfen nach der Kalibrierung keine Veränderungen an der Kamera vorgenommen werden, wie Brennweiteveränderungen oder Verschiebungen des Stativs. Die Objektpunkte werden mit der Genauigkeit bestimmt, mit der sie auf dem Videobild identifiziert werden können. Hohe Genauigkeit erhält man durch die Verwendung möglichst kleiner Objektpunkte und eines möglichst großen Abbildungsmaßstabs. Die Position einer Glühlampen im Abstand von 10 m kann auf 1-2 cm ermittelt werden [8].

Schon bei der Vorbereitung der Messung ist darauf zu achten wieviele Freiheitsgrade das geplante Modell besitzen und über welche Bewegungsmöglichkeiten es verfügen soll.

Erst dann können die Marker so gesetzt werden, dass alle für die Inverse Dynamik (siehe Kapitel 3) benötigten Daten bestimmt werden können. Außerdem muss man die Position der Marker auf den Körpersegmenten kennen, um am Modell eine gewissenhafte Markerplatzierung vornehmen zu können.

## 2.4 Anthropometrie

In der Anthropometrie werden die Geometrie und Massengeometrie des Menschen bestimmt. Dazu können mechanische Verfahren verwendet werden, wie Längen- oder Massenmessung die hauptsächlich an Leichen durchgeführt werden. Auf dieser Basis können dann statistische Modelle erstellt werden. Am Institut für Astronomie und Astrophysik in Tübingen wurde das Programm *„calcman“* entwickelt, das Regressionsgleichungen aus „NASA-Anthropometrie-Daten“ berechnet. Somit erhält man nach der Angabe des Geschlechts, der Größe und des Gewichts des Probanden die für die Modellierung notwendigen anthropometrischen Merkmale, die in guter Näherung mit den tatsächlichen Maßen übereinstimmen. Dazu gehören zum Beispiel die Lagen der Schwerpunkte der Körpersegmente, die Abstände zwischen den Schwerpunkten zu den Gelenken, die Trägheitstensoren oder die Masse der Segmente.

# Kapitel 3

# Biomechanische Analysemethoden

Die beiden Methoden der Direkten Dynamik und der Inversen Dynamik sollen nun genauer vorgestellt werden. In der Gegenüberstellung sollen das Vorgehen, die Verwendung, sowie die Probleme verdeutlicht werden. In Abbildung 3.1 werden ausgehend von den Bewegungsgleichungen, hier formal dargestellt, die entgegengesetzten Richtungen der beiden Methoden deutlich. Nach einem groben Überblick werden in den Kapitel 3.1 und 3.2 die Methoden näher beschrieben.

Direkte Dynamik

$$F = m \cdot a$$

Inverse Dynamik

Abbildung 3.1: Formale Darsellung der Bewegungsgleichungen mit dem Weg der beiden Methoden der Direkten und Inversen Dynamik.

**Direkte Dynamik:** Die Bewegungsgleichungen des verwendeten Modells werden nach Eingabe der inneren Kräfte und Momente, und nach Vorgabe der aus Messungen gewonnenen Anfangs- und Randbedingungen gelöst. Als Ergebnis erhält man die Beschleunigungen. Durch Integration ergibt sich dann die Bewegung des Modells.

Die Momente können parametrisiert mit dem Zustandsvektor, also abhängig von Gelenkwinkel und Gelenkgeschwindigkeit, vorgegeben werden oder quasi „von Hand“ durch eine zeitliche Funktion eingegeben werden. Als Randbedingungen gehen in das Modell Gelenkwinkelanschläge und die Bodenreaktionskraft ein.

**Inverse Dynamik:** Dies ist eine Methode zur Bestimmung der Momente und Kräfte. Dazu zählen die Gelenkmomente, Muskelkräfte oder die Belastungen des Stützapparates. Im Gegensatz zum direkten Vorgehen werden die kinematischen Daten aus einer Messung als Eingabe verwendet. Beziehen sich die Daten auf den zeitlichen Verlauf von Markerpunkten auf dem Körper, wie z.B. bei einer Videobildanalyse, müssen die Koordinaten noch zweimal numerisch abgeleitet werden. Die damit zusammenhängende Problematik wird in Abschnitt 3.1 beschrieben.

Der Übergang zwischen beiden Methoden ist fließend. Zu keiner der beiden Methoden lässt sich der Ansatz der pseudoinversen Dynamik mit Regelalgorithmus ganz zuordnen. Er liegt gewissermaßen dazwischen. Wie bei der inversen Vorgehensweise liegt der Ausgangspunkt bei den gemessenen Bewegungsgrößen. Diese dienen als Regelvorgabe. Der Regelalgorithmus liefert durch Vergleich mit den aktuellen kinematischen Größen Kräfte oder Momente. Diese dynamischen Größen wirken dann durch die Mechanik (Bewegungsgleichungen) wieder auf die Bewegungen ein. Es tritt so eine Rückkopplung auf, die für eine Regelung charakteristisch ist (siehe auch Kapitel 4.1).

## 3.1 Inverse Dynamik

In Abbildung 3.2 ist das Vorgehen bei der inversen Dynamik als Prozess dargestellt. Am Anfang stehen die kinematischen Messdaten und das MKS-Modell. Sowohl die Messdaten als auch das Modell sind fehlerbehaftet.

Die Messfehler entstehen durch die bei der Videobildanalyse anfallenden verrauschten und diskreten kinematischen Daten der Marker. Diese müssen als Eingabe für das inversdynamische Vorgehen zweimal numerisch abgeleitet werden. Dadurch werden die hohen Frequenzen verstärkt, d.h. die numerisch ermittelten Geschwindigkeiten und Beschleunigungen sind sehr verrauscht und besitzen eine hohe Rauschamplitude. Deshalb müssen die Kurven geglättet werden, was zu Informationsverlusten führt. Eine anschauliche Erklärung und einige Glättungs- und Filterverfahren sind bei Krebs [9] zu finden.

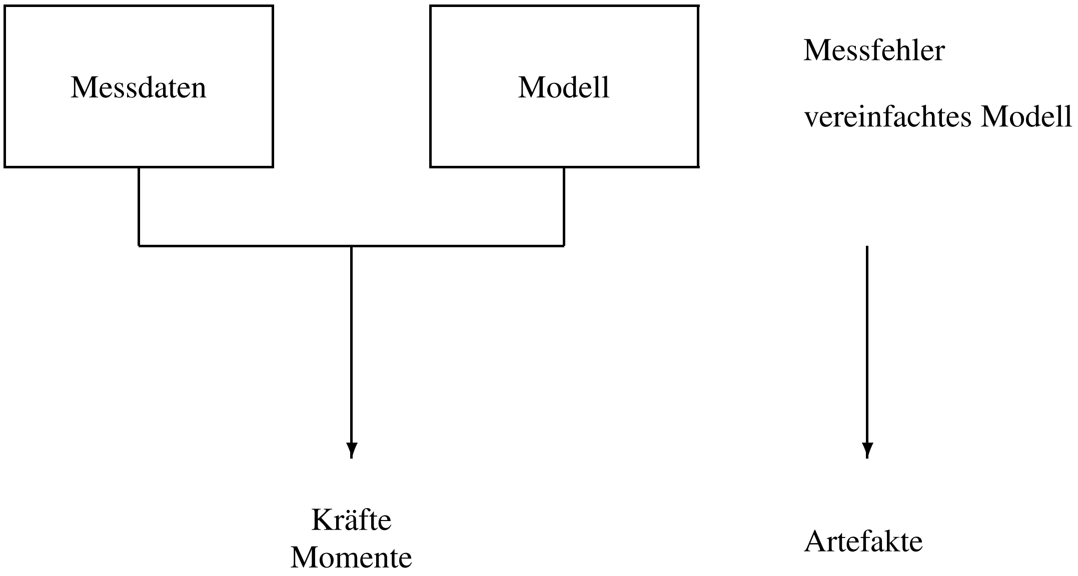


Abbildung 3.2: Inverse Dynamik: Ausgehend von gemessenen kinematischen Größen werden mit Hilfe eines Modells die Kräfte und Momente berechnet. Aufgrund von Messungenauigkeiten und einem Modell, das die Realität nicht genau abbildet, ergeben sich Artefakte.

Ein Modell bildet nur einzelne Merkmale der Realität ab. Es entspricht nicht der Wirklichkeit. Als Beispiel kann das Schultergelenk dienen. Es besitzt außer den rotatorischen Freiheitsgraden noch translatorische Bewegungsfreiheiten [17]. Ein ebenes Menschmodell, welches das Schultergelenk durch ein Scharniergelenk beschreibt, kann eine Bewegung der Schulter nach vorne, in der horizontalen Ebene, nicht durchführen. Stattdessen wird sich der gesamte Oberkörper nach vorne neigen. Daher treten bei der Simulation Artefakte auf.

Diese grundsätzlich auftretenden Probleme führen dazu, dass die Koordination von gemessenen äußeren Kräften und der Modellbewegung nicht mehr stimmt. Bei einer vorwärtsdynamischen Simulation mit den Kräften aus der inversen Berechnung weichen die Trajektorien der Simulation immer mehr von den gemessenen ab. Das Modell von Böhm [2] konnte eine Riesenfelge nicht mehr reproduzieren. Die gewonnenen inneren Kräfte sind als Eingabe für die direkte Dynamik des selben Modells nicht sinnvoll. Sie sind aber ein Anhaltspunkt für die tatsächlich bei der Bewegung auftretenden Kräfte.

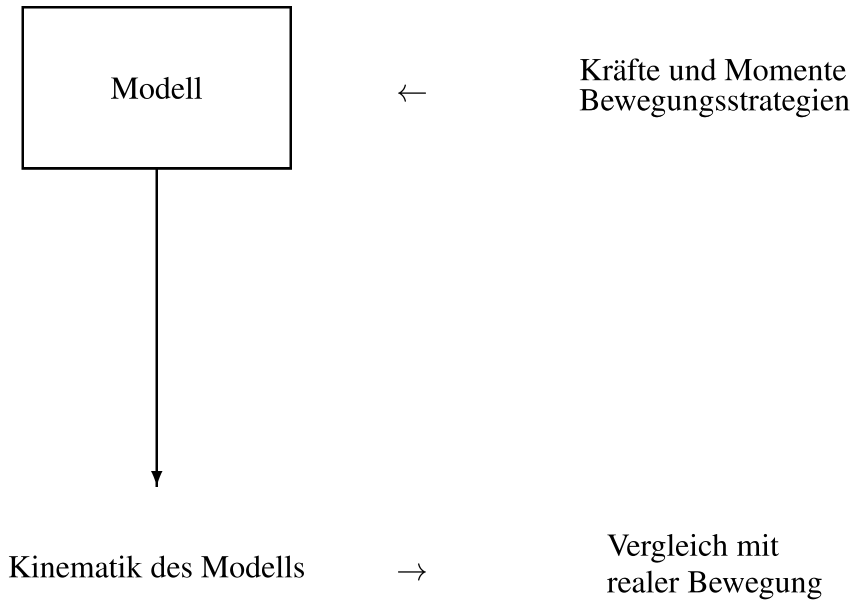


Abbildung 3.3: Direkte Dynamik: In das Modell gehen Kräfte und Momente, sowie die Strategien zur Bewegungssteuerung ein. Damit wird die Bewegung des Modells berechnet. Es treten keine Artefakte auf, da die Kräfte und Momente zum Modell passen. Das Modell wird durch den Vergleich mit der realen Bewegung validiert.

## 3.2 Direkte Dynamik

Bei der Direkten Dynamik gehen Kräfte und Momente in das Modell ein. Um ein definiertes Bewegungsziel zu erreichen, können übergeordnete Strategien verwendet werden. So existieren bei „Gangsimulationenen“ Strategien zur Beinplatzierung oder zum Stolpern.

Das aus der Vorwärtssimulation gewonnene Ergebnis (Kinematik des Modells) wird der realen Bewegung anpasst. Durch diesen Vergleich lässt sich das Modell validieren und durch die ständige gezielte Verbesserung des Modells sind ursächliche Zusammenhänge bei der Bewegungsentstehung zu verstehen. Günther [3] vergleicht diese Vorgehensweise mit der wissenschaftlichen Methodik in der Physik, wo Vorhersagen aus Modellen im Experiment bestätigt werden müssen.

Da bei dieser Methode die passiven wie aktiven Modellkomponenten, sowie die verwendeten Regelungs- oder Steuerungskonzepte in der Modellierung integriert sind, kann man sie als eine synthetische Arbeitsweise bezeichnen [9].

Die Vorwärtssimulation ist dann von Bedeutung, wenn zu einer Bewegung keine Versuche möglich sind, wie bei der Unfallforschung und bei der Klärung der Entstehung einer Bewegung.

# Kapitel 4

# Biomechanische Modellierung

Das Modell spielt bei biomechanischen Untersuchungen eine zentrale Rolle. Es ist die Grundlage für die Simulation und auch für die Durchführung einer Videobildanalyse muss eine Modellvorstellung vorhanden sein, damit die Marker sinnvoll gesetzt werden können und die Daten später verwertbar sind (siehe auch Kapitel 2.3 und Abbildung 5.2 in Kapitel 5.2).

Soll ein Modell verändert werden, etwa zur Validierung oder Erweiterung, so muss jedesmal ein Modellierungszyklus durchlaufen werden [10]. Dazu gehören die Modellierung, die Generierung der Bewegungsgleichungen, das Lösen dieser Gleichungen und die Integration der Ergebnisse, sowie die anschließende Datenverarbeitung, um die Simulationsergebnisse in Diagrammen zu betrachten oder die Bewegung des Modells durch eine Animation zu bewerten.

Bei der Modellierung sollte für die Zahl der Freiheitsgrade ein Minimierungsgrundsatz [3] beachtet werden, jeder Freiheitsgrad erfordert nämlich Informationen, z.B. aus einer Messung, und muss kontrolliert werden. Die Anzahl richtet sich nach der Problemstellung und nach den verfügbaren Informationen für das Modell.

Die in der Biomechanik verwendeten MKS-Modelle bestehen aus kinematischen Ketten, d.h. mehrere Starrkörper sind über Gelenke miteinander verbunden. Um das Verhalten biologischer Systeme nachzuahmen gibt es Modellierungskomponenten, welche die komplizierten Eigenschaften des biologischen Systems abbilden:

**Gelenkanschläge:** Jedes Gelenk besitzt ein natürliches Maß an Bewegungsfreiheit, das durch physiologische Strukturen, wie Bänder, Knochenform oder die über das Gelenk ziehenden Muskeln beschränkt wird. In der Modellierung verhindern winkelabhängige Anschlagmomente ein Durchschlagen des Gelenks oder die Durchdringung von Körperteilen [2] [10].

**Schwabbelmassen:** Sie repräsentieren die Eigenschaften des nicht starr mit den Knochen verbundenen Gewebes (Muskeln, Bindegewebe oder innere Organe), indem Starrkörper oder Punktmassen elastisch und viskos an das Knochensystem gekoppelt werden. Besonders bei hochdynamischen Bewegungen, aber auch beim Laufen ergeben Modelle mit Schwabbelmassen realistischere Belastungswerte [6].

**Kontaktmodellierung:** Erst durch den Kontakt mit dem Boden kann der Mensch sich fortbewegen. Die dabei wirkenden äußeren Kräfte müssen als Bodenreaktionskräfte modelliert werden. Die dafür notwendigen Kraftgesetze werden aus Experimenten ermittelt [5].

Zur Kinematik und Dynamik von Starrkörpern sei auf entsprechende Lehrbücher der klassischen Mechanik [4] [11] und auf das Theoretical Manual von Haug [7] verwiesen.

## 4.1 Regelung

Eine Regelung ist nach Merz [12] ein Vorgang, bei dem die Ist-Größe fortlaufend erfasst, mit der Soll-Größe verglichen und, abhängig vom Ergebnis, so beeinflusst wird, dass es zu einer Angleichung kommt. Diese Rückkopplung findet in einem Regelkreis statt, wie er in Abbildung 4.1 zu sehen ist. Siehe auch Kapitel 3 auf Seite 12.

Hier steht $x_{ist}$ für die Ist-Größe, $x_{soll}$ für die Soll-Größe und $x_d = x_{ist} - x_{soll}$ für die Regelabweichung. Der durch ein Rechteck dargestellte Block versinnbildlicht eine wirkungsmäßige Abhängigkeit zwischen Eingangs- und Ausgangssignal, die durch eine Übertragungsfunktion oder eine Differentialgleichung ausgedrückt werden kann. Am Ausgang des Reglers fällt die Stellgröße an, mit der die Regelgröße $x_{ist}$ der Führungsgröße $x_{soll}$ angeglichen wird. Zwischen dem Stellort und dem Messort von $x_{ist}$ liegt die Regelstrecke.

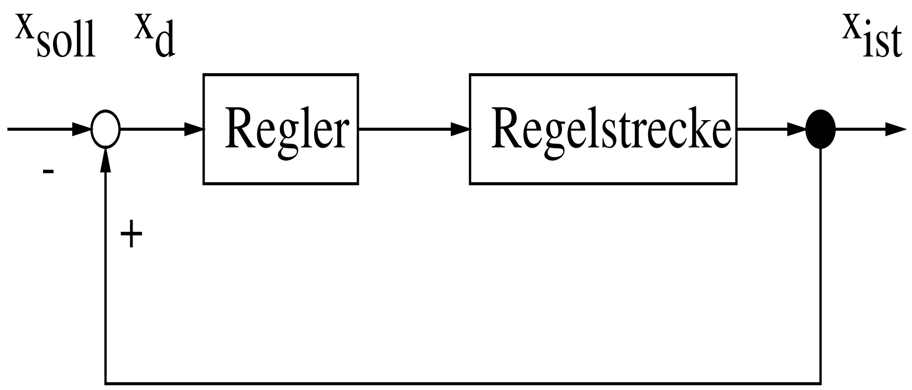


Abbildung 4.1: Regelkreis: Aus den Regelabweichungen $x_d$ (Koordinaten oder Winkel) berechnet der Regler Stellwerte (Kräfte oder Momente), die in die Regelstrecke (Bewegungsgleichungen) eingehen und somit wieder auf die Bewegung zurückwirken.

Beispielsweise wurde in der Arbeit von Krieg [10] eine Regelung benutzt, um einen Bohrhammer mit einem Hand-Arm-System zu halten. Als Führungsgrößen wurden Anfangsgelenkwinkel von Hand-, Ellenbogen- und Schultergelenk gewählt, die Regelgrößen waren die entsprechenden Gelenkwinkel und die Stellgrößen waren die Gelenkmomente.

Das weitere Vorgehen in der vorliegenden Arbeit beinhaltet zwei Anwendungen des verwendeten Regelalgorithmus (siehe Kapitel 5.3). Zum einen werden Gelenkwinkel geregelt, zum anderen werden Markerpunkte geregelt. Im Falle der Markerregelung entspricht $x_{soll}$ einer aus der Videobildanalyse gewonnenen Markerposition, projiziert auf eine Koordinatenachse, beispielsweise der Ort des Kniemarkers in z-Richtung. An jedem Rechenschritt wird dieser Wert mit $x_{ist}$, der aktuellen Modellmarkerposition verglichen. Aus der Differenz $x_d$, der Regelabweichung, berechnet der Regler dann die Kraft, die auf den Modellmarker wirkt, um letzteren der Regelvorgabe näher zu bringen. Da das für alle Marker gilt, kann man sagen, das Modell wird an seinen Markern hinterhergezogen. Wenn der Regelalgorithmus dämpfende Eigenschaften besitzt, gelten auch die Geschwindigkeitsdifferenzen zwischen der gemessenen Markerbewegung und der Modellmarkerbewegung als Regelgröße. Der Regelstrecke entsprechen die Bewegungsgleichungen des MKS, die als Eingabe die Kräfte des Reglers erhalten. Störgrößen treten nicht wie in technischen Systemen von außen auf, sondern kommen durch die gekoppelten Bewegungsgleichungen des mechanischen Systems zustande. Die Bewegung eines Körpersegments beeinflusst die Bewegung der anderen Segmente durch die Zwangskräfte.

In der Technik gibt es verschiedene Realisierungen einer Regelung. Ein Beispiel für ein proportional wirkenden Regler (P-Regler) ist eine lineare Feder: Wird sie durch ein Gewicht in Richtung der Schwerkraft aus ihrer Ruhelage (Solllage) ausgelenkt, wirkt sie der Störung mit der Federkraft entgegen, die proportional zur Auslenkung ist:

$$F_p(t) = -\,p_0\,x_d(t). \tag{4.1}$$

Der Nachteil liegt darin, dass die Feder die ursprüngliche Ruhelage nicht mehr erreicht, die Regeldifferenz wird nicht vollständig ausgeregelt.

Dazu ist der I-Regler da, der die Regelabweichung von Beginn der Regelung an aufintegriert. Bei fortbestehender Regelabweichung wird die Kraft des I-Reglers immer stärker, bis die Differenz ausgeregelt ist. Die Kraft für den I-Regler zeigt Gleichung 4.2 :

$$F_i(t) = -\,c_0 \int_{t_0}^{t} x_d(t')\,dt'. \tag{4.2}$$

Zur Dämpfung wird normalerweise ein parallel zur Feder verlaufender Kolben verwendet, der sich in einer zähen Flüssigkeit bewegt. Die Kraft des D-Reglers ist abhängig von der Geschwindigkeit:

$$F_d(t) = -\,d_0\,\frac{d}{dt}\Big(x_{ist}(t) - x_{soll}(t)\Big). \tag{4.3}$$

Werden die drei Regelanteile gleichzeitig verwendet, spricht man von einem PID-Regler, dessen Stellgröße sich aus der Summe der einzelnen Anteile ergibt.

Selbst bei der Verwendung eines PID-Reglers können sich Schwingungen aufschaukeln, wie es z.B. in der schon zitierten Arbeit von Krieg vorgekommen ist, als die Parameter nicht günstig gewählt worden sind.

# Kapitel 5

# Pseudoinverse Dynamik mit Regelalgorithmus

## 5.1 Messung

Die Messung des Flick-Flacks fand am Olympiastützpunkt in München statt. An der Messung beteiligt waren der Lehrstuhl für Bewegungs- und Trainingslehre der Technischen Universität München, das Institut für Sportwissenschaft der Universität Koblenz-Landau, sowie dem BASiS Institut des TÜV.

Der Bewegungsablauf eines Flick-Flacks kann idealisiert als eine Bewegung angesehen werden, die in einer Ebene verläuft. Daher umfasste das Aufnahmesystem nur eine Digitalkamera, die mit einer Sample-Frequenz von 500 Hz betrieben wurde. Zur Messung der Bodenreaktionskraft beim Absprung diente eine Kraftmessplatte. Mit Hilfe eines Lichtsignals wurden Video und Kraftmessung synchronisiert. Die reflektierenden Marker wurden auf der Haut des Sportlers befestigt. Insgesamt waren es 10 Marker, die an Kopf, Schulter, Ellenbogen, Handgelenk, Brust, Hüfte, Knie, Knöchel, Fußspitze und lateral an der untersten Rippe angebracht wurden. Um die Marker während der gesamten Bewegungsdurchführung deutlich auf dem Video erkennen zu können mussten sie mit Leuchtstrahlern beleuchtet werden.

Der Aufbau sah folgendermaßen aus: An der Wand der Halle wurden Weichbodenmatten aufgestellt, damit ein einheitlicher dunkler Hintergrund entstand und die Marker besser zu erkennen waren. In die davor liegende Mattenbahn wurde die Kraftmessplatte zwischen zwei Bodenmatten platziert. Die Kamera wurde so aufgestellt, dass der Bildausschnitt mit größtem Abbildungsmaßstab den kompletten Flick-Flack enthielt und die Kameraachse senkrecht zur Bewegungsebene stand. Die Startlampe der Kraftmessplatte musste im Videobild unbedeckt zu sehen sein und die Leuchtstrahler durften die Sicht auf einen Marker nicht verdecken.

Während der Justierung der Kamera führte der Sportler Probedurchgänge durch, damit er den Anlauf besser abstimmen konnte, um beim Absprung genau auf der Kraftmessplatte zu landen. Danach wurde eine Kalibrierungsmessung durchgeführt. Zwei Normierungslatten wurden in der Bildebene senkrecht zueinander aufgestellt und aufgenommen. Diese Messung war die Grundlage für die spätere Umrechnung der Bildkoordinaten in reale Koordinaten. Dann wurden sieben Flick-Flack-Messung durchgeführt. Nach jedem Durchgang wurde kontrolliert, ob die Kraftmessplatte einen Ausschlag geliefert hatte, ob alle Marker deutlich auf der Videoaufnahme zu erkennen waren und ob noch alle Marker an ihrer Position klebten, da sie durch die schwitzige Haut leicht abfallen konnten. Eine Sequenz aus drei Bildern der Videoaufnahme ist in Abbildung 5.1 zu sehen.

Die Markerkoordinaten wurden mit dem Softwareprogramm SIMI Motion ausgewertet, so dass sie für diese Arbeit in Realkoordinaten vorlagen, d.h. in Koordinaten bezüglich eines Inertialsystems, dessen Ursprung auf Bodenhöhe lag und so orientiert war, dass die Bewegung in der (y,z)-Ebene verlief. Die Rohdaten hatten eine sehr gute Qualität, sie wurden durch einen gleitenden Mittelwert aus fünf Datenpunkten geglättet. Bei den Bodenkraftwerten wurden die negativen Werte, hervorgerufen durch Eigenschwingungen beim Verlassen der Füsse, durch null ersetzt.

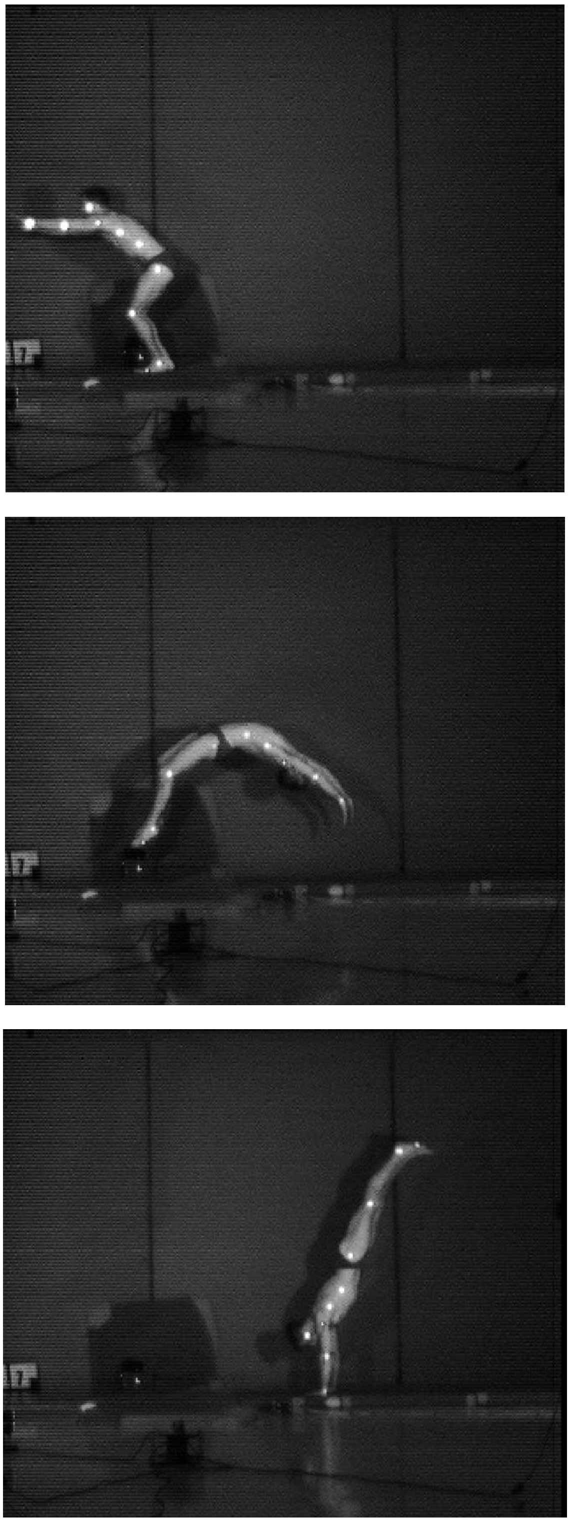

Abbildung 5.1: Sequenz der Videobildaufnahme: Deutlich sind die zehn Marker zu erkennen. Ihre Koordinatenverläufe werden bei der Markerregelung als Eingabe verwendet.

## 5.2 Modell

Das Ziel der Simulation, als dessen Grundlage das Modell dient, ist die möglichst genaue Abbildung des Flick-Flacks, um daraus die Gelenkmomente bestimmen zu können. Die Daten der Videobildanalyse des Flick-Flacks standen schon vor der Modellierung zur Verfügung, deshalb wurde die Struktur des Modells an diese Daten angepasst. Es ist ein ebenes MKS-Modell gewählt worden, dessen Segmente mit Scharniergelenken verbunden sind. Das Modell sollte eine möglichst große Zahl an Gelenken aufweisen. Im Nachhinein kann die Anzahl leicht verringert werden indem ein Scharniergelenk durch ein Gelenk ohne Freiheitsgrad ersetzt wird. Für die Wahl der Gelenkanzahl mussten die Informationen der bei der Messung verwendeten Marker berücksichtigt werden, denn die Winkelstellung eines Gelenks kann erst dann eindeutig bestimmt werden, wenn die Position von drei Markern auf den zwei beteiligten Körpersegmenten bekannt sind. Dies soll in Abbildung 5.2 verdeutlicht werden. Links gibt es mit Marker 1 und Marker 2 auch die gestrichelte Konfiguration. Kommt rechts Marker 3 hinzu, ist das System kinematisch eindeutig festgelegt.

Durch die Vorgabe der Segmentlängen und einem Scharniergelenk, besitzt das zweigliedrige System vier Freiheitsgrade, zwei für die Lage des Schwerpunktes des oberen Körpers

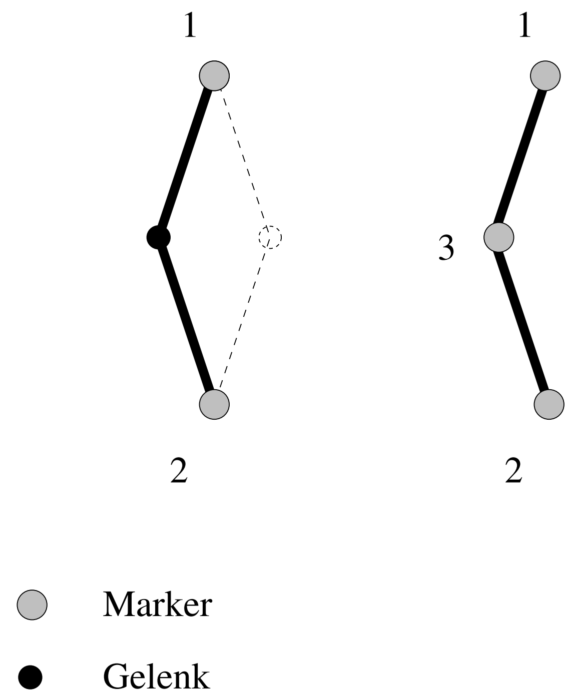


Abbildung 5.2: Markersetzung: Mit drei Markern ist ein Scharniergelenk kinematisch bestimmt (Erklärung im Text).

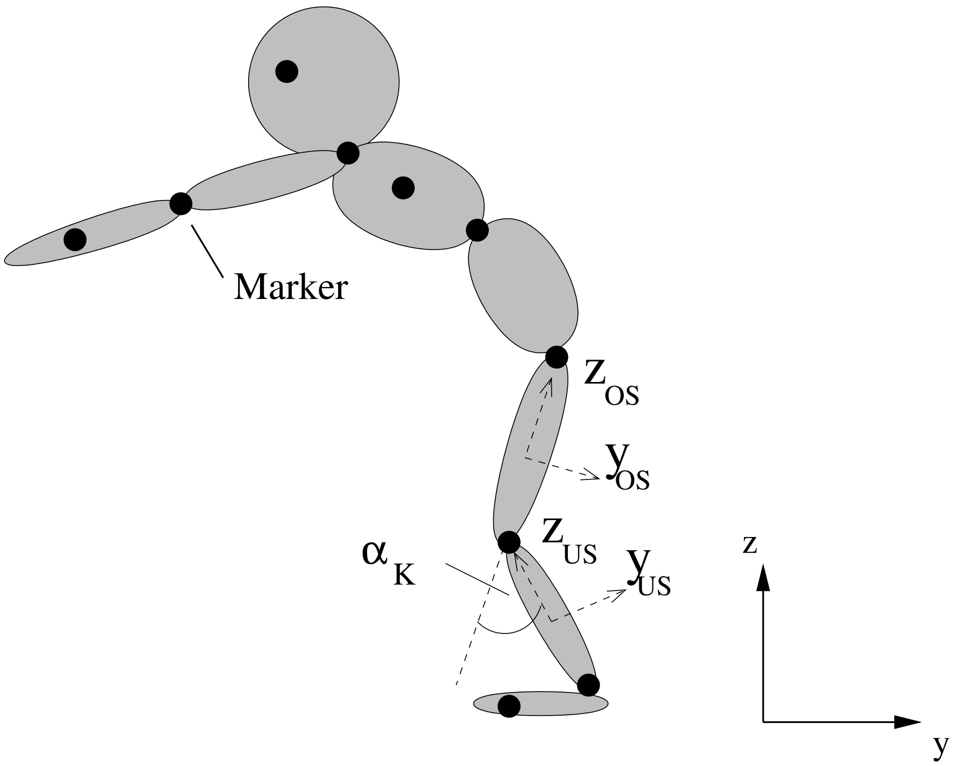


Abbildung 5.3: Achtgliedriges Modell

in der Ebene, einen für seine Orientierung und einen für die Orientierung des unteren Körpers bezüglich des oberen. Die vier Freiheitsgrade der Marker 1 und 2 reichen, wie gesehen, für die eindeutige Beschreibung des Systems nicht aus.

Die zehn Marker aus der Videobildanalyse ließen ein achtgliedriges Modell zu, welches aus Kopf/Hals, Oberarm, Unterarm/Hand, Brustkorb, Lende/Hüfte, Oberschenkel, Unterschenkel und Fuß besteht, so dass sich auf jedem modellierten Gelenk ein Modellmarker befindet (siehe Abbildung 5.3). Die kinematische Kette beginnt am Kopf, alle weiteren Glieder sind über Scharniergelenke gekoppelt. Die Orientierung eines Segments ist relativ zum körperfesten System des in der kinematischen Kette nächsthöher gelegenen Segments angegeben. Wenn sich der Gelenkwinkel $\alpha_K$ des Knies vergrößert, wird der Unterschenkel gegen den Oberschenkel gebeugt. Die Definition der Marker erfolgte analog zu den Markerpositionen auf den Körpersegmenten, wie sie bei der Messung am Sportler angebracht worden sind. Dabei wurden die gelenknahen Marker auf die Gelenkpunkte gesetzt.

Anders als in der bisherigen Beschreibung verläuft die Bewegungsrichtung in der Simulation dann entlang der y-Achse, die Körperlängsachse verläuft weiterhin entlang der

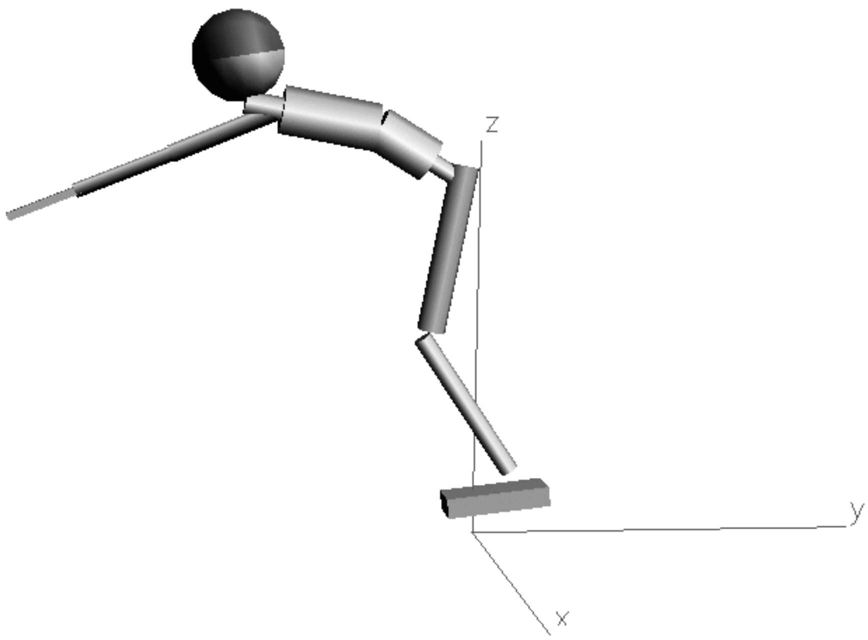


Abbildung 5.4: Darstellung des Simpack-Modells

z-Achse. Die Körperschwerpunkte und die Gelenkpunkte liegen alle in der Sagitalebene[1] mit $x = 0$.

Zur Erstellung des Modells wurden die anthropometrischen Daten aus dem Programm *calcman* (siehe Kapitel 2.4) verwendet. Diese Daten wurden aus einer Parameter-Datei von Simpack eingelesen, so dass sich das Modell leicht parametrisieren ließ. Bei der Erstellung des Modells wurde ein Mehraufwand getrieben, weil zusätzlich Hals, Hand und Hüfte geschaffen wurden, die nachträglich durch das Sperren der Gelenke mit Kopf, Unterarm und Lende fest verbunden wurden. Dies hatte ausschließlich optische Gründe. Für die Darstellung des Modells wurden einfache geometrische Körper gewählt (siehe Abbildung 5.4).

Das bis dahin passive Modell benötigte noch aktive Elemente für die pseudoinverse Dynamik, wozu zwei Kraftroutinen implementiert wurden (siehe Anhang). Die benutzerdefinierten Programme in Simpack enthalten standardmäßig Fortran-Code. Die Gleichun-

[1]Die Sagitalebene steht senkrecht zur Körperquerachse und teilt den Körper in zwei annähernd gleiche Hälften( [14] ).

gen für den Regelalgorithmus sind im nächsten Abschnitt 5.3 angeführt und werden dort erläutert. Die Kraftroutine für die Markerregelung liefert die Kräfte des Reglers zwischen einem definierten Marker auf dem Modell und dem Marker im Ursprung des Inertialsystems, getrennt in y- und z-Komponente. Bei der Kraftroutine der Gelenkwinkelregelung wirkt das Moment des Reglers am definierten Gelenk und zeigt in positive x-Richtung, so dass sich das Knie beugt, wenn das Moment ein positives Vorzeichen besitzt. Eingangsgrößen der Regler sind einerseits die Markerpositionen und Gelenkwinkel, sowie die zugehörigen Geschwindigkeiten, die in der Simpack-Routine während der Simulation berechnet werden. Andererseits werden für die Bildung der Regelabweichungen und der Differenzgeschwindigkeiten auch die entsprechenden Größen der Sollfunktionen benötigt. Die Sollstellungen der Marker wie der Gelenkwinkel sind durch Input-Funktionen zugänglich. Die Berechnung der Soll-Geschwindigkeiten erfolgt aus der Taylor-Reihe der Inputfunktionen in erster Ordnung.

Simpack lässt in den Kraftroutinen nur Winkel zwischen $|0\dots2\pi|$ zu. Das stellt ein Problem dar, wenn der Winkel einer Input-Funktion diesen Bereich verlässt, weil dann ein Sprung von $\pm2\pi$ erfolgt. Das entsprechende Gelenk dreht einmal durch, was in der Animation zu sehen ist. Außerdem entsteht ein Geschwindigkeitspeak, der sich durch den Regelalgorithmus auch auf andere Gelenke überträgt. Die Kraftroutine musste so erweitert werden, dass der $2\pi$-Sprung kompensiert wird, in diesem Falle wird $\mp2\pi$ zu dem Winkel addiert.

Die Anfangsstellung des Modells wird durch die Position und Orientierung des Kopfes und allen Winkeln der nachfolgenden Gelenke festgelegt. Sie wurden durch einen Simulationslauf ermittelt, bei dem die Input-Funktionen der Kraftroutinen konstant auf den Anfangswert gesetzt waren. Die Anfangsgeschwindigkeiten wurden aus den ersten drei Punkten der Input-Funktionen bestimmt. Die Punkte wurden durch quadratische Polynome interpoliert und aus den Koeffizienten die Ableitung berechnet. Dazu wurde das Programm *polcoe*, aus den Numerical Recipes in C [15], verwendet.

## 5.3 Regelalgorithmus

Der Regelalgorithmus in der vorliegenden Arbeit ist an einen PID-Regler angelehnt (siehe Kapitel 4.1), seine Anteile sind mit nichtlinearen Funktionen versehen. Die verwendeten Parameter wurden mit einer Parameterstudie gesucht, als Gütekriterium galt die Bewegungsreproduktion der Simulation. Dabei wurde nicht eine Optimierung mit minimalem Abstandsquadraten zwischen Ist- und Soll-Kurven durchgeführt, sondern aus dem subjektiven Vergleich zwischen Ist- und Soll- Kurven und dem Vergleich zwischen der Gesamtmodellbewegung mit der auf dem Video festgehaltenen Bewegung wurden günstige Parameter ausgewählt. Je größer die Regelanteile sind, desto besser wird der Flick-Flack reproduziert. Das steht im Gegensatz zu dem Ziel, möglichst physiologische Kräfte oder Momente zu erhalten. Daher musste ein Kompromiss gefunden werde, der darin bestand, die geringsten Kräfte oder Momente zu wählen, welche die Bewegung noch nachahmen.

In den nächsten Unterkapiteln sind die drei Regelanteile des Regelalgorithmus beschrieben. Die Erklärungen beziehen sich auf den Fall, bei dem Marker geregelt werden und durch Kräfte in ihre Solllage gezogen werden. Die Gleichungen gelten analog für die Regelung von Gelenkwinkeln, wenn die Kräfte durch Momente und die Koordinaten durch Winkel ersetzt werden.

Die Kräfte der einzelnen Regelanteile und die Regelabweichung sind nicht explizit von der Zeit abhängig.

### 5.3.1 P-Regler

Der P-Regler liefert komponentenweise eine Kraft $F_p(t)$ in Abhängigkeit von der Regelabweichung $x_d(t)$.

$$\begin{aligned} F_p(t) &= p_0\Big(x_d(t) + p_a \arctan(p_s\, x_d(t))\Big). \qquad (5.1) \\ x_d(t) &= x_{ist}(t) - x_{soll}(t). \end{aligned}$$

Gleichung 5.1 lässt sich als Feder mit künstlich nichtlinearem Kraftgesetz verstehen. Für die Konstante $p_s > 0$ steigt die Rückstellkraft $F_p(t)$ an der Ruhelage überproportional an und nähert sich dann dem proportionalen Verhalten. Da der Arcustangens, im positiven Limes gegen $\pi/2$ strebt, nähert sich $F_p(t)$ für positive $x_d(t)$ einer Geraden mit einer Steigung von $p_0$ und einem Offset von $p_0\pi/2$ asymptotisch an. Hintergrund für die Wahl dieses, um die Ruhelage steilen, P-Reglers ist ein schneller Kraftaufbau, damit der Marker schneller wieder an die Solllage gelangt. Befindet sich der Marker in der Sollstellung, dann wird er nach einer kleinen Auslenkung stark zurückgezogen. Die Konstante

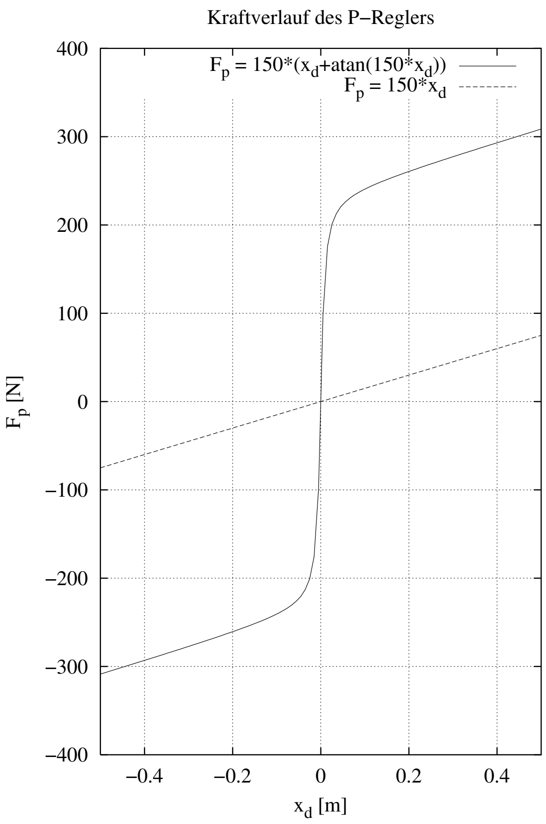


Abbildung 5.5: Kraftverlauf des P-Reglers in Abhängigkeit von der Regelabweichung, mit den Parametern $p_0 = 150\,\mathrm{Nm}^{-1}$ und $p_s = 150\mathrm{m}^{-1}$.

$p_a = 1\,\mathrm{m}$ dient zum Transport der Einheit. Der Kraftverlauf des P-Reglers mit den Parametern $p_0 = 150\,\mathrm{Nm}^{-1}$ und $p_s = 150\,\mathrm{m}^{-1}$ ist in Abbildung 5.5 zu sehen.

Die Regelkurve ist symmetrisch zum Nullpunkt, deutlich ist der steile Anstieg an der Ruhelage zu erkennen, bei $x_d(t) = 0.1\,\mathrm{m}$ hat der Arcustangens schon 95.8 % seines Maximalwertes angenommen, die Kraft beträgt hier $240,6\,\mathrm{N}$. Im weiteren Verlauf wird sich die Kurve immer mehr einer Geraden mit der Steigung $150\,\mathrm{Nm}^{-1}$ annähern (gestrichelte Kurve).

### 5.3.2 D-Regler

Der D-Regler liefert komponentenweise eine Kraft in Abhängigkeit von der Geschwindigkeitsdifferenz $v_d(t)$ und von der Regelabweichung $x_d(t)$, was von einem typischen D-Regler abweicht. Die Kraft $F_d(t)$ auf einen Modellmarker stellt Gleichung 5.2 dar:

$$\begin{aligned} F_d(t) &= d_0 \left( \frac{1}{\left(\frac{x_d^2(t)}{d_b}+1\right)} + d_g \right) v_d(t). \\ v_d(t) &= \frac{d}{dt}\left( x_{ist}(t) - x_{soll}(t) \right). \end{aligned} \tag{5.2}$$

Wegen der proportionalen Abhängigkeit von $v_d(t)$, kann $F_d(t)$ als eine dämpfende Kraft angesehen werden. Sie ist zudem abhängig von der Konstanten $d_0$ und dem in Klammern stehenden Ausdruck. Beide, im weiteren als Dämpfungsterm bezeichnete Größen, besitzen einen peakförmigen Verlauf, wie es in Abbildung 5.6 mit den Parametern $d_0 = 80\,\mathrm{Nsm}^{-1}$, $d_g = 0.1$, $d_b = 0.005\,\mathrm{m}^{-2}$ zu sehen ist. Das absolute Maximum bei $x_d(t) = 0$ beträgt 88 N. Mit wachsendem $x_d^2(t)$ geht der Dämpfungsterm gegen eine Grunddämpfung von $d_0\,d_g = 8\,\mathrm{Nsm}^{-1}$, die Breite der Kurve wird durch $d_b$ bestimmt. Die halbe Höhe wird bei $x_d(t) = \pm 0.025\,\mathrm{m}$ erreicht und die Halbwertsbreite beträgt 0.050 m.

Dahinter steckt die Idee, dass die Dämpfung nahe der Sollstellung am höchsten sein soll. Der Marker bewegt sich dort wie in einer zähen Masse, wird nicht so stark weggezogen und wirkt dadurch glättend gegenüber Ausreisern in den gemessenen Markerwerten, die als Regelvorgabe dienen.

Bei der Wahl des Parameters $d_b$ war darauf zu achten, dass die Halbwertsbreite des Dämpfungsterms nicht zu groß wurde, da sonst die Sollstellung nicht mehr erreicht wird, die Dämpfung verhindert dann ein Heranziehen an den Sollwert.

### 5.3.3 I-Regler

Der I-Regler soll ein „Gedächtnis“ besitzen, das heisst, er soll zurückliegende Werte „vergessen“. Eine Möglichkeit der Realisierung bestände darin, den Integranden eines üblichen I-Reglers, aus Gleichung 4.2, mit einem Faktor $e^{\frac{t_0-t}{\tau}}$ zu versehen, wie es in Gleichung 5.3 dargestellt ist. Dadurch würde das Gedächtnis exponentiell abnehmen und weiter in der Vergangenheit liegende Werte von $x_d$, würden vergessen werden.

$$F_i(t) = -i_0 \int_{t_0}^{t} x_d(t)\, e^{\frac{t_0-t}{\tau}}\, dt. \tag{5.3}$$

Da in den Simpack-Kraftroutinen keine integralen Ausdrücke möglich sind, wurde der I-Regler durch ein gekoppeltes Differentialgleichungssystem verwirklicht. Die im letzten Absatz beschriebene Möglichkeit der Realisierung war aufgrund von numerischen

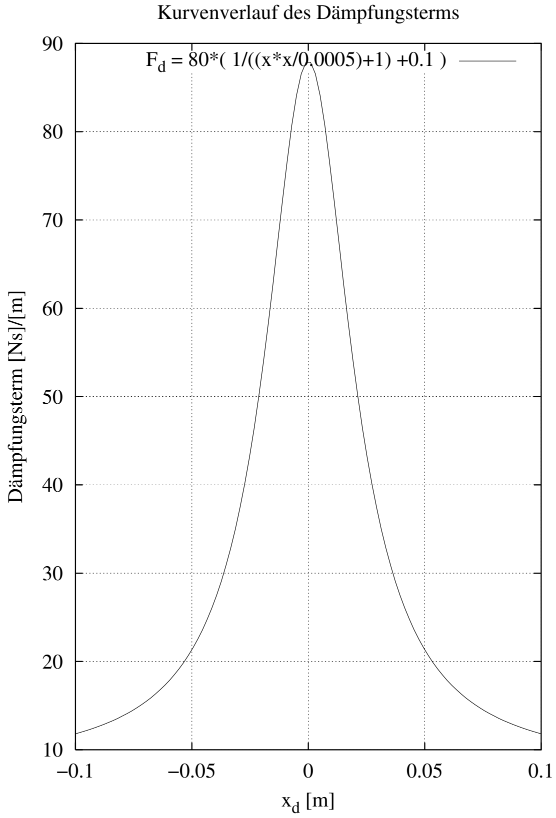


Abbildung 5.6: Kurvenverlauf des Dämpfungsterms in Abhängigkeit von $x_d$, mit den Parametern $d_0 = 80\,\mathrm{Nsm}^{-1}$, $d_g = 0.1$ und $d_b = 0.005\,\mathrm{m}^{-2}$.

Problemen mit divergierenden Termen nicht möglich, stattdessen wurde folgendes System gewählt, wobei $y_1(t)$, multipliziert mit einer Konstanten $i_0 = 200$, der Beitrag des I-Reglers ist :

$$\dot{y}_1(t) = c_1 \left( x_d(t) + c_2 \arctan(i_s\, x_d(t)) \right) - i_v |v_d(t)| y_2(t). \tag{5.4}$$

$$\dot{y}_2(t) = -y_2/\tau + c_3\, y_1. \tag{5.5}$$

Im Nachfolgenden soll das Gleichungssystem erläutert werden. Der Ausdruck $\dot{y}_1(t) = x_d(t)$ in Gleichung 5.4, für sich genommen, ergäbe den I-Regler von Gleichung 4.2. Durch die Konstante $i_s$ kann, wie beim P-Regler, die Steigung um den Nullpunkt verändert werden. Mit einem Wert von $i_s = 0.5\,\mathrm{m}^{-1}$ beträgt die Gesamtsteigung des in Klammern stehenden Ausdrucks am Nullpunkt 1.5. Dies ist um ein Faktor 100 kleiner als beim

P-Regler, denn es kommt für den I-Regler nicht darauf an möglichst schnell zu sein, sondern die Regelabweichung möglichst vollständig auszuregeln. Dann wird der Term $i_v|v_d(t)|y_2(t)$ abgezogen, $i_v = 10\,\mathrm{m}^{-1}$ ist die Konstante für das "Vergessen", $v_d(t)$ die Differenzgeschwindigkeit zwischen Soll- und Ist-Stellung und $y_2(t)$ ist eine exponentiell abklingende Größe mit der Zeitkonstanten $\tau = 0,158\,\mathrm{s}$ aus Gleichung 5.5. Wenn $v_d(t)$ groß wird, vergisst der I-Regler also mehr. Dieser Fall tritt beispielsweise ein, wenn der Marker weit von der Sollstellung abweicht. Der P-Regler beschleunigt den Marker, der möglichst schnell an den Sollwert gezogen werden soll, während der Beitrag des I-Regler kleiner wird. Die Konstanten $c_1 = 1\,\mathrm{Nms}^{-1}$, $c_2 = 1\,\mathrm{m}$, $c_3 = 1\,\mathrm{s}^{-1}$ transportieren die Einheiten.

## 5.4 Ergebnisse

**Zwei Simulationsschritte zur Berechnung der Gelenkmomente**

Das weitere Vorgehen umfasst zwei Simulationsschritte, nach denen die Gelenkmomente des Flick-Flacks berechnet werden. Aus dem ersten Simulationsschritt, der Markerregelung, werden die Gelenkwinkelverläufe des Modells bestimmt und als Eingabe für den zweiten Simulationsschritt verwendet. In dieser Gelenkwinkelregelung werden zusätzlich noch zwei Marker des Modells ihren experimentellen Trajektorien hinterhergezogen und stellen damit eine künstliche äußere Kraft dar, welche die Bodenreaktionskraft ersetzen soll. Nachteil der Vorgehensweise ist, dass diese Kraft ständig auf das Modell wirkt, also auch währen sich das Modell ballistisch in der Luft gewegt. Sie wurde dennoch gewählt, da die Reaktionskraft auf die Hand nicht gemessen worden war und die alleinige Verwendung der gemessenen Bodenreaktionskräfte für die Füße keine guten Ergebnisse lieferte. Die beiden Schritte sind in Abbildung 5.7 veranschaulicht.

Mit diesem Verfahren wurde der Flick-Flack analysiert. Die Ergebnisse sind in Kapitel 5.4.1 dargestellt.

**Pseudoinverse Dynamik mit einer „idealen“ Messung**

Um zu untersuchen welchen Einfluss die Qualität der experimentellen Daten auf das Ergebnis des verwendeten Verfahrens hat, wurden in einer zweiten Analyse angenommen, dass sich das Modell ab dem Zeitpunkt $t_a$ = 0,2 s frei bewegt, d.h. es wirken keine Kräfte oder Momente mehr auf das Modell. Die Gelenkwinkel, die sich so ergeben, können als „ideal“ bezeichnet werden, und die Momente, die sich im zweiten Schritt ergeben, müssten konstant null sein. Die Regelung erfolgt in diesem Fall nur mit dem P- und dem D-Anteil, da der I-Anteil bei $t_a$ nicht null ist. Außerdem werden keine Marker berücksichtigt, die zusätzlich am Modell ziehen. Die Ergebnisse dieser Simulation sind in Kapitel 5.4.2 dargestellt.

Für die durchgeführten Simulationen wurden folgende Regelkonstanten gewählt: $i_0$ = 20000, $p_0$ = 15000 Nm$^{-1}$, $d_0$ = 8000 Nsm$^{-1}$.

### 5.4.1 Pseudoinverse Dynamik eines Flick-Flacks

**Erster Simulationsschritt: Markerregelung**

Die Abbildungen 5.8 bis 5.19 zeigen die Ergebnisse des ersten Simulationsschrittes für die Regelung der Knie-, Hüft- und Schultermarker entlang der z-Richtung. Jeweils unterhalb der Diagramme von Soll- und Ist-Koordinaten sind die Regelabweichungen und unter

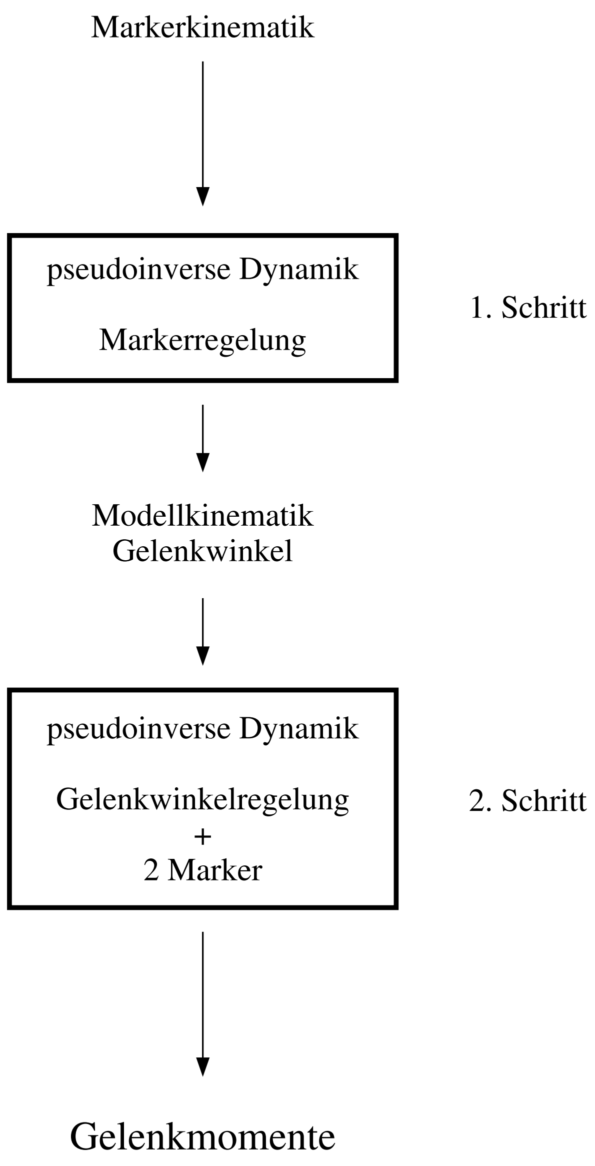


Abbildung 5.7: Vorgehen: Zwei Simulationsschritte zur Bestimmung der Gelenkmomente.

den Gesamtmomenten sind die Anteile des Reglers dargestellt. Exemplarisch werden die Diagramme des Kniereglers erläutert .

Die Ungenauigkeit der Anfangsbedingungen wurde durch die Regelung ausgeglichen, daher steigt die Kurve des Kniemarkers in Abbildung 5.8 zuerst an, dann erkennt man den Landevorgang ($t = 0,1\,\mathrm{s}$). Während der Streckung des Knies zum Absprung und während der ersten Flugphase, bis die Hände den Boden wieder verlassen, steigt der z-Wert bis auf 1.5 m an. In der Landephase nimmt er wieder bis auf 0.4 m ab. Nach der prellenden Landung bewegt sich der Sportler wieder nach oben.

Die Kurve des Markers ist der Sollkurve sehr ähnlich, doch die Bewegung des Markers verläuft versetzt zur Sollbewegung. In der Aufwärtsbewegung befindet sich der Marker unterhalb des Sollwertes, seine Regelabweichung (siehe Abb. 5.10) pendelt bei 0.3 s um $-6$ cm, was eine zeitliche Differenz von 10 ms zur Sollposition ausmacht, das entspricht 5 Datenpunkten. Die dynamische Bewegung des Flick-Flacks führt zu einer Phasenverschiebung, während der die Kraft betragsmäßig in der Größenordnung von $10^4\,\mathrm{N}$ liegt. Solche Kräfte sind unphysiologisch hoch, man stelle sich vor, ein Gewicht von 3 t hänge am Knie des Sportlers, um ihn auf den Boden zu ziehen.

Mit den anderen Markern verhält es sich ebenso. Je größer die Regelkonstanten $p_0$, $d_0$, $i_0$ (siehe Kapitel 5.3) gewählt werden, desto kleiner werden die Regelabweichungen und desto größer aber auch die Kräfte. Umgekehrt führen kleinere Parameter zu einer größeren Regelabweichung und der Flick-Flack kann nicht mehr reproduziert werden. Bei der Verwendung eines standardmäßigen Feder-Dämpfer-Elements, wie es in Simpack zur Verfügung steht, um das Modell den Markern hinterherzuziehen, waren ähnlich hohe Kräfte wie in dieser Simulation erforderlich, um einen Flick-Flack zu erhalten.

Diese systematische Regelabweichung oder Phasenverschiebung verhält sich anders als die statistisch verteilten Messfehler bei der Videobildauswertung, das Modell ist gewissermaßen „immer zu spät dran“.

Die durchgezogenen Kurven der Markerkoordinaten verlaufen glatter als die gestrichelten Kurven ihrer Sollwerte, was mit der Wahl einer Regelung beabsichtigt war. Das Modell ist nicht, wie bei der inversen Dynamik, starr an die gemessene Markerbewegung gekoppelt, sondern besitzt durch den elastischen Anteil des P-Reglers noch Freiheiten, wodurch das Modell optimal in die Markerwolke hineinpasst. Die Markerposition bezüglich eines Gelenks muss daher nicht genau bekannt sein.

In den Abbildungen 5.11, 5.15 und 5.19 ist zu erkennen, dass die P- und D-Anteile gegeneinander arbeiten: Wenn der P-Anteil einen Marker beschleunigt, erhöht sich der dämpfende D-Anteil. Die hohen Kräfte zu Beginn kommen durch nicht exakt gewählte Anfangsbedingungen zustande.

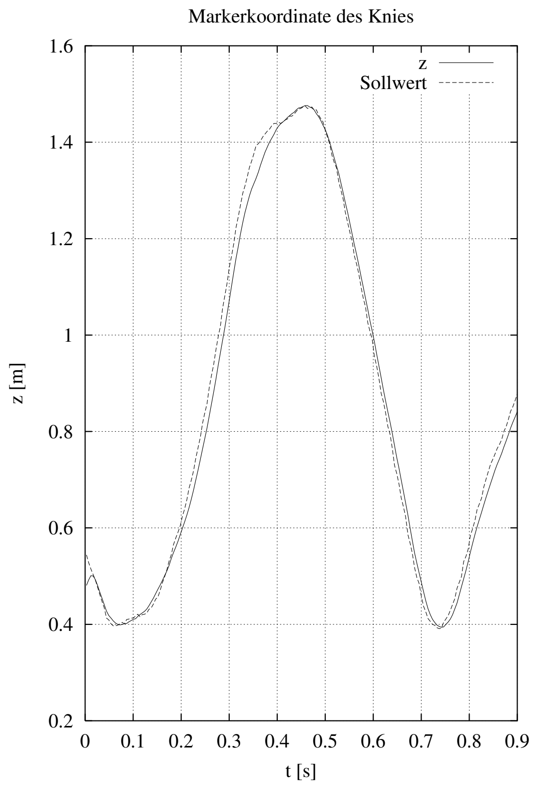


Abbildung 5.8: z-Koordinate des Kniemarkers (durchgezogen) und der zugehörige Sollwert (gestrichelt).

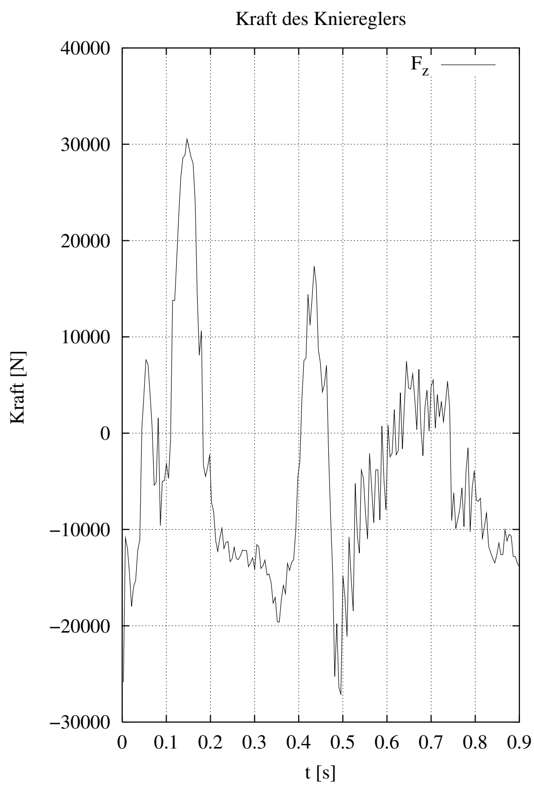


Abbildung 5.9: Kraft $F_z$ des Reglers auf den Kniemarker. Die Kraft besteht aus der Summe der drei Anteile.

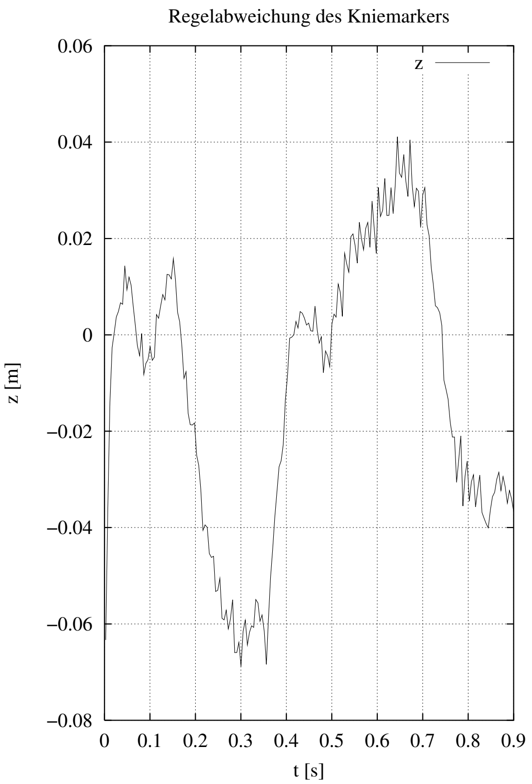


Abbildung 5.10: Regelabweichung des Kniemarkers in z-Richtung.

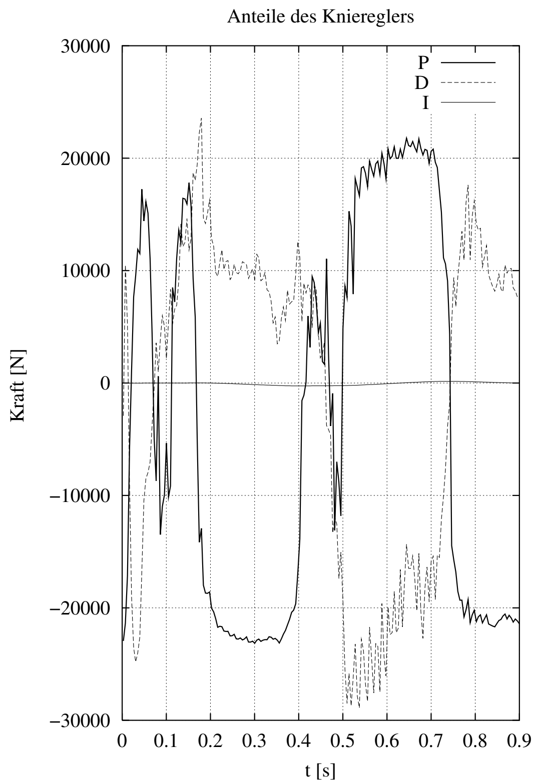


Abbildung 5.11: P-, D- und I-Anteil des Kniemarkerreglersg.

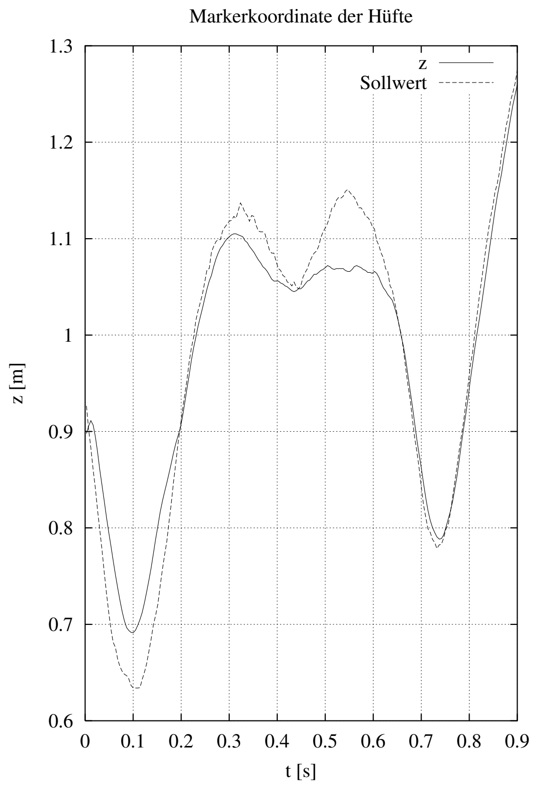


Abbildung 5.12: z-Koordinate des Hüftmarkers (durchgezogen) und der zugehörige Sollwert (gestrichelt).

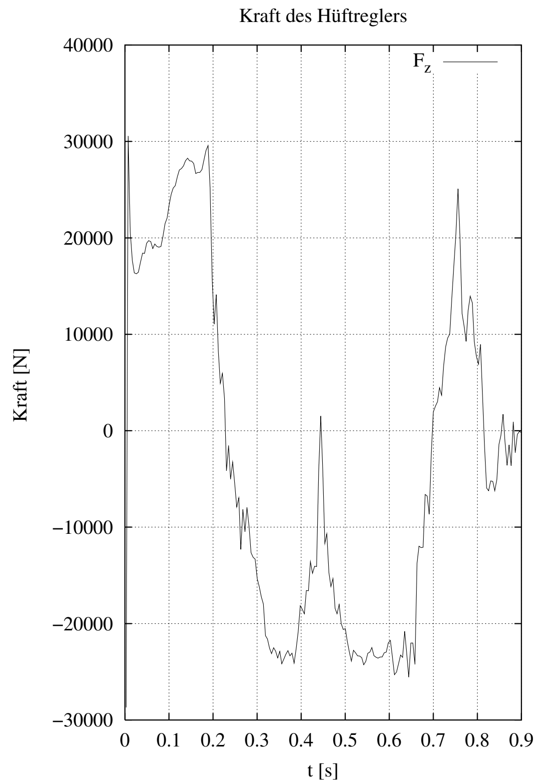


Abbildung 5.13: Kraft $F_z$ des Reglers auf den Hüftmarker. Die Kraft besteht aus der Summe der drei Anteile.

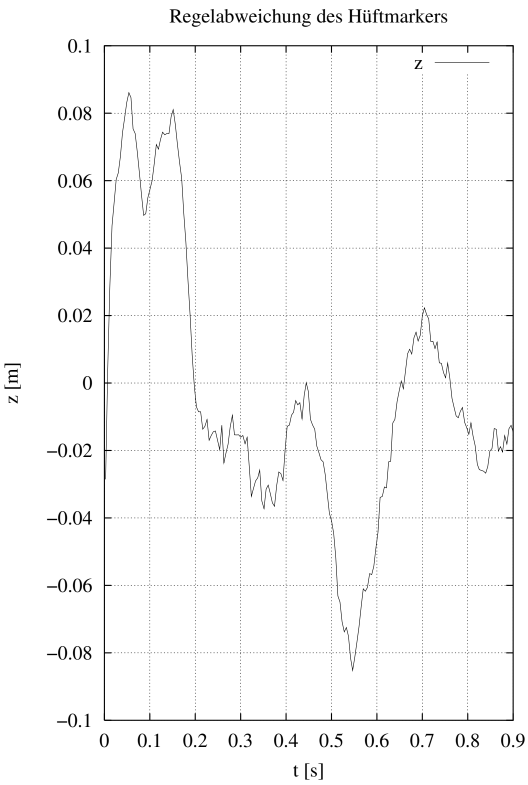


Abbildung 5.14: Regelabweichung des Hüftmarkers in z-Richtung.

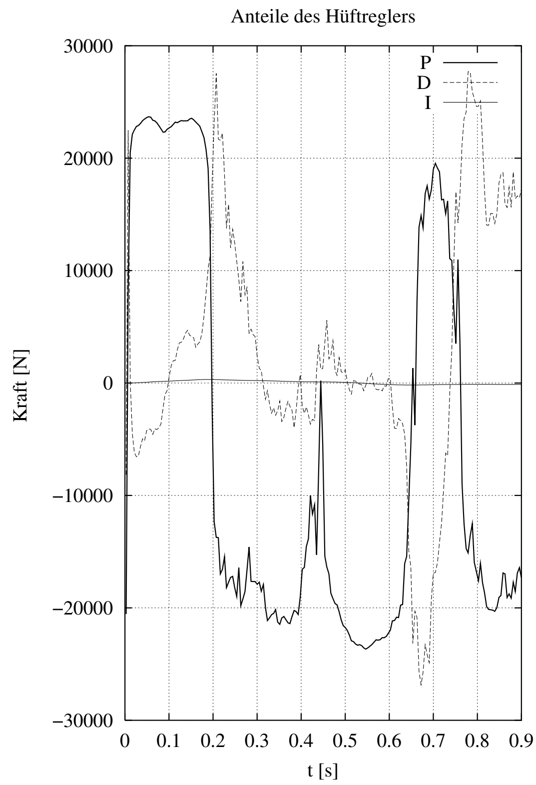


Abbildung 5.15: P-, D- und I-Anteil des Hüftmarkerreglers.

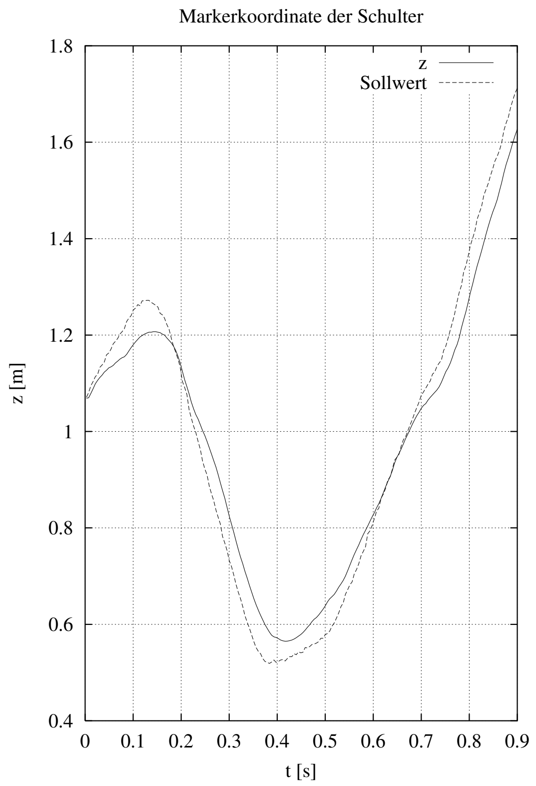


Abbildung 5.16: z-Koordinate des Schultermarkers (durchgezogen) und der zugehörige Sollwert (gestrichelt).

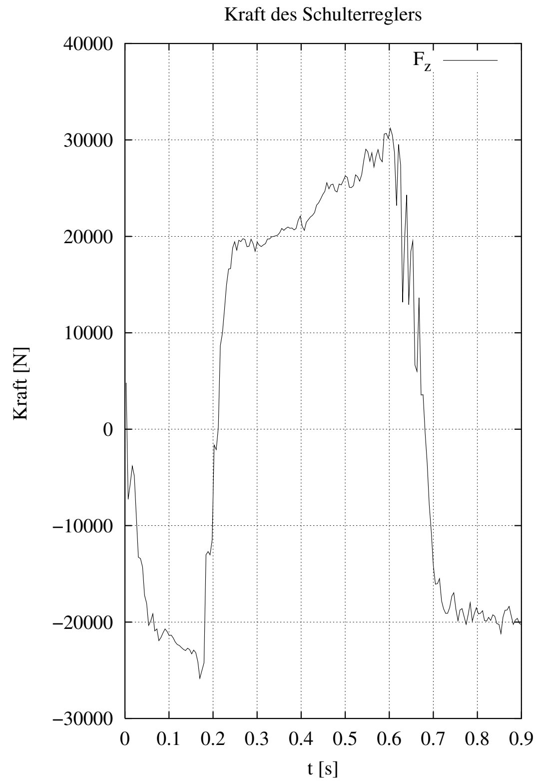


Abbildung 5.17: Kraft $F_z$ des Reglers auf den Schultermarker. Die Kraft besteht aus der Summe der drei Anteile.

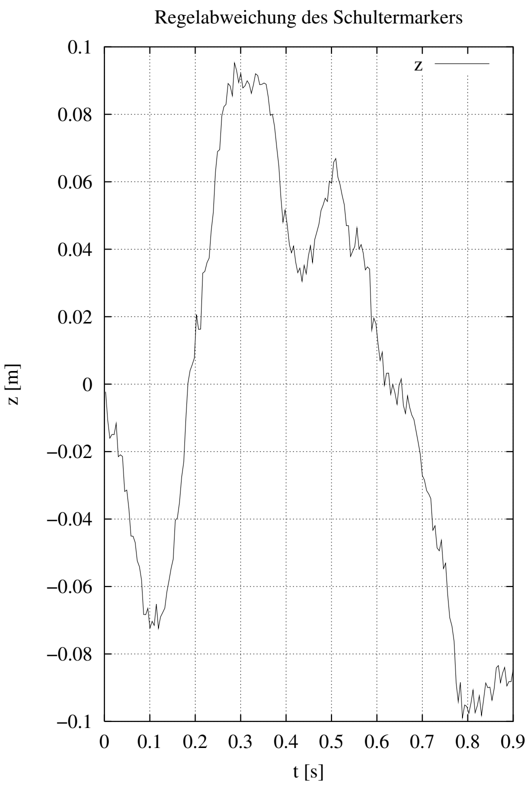


Abbildung 5.18: Regelabweichung des Schultermarkers in z-Richtung.

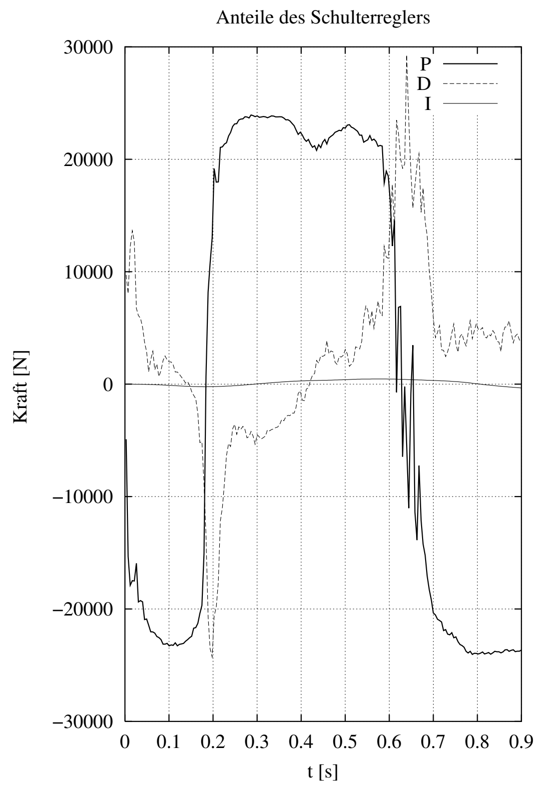


Abbildung 5.19: P-, D- und I-Anteil des Schultermarkerreglers.

### Zweiter Simulationsschritt: Gelenkwinkelregelung

Die Abbildungen 5.20 bis 5.40 zeigen die Ergebnisse des zweiten Simulationsschrittes für die Winkelregelung der Knie-, Hüft- und Schultergelenke. Zusätzlich sind hier noch Diagramme für die Winkelgeschwindigkeiten der Gelenke, die Geschwindigkeitsdifferenz zwischen Soll- und Ist-Winkelgeschwindigkeit und die D-Anteile der Regler angeführt.

Zunächst soll der Winkelverlauf für das Kniegelenk kurz besprochen werden, dazu muss man wissen, dass der Winkel null beträgt, wenn das Knie gestreckt ist und größer wird, wenn das Knie gebeugt wird. Das Gelenk wird während der Landung zuerst gebeugt und beim Absprung gestreckt. Zu Beginn der Drehung um die Querachse des Sportlers beugt er das Knie wieder, zum einen, um sein Trägheitsmoment für die Drehung zu verringern, und zum anderen, um bei der anschließenden Stützphase der Hände, nocheinmal Schwung durch das Strecken des Knies zu holen. Dabei wird das Knie des Modells überstreckt. Dies ist ein Artefakt und rührt wahrscheinlich von einer falschen Markersetzung her. Während der zweiten Landung, bei etwa 0.7 s, wird das Knie wieder gebeugt.

In den Diagrammen der Gelenkwinkel ist wieder die Phasenverschiebung zu erkennen, wie sie bei den Markerkoordinaten besprochen wurde. Durch das bei der Markerregelung auftretende Zurückbleiben des Modells, ist zu verstehen, warum die Bodenreaktionskräfte in diesem zweiten Schritt nicht zu der Modellbewegung gepasst haben. Durch die zeitliche Phasenverschiebung wirkte die Kraft auf das Modell in einer zu frühen Bewegungsphase.

Durch die numerische Ableitung in erster Ordnung der Sollwinkel, kommt es zu dem sehr rauhen Kurvenverlauf der Sollwinkelgeschwindigkeiten. Die einzelnen Sekanten fügen sich nicht zu einen glatten Kurvenverlauf. Dieses Verhalten überträgt sich auch auf die Geschwindigkeitsdifferenzen und die von ihnen abhängigen Größen, wie die D-Anteile (Gleichung 5.2).

In den Abbildungen 5.20, 5.27 und 5.34 zeigen die Gelenkwinkel einen glatteren Verlauf als die Sollwinkel und ähneln diesen bis auf die bereits erwähnte Phasenverschiebung. In einer Animation ist der Bewegungsablauf schön als Flick-Flack zu erkennen. Die dahinter steckenden Momente sind jedoch unphysiologisch und von einem hochfrequenten Rauschen überlagert. Solche Momente erscheinen ziemlich sinnlos. Man könnte durch einen Tiefpassfilter die groben Verläufe der Gelenkmomente erhalten, doch übersteigen deren Größenordnungen die physiologischen Maximalmomente um das 500fache.

Die Gründe für diese Gelenkmomentverläufe können in den experimentellen Messwerten der Videobildanalyse liegen. Numerisch Artefakte durch Interpolation, Differentiation oder Rundung können die Werte weiter verschlechtern. Denkbar wären auch ungünstig gewählte Regelparameter, durch die sich das mechanische System. Dabei beinflussen sich die einzelnen Segmente gegenseitig. Bei einem fünfgliedrigen Modell, das durch Sperren von Gelenken entstand, konnten die Momente um einen Faktor 10 verkleinert werden.

Eine weitere Ursache könnte die künstliche Führung durch die zwei zusätzlichen Marker an Knöchel und Rippe sein.

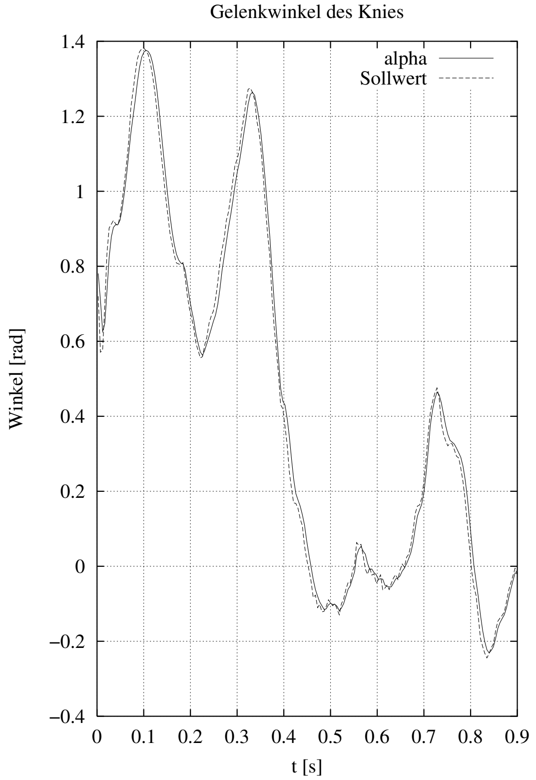


Abbildung 5.20: Gelenkwinkel des Knies (durchgezogen) und der zugehörige Sollwert (gestrichelt).

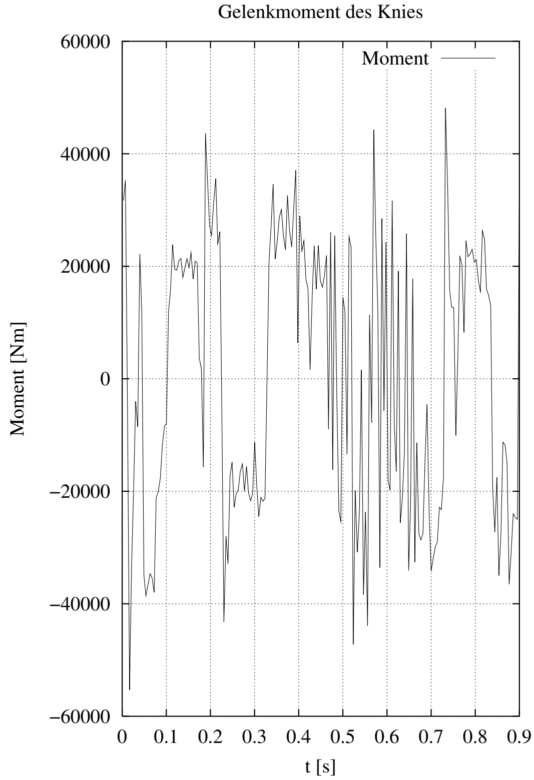


Abbildung 5.21: Moment des Reglers auf das Kniegelenk.

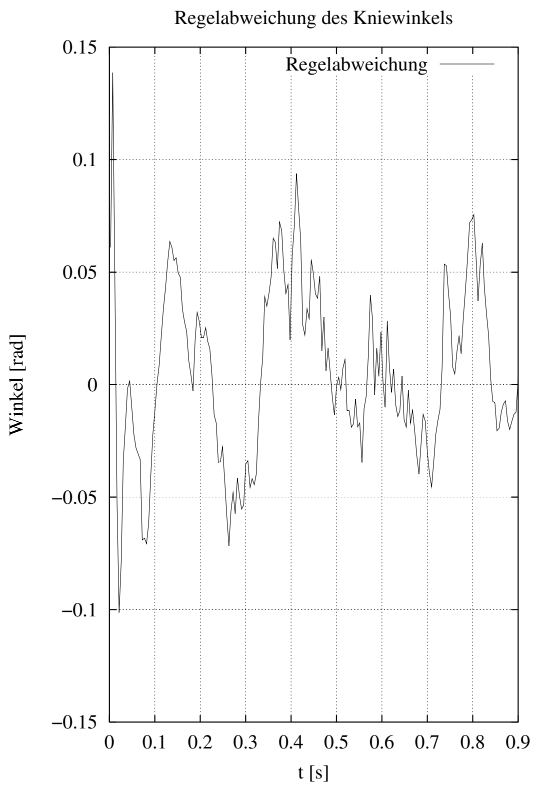


Abbildung 5.22: Regelabweichung des Kniewinkels.

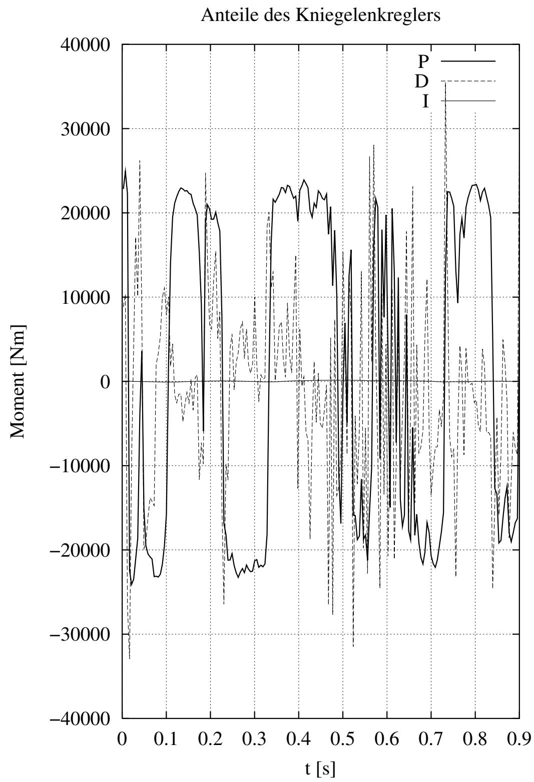


Abbildung 5.23: P-, D- und I-Anteil des Kniegelenkreglers.

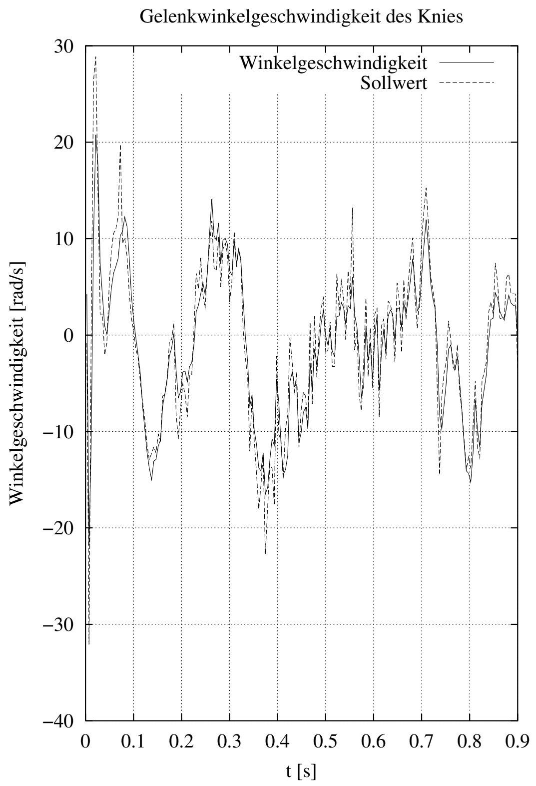


Abbildung 5.24: Gelenkwinkelgeschwindigkeit des Knies (durchgezogen) und der zugehörige Sollwert (gestrichelt).

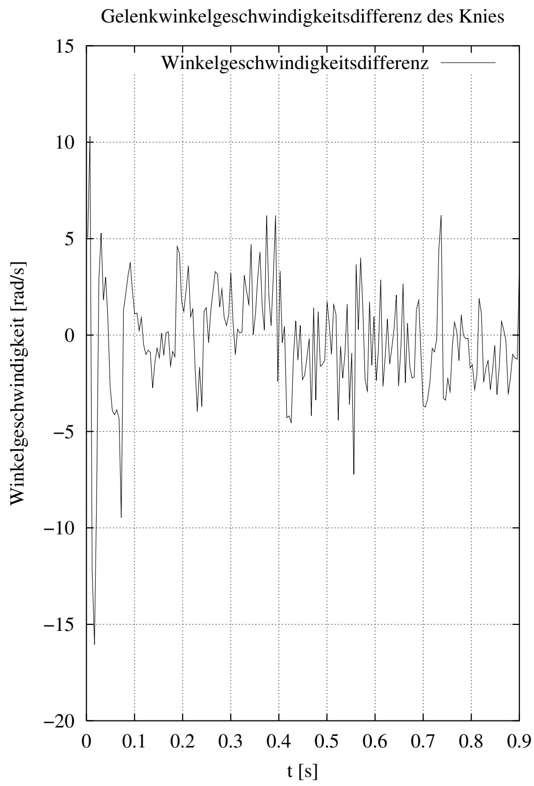


Abbildung 5.25: Differenz zwischen Ist- und Soll-Winkelgeschwindigkeit bei der Rotation des Kniegelenks

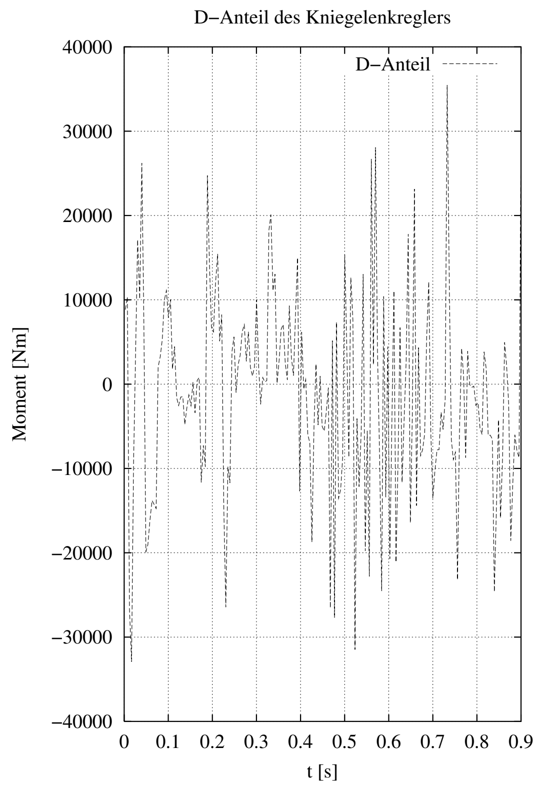


Abbildung 5.26: P-, D- und I-Anteil des Kniegelenkreglers.

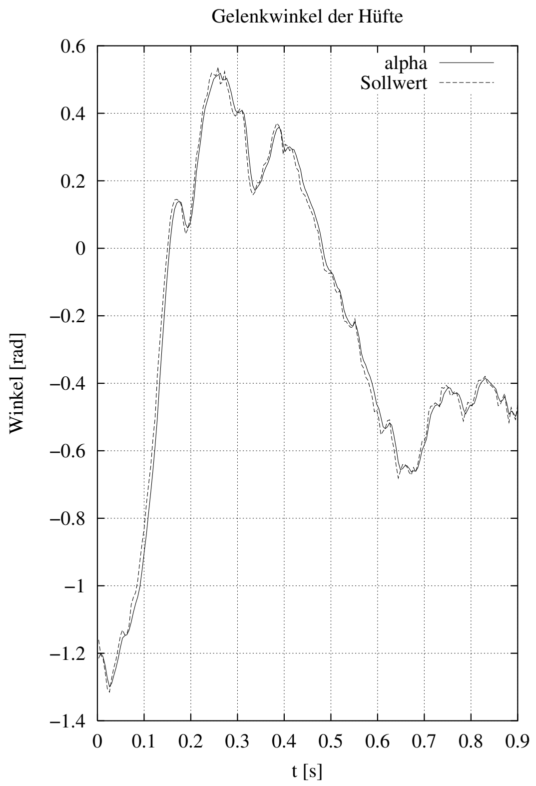


Abbildung 5.27: Gelenkwinkel der Hüfte (durchgezogen) und der zugehörige Sollwert (gestrichelt).

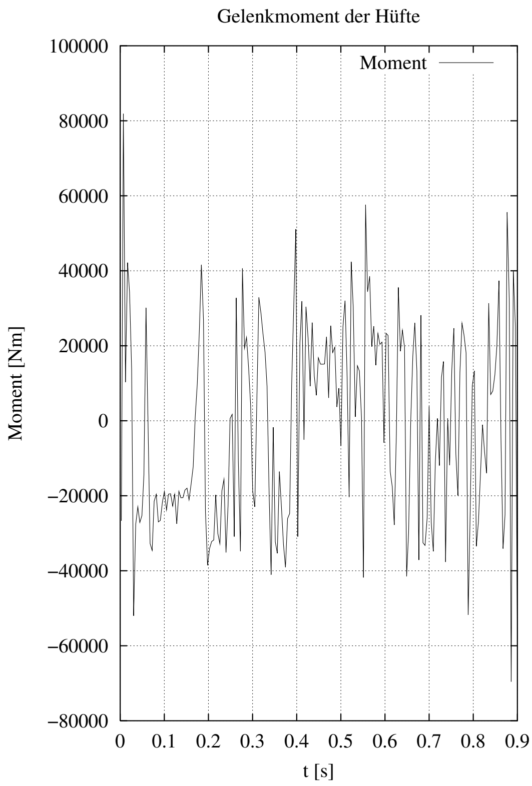


Abbildung 5.28: Moment des Reglers auf das Hüftgelenk.

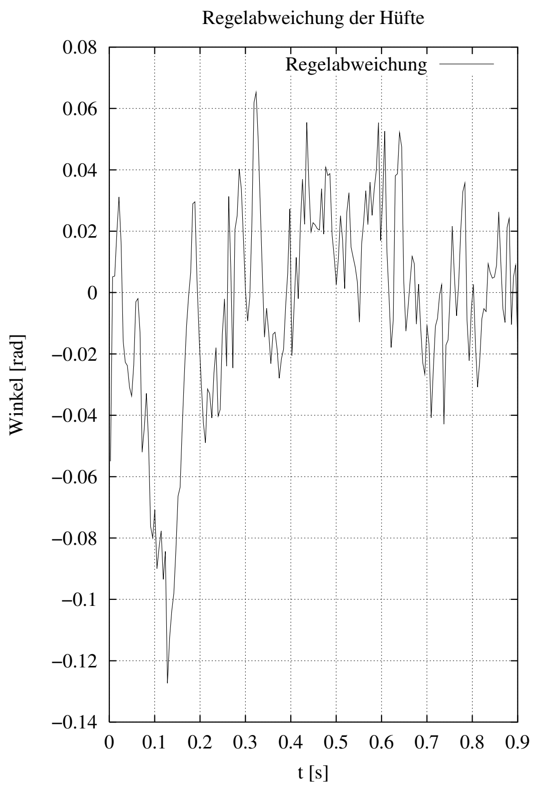


Abbildung 5.29: Regelabweichung des Hüftwinkels.

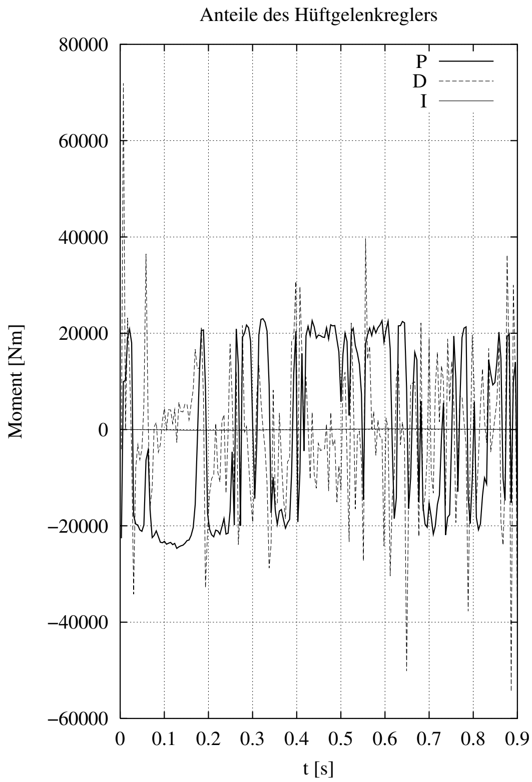


Abbildung 5.30: P-, D- und I-Anteil des Hüftgelenkreglers.

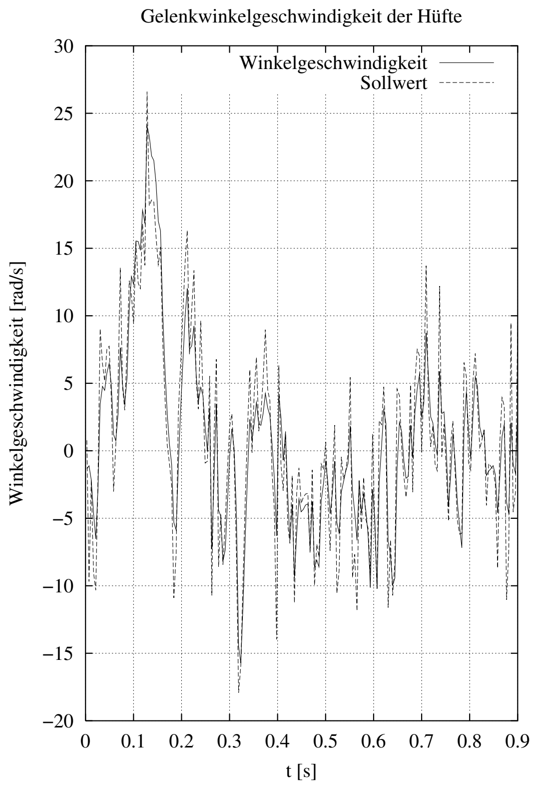


Abbildung 5.31: Gelenkwinkelgeschwindigkeit der Hüfte (durchgezogen) und der zugehörige Sollwert (gestrichelt).

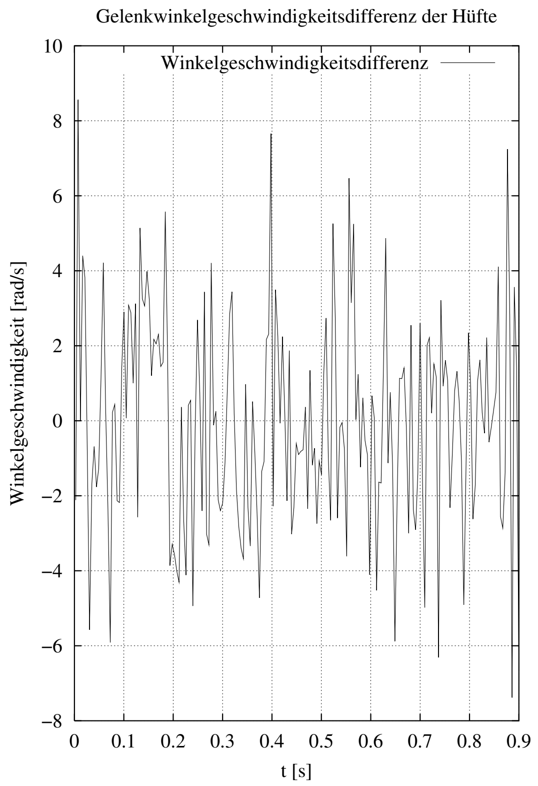


Abbildung 5.32: Differenz zwischen Ist- und Soll-Winkelgeschwindigkeit bei der Rotation des Hüftgelenks

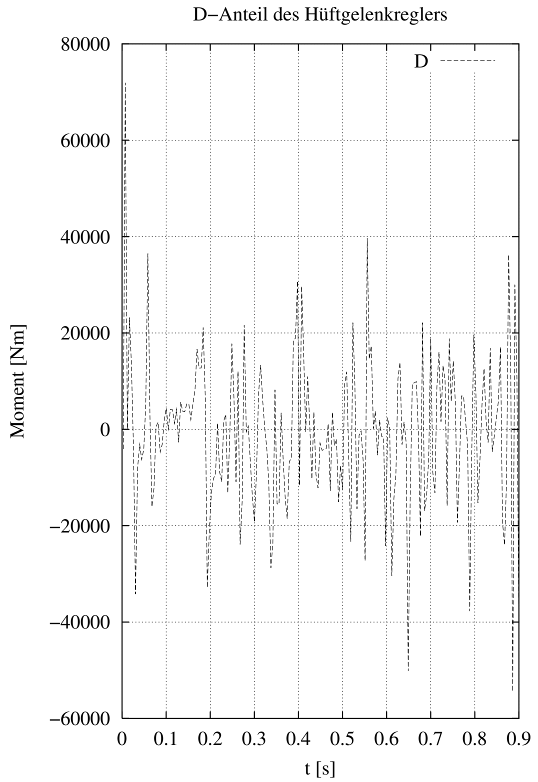


Abbildung 5.33: P-, D- und I-Anteil des Hüftgelenkreglers.

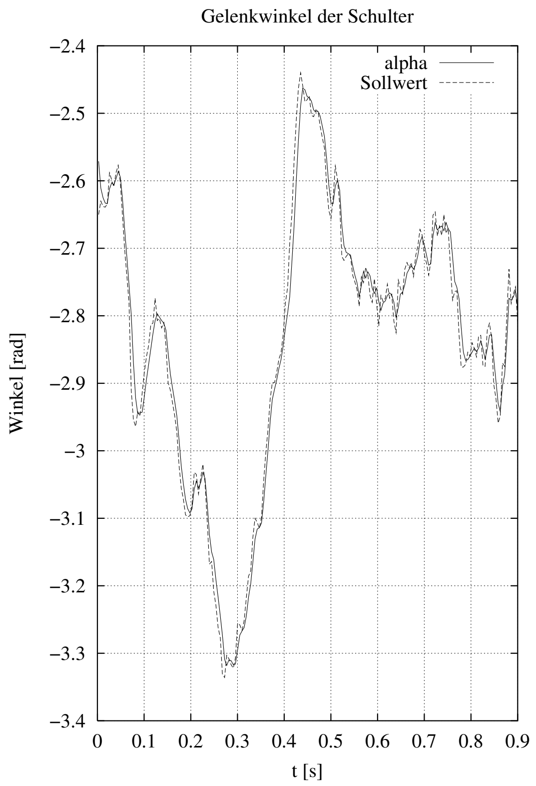


Abbildung 5.34: Gelenkwinkel der Schulter (durchgezogen) und der zugehörige Sollwert (gestrichelt).

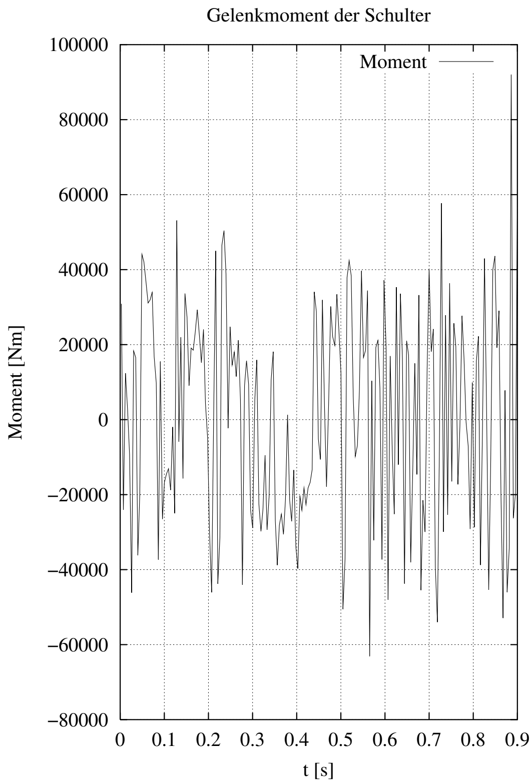


Abbildung 5.35: Moment des Reglers auf das Schultergelenk.

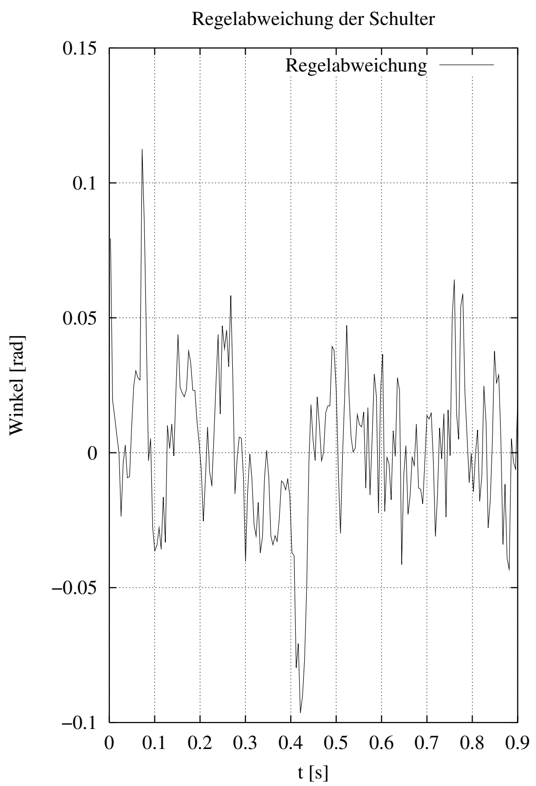


Abbildung 5.36: Regelabweichung des Schulterwinkels.

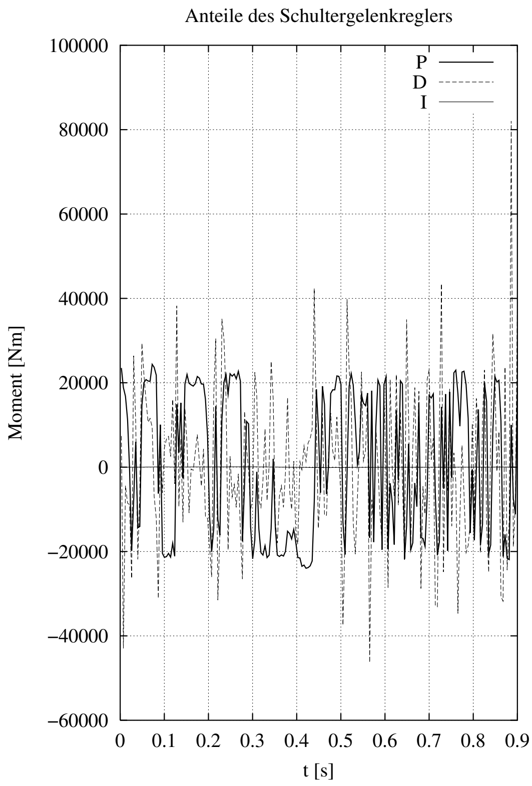


Abbildung 5.37: P-, D- und I-Anteil des Schultergelenkreglers.

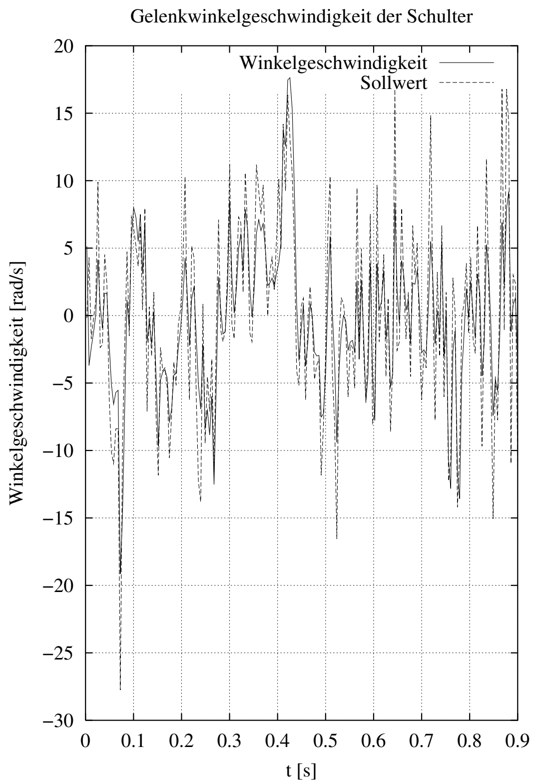


Abbildung 5.38: Gelenkwinkelgeschwindigkeit der Schulter (durchgezogen) und der zugehörige Sollwert (gestrichelt).

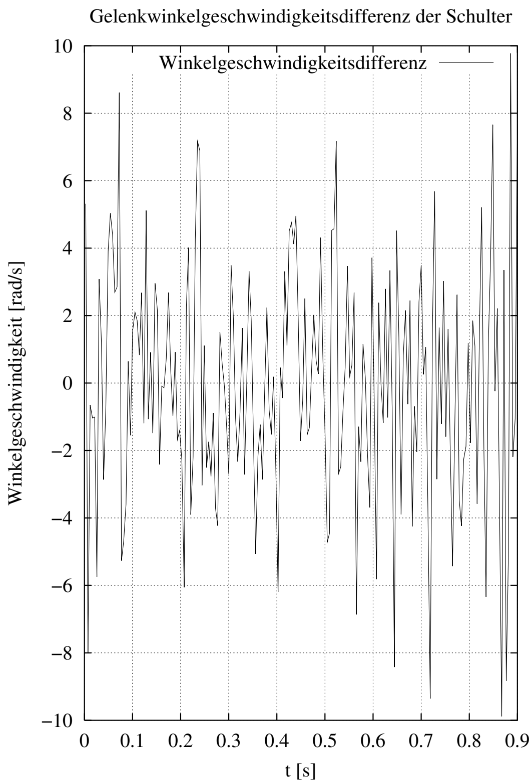


Abbildung 5.39: Differenz zwischen Ist- und Soll-Winkelgeschwindigkeit bei der Rotation des Schultergelenks

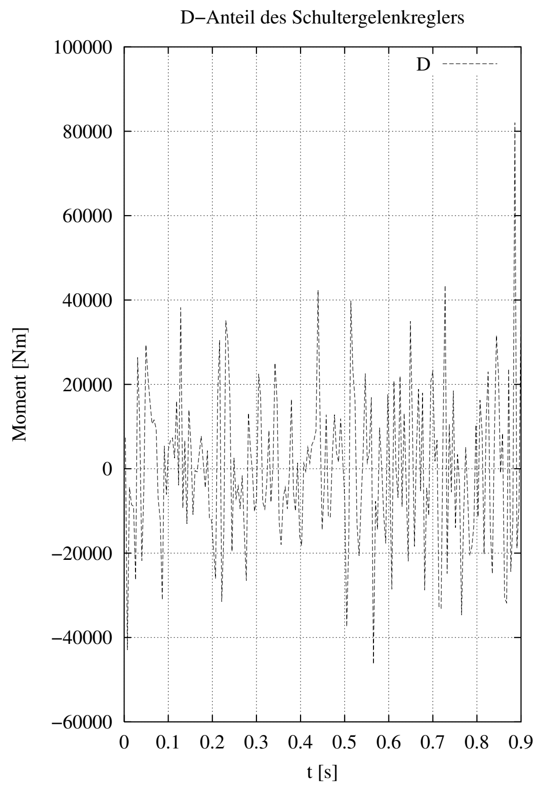


Abbildung 5.40: P-, D- und I-Anteil des Schultergelenkreglers.

### 5.4.2 Bewertung des Regelalgorithmus mit einer idealen Messung

Die Ergebnisse des zweiten Simulationsschrittes in den Abbildungen 5.41 bis 5.61 sind analog denen im letzten Abschnitt angeordnet.

Die Kurven in den Diagrammen verlaufen nach dem Zeitpunkt $t_a$ glatter als davor. Besonders deutlich ist dies bei den Winkelgeschwindigkeiten zu sehen. Trotzdem sind die Momente ähnlich groß wie bei den vorigen Simulationen. Ein Effekt durch Mess- und Auswertefehler kann jetzt ausgeschlossen werden, ebenso der Einfluss der Markerregelung auf die Momente. Daher kann der gewählte Ansatz nicht sinnvoll sein, um aus den kinematischen Daten die Momente zu berechnen. Schon nach kleinen Abweichungen von den Sollwinkeln führt die Regelung zu sehr großen Momenten und ist nicht mehr in der Lage die Sollwerte zu erreichen.

Bei der Suche nach der Ursache der unphysiologisch hohen Gelenkmomente ist noch zu überprüfen, ob die Nettomomente, die sich aus den Gelenkmomenten und den Zwanskräften ergeben, der Bewegungen der Segmente entsprechen.

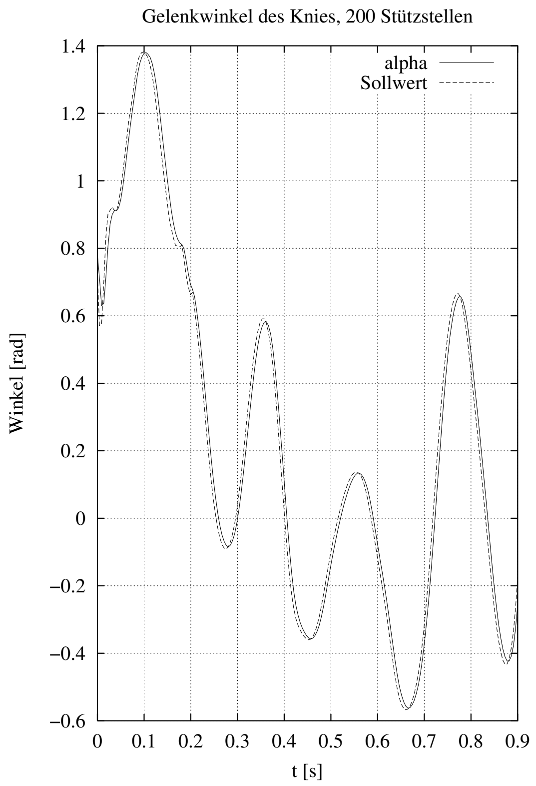


Abbildung 5.41: Gelenkwinkel des Knies (durchgezogen) und der zugehörige Sollwert (gestrichelt).

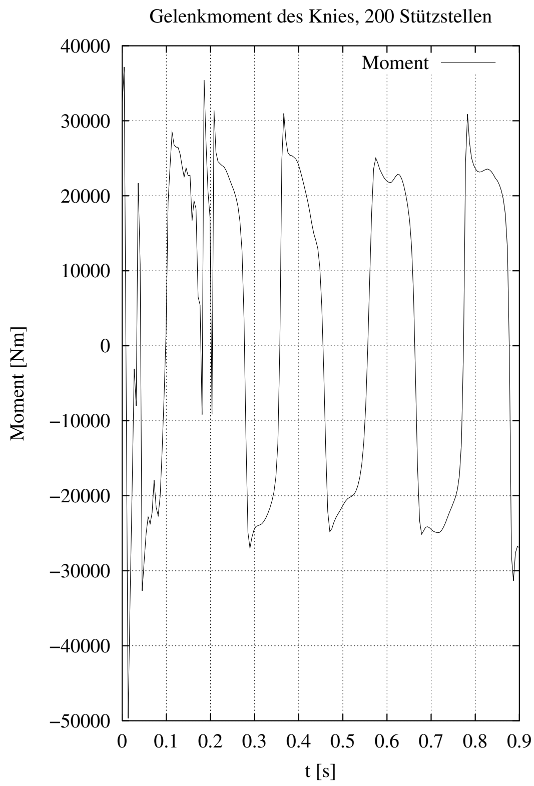


Abbildung 5.42: Moment des Reglers auf das Kniegelenk.

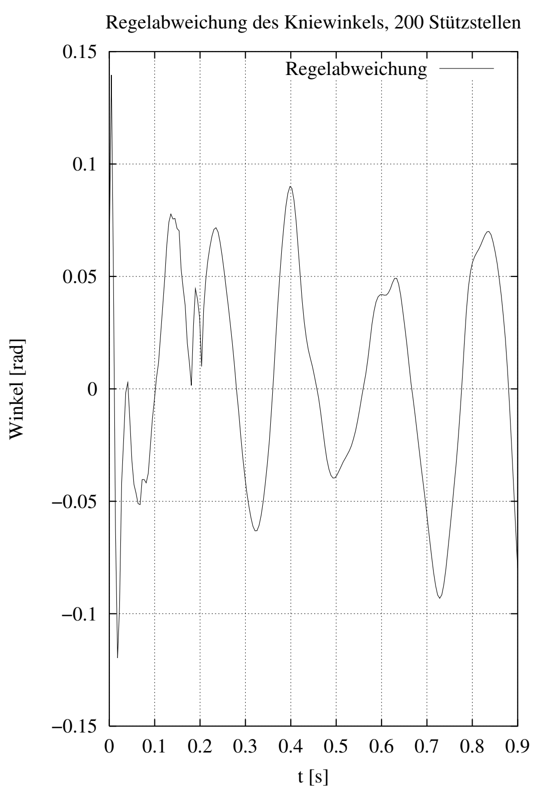


Abbildung 5.43: Regelabweichung des Kniewinkels.

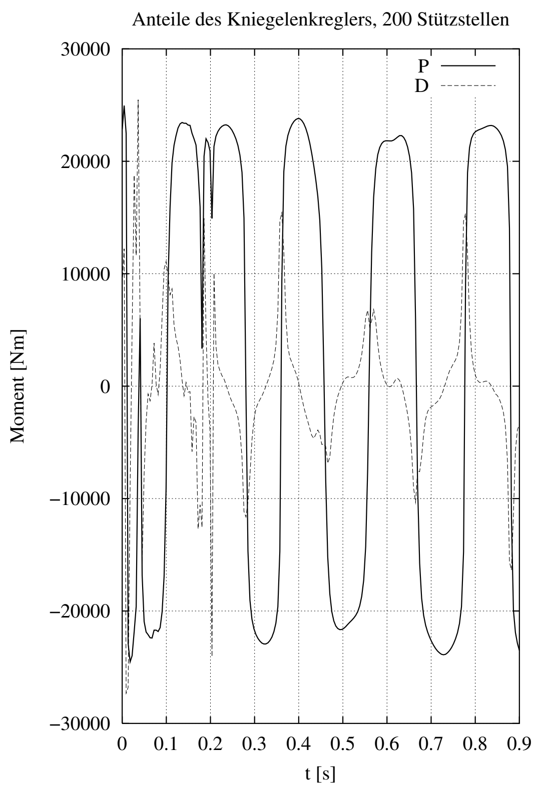


Abbildung 5.44: P- und D-Anteil des Kniegelenkreglers.

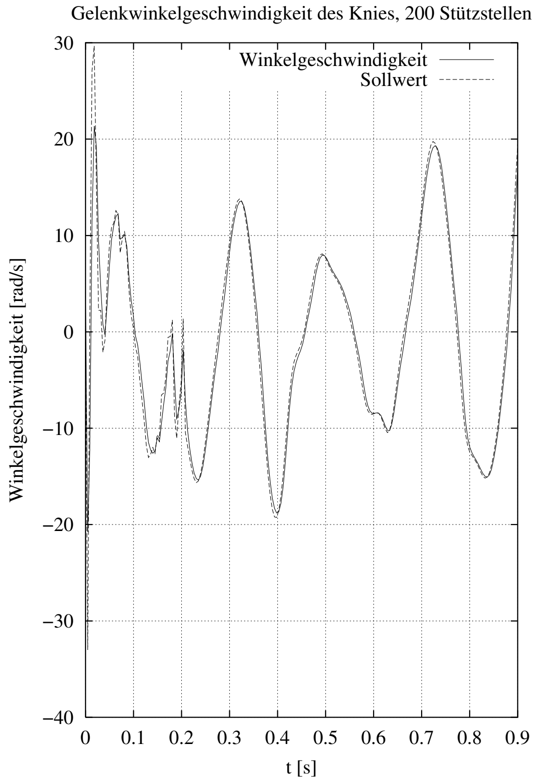


Abbildung 5.45: Gelenkwinkelgeschwindigkeit des Knies (durchgezogen) und der zugehörige Sollwert (gestrichelt).

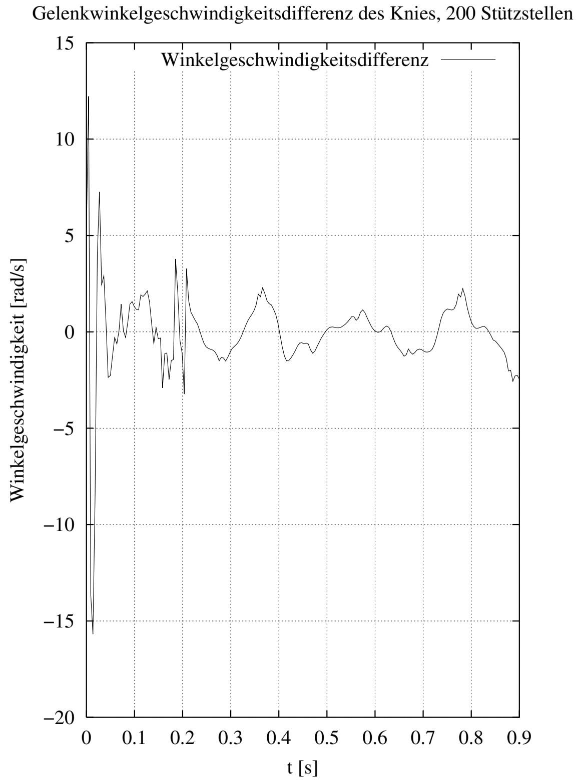


Abbildung 5.46: Differenz zwischen Ist- und Soll-Winkelgeschwindigkeit bei der Rotation des Kniegelenks.

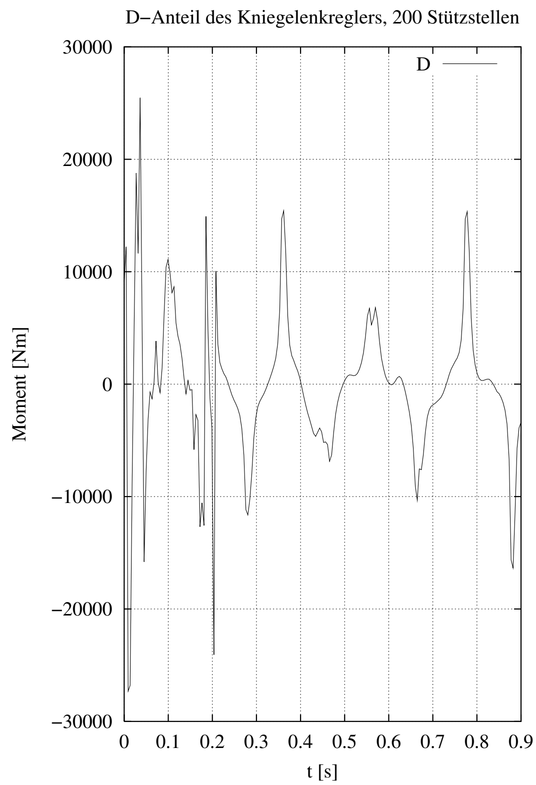


Abbildung 5.47: P- und D-Anteil des Kniegelenkreglers.

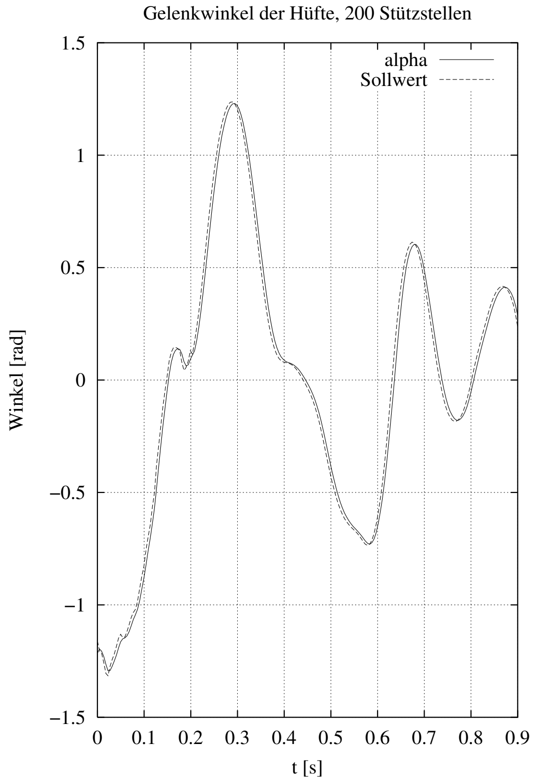


Abbildung 5.48: Gelenkwinkel der Hüfte (durchgezogen) und der zugehörige Sollwert (gestrichelt).

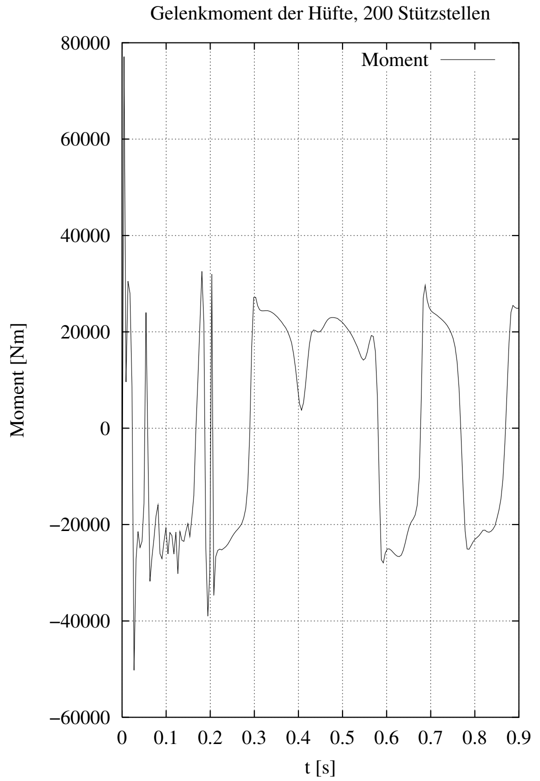


Abbildung 5.49: Moment des Reglers auf das Hüftgelenk.

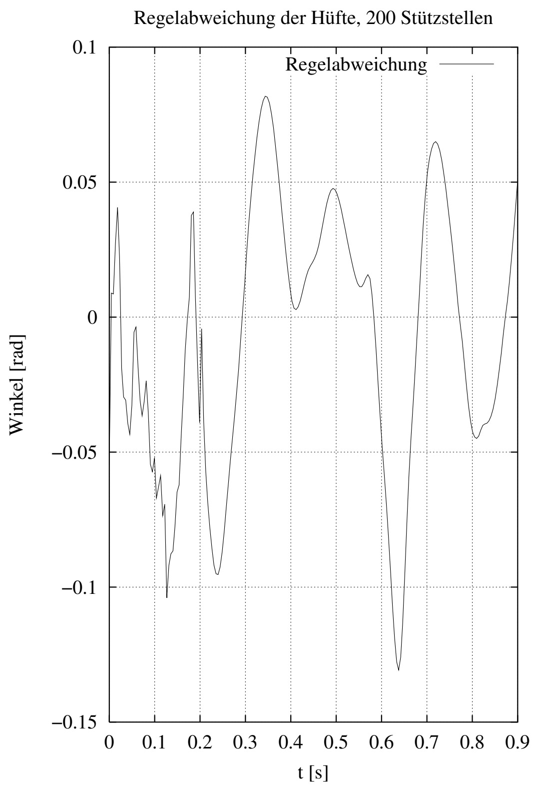


Abbildung 5.50: Regelabweichung des Hüftwinkels.

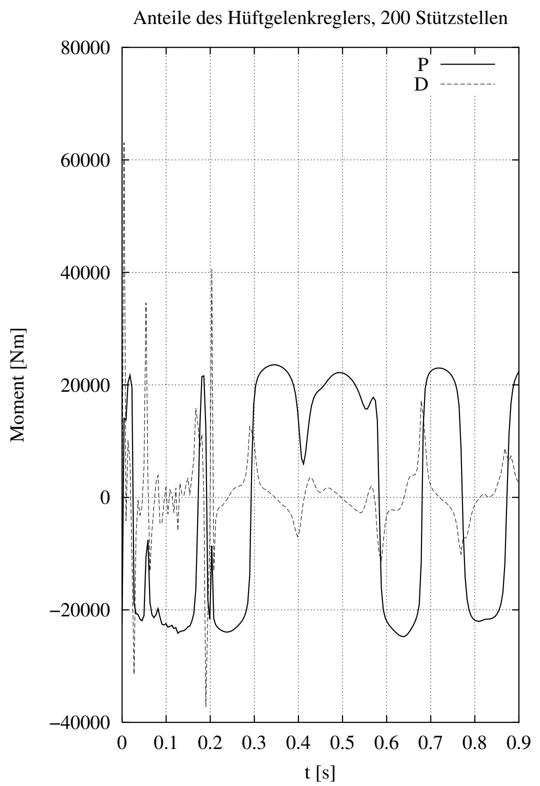


Abbildung 5.51: P- und D-Anteil des Hüftgelenkreglers.

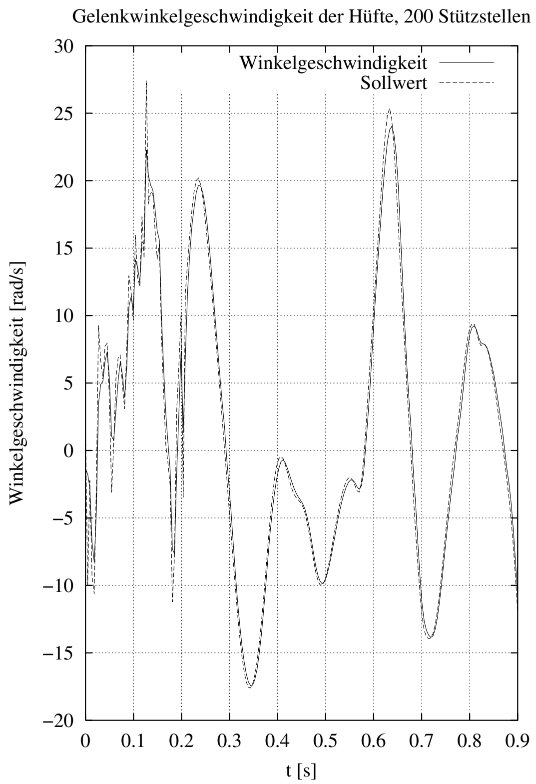


Abbildung 5.52: Gelenkwinkelgeschwindigkeit der Hüfte (durchgezogen) und der zugehörige Sollwert (gestrichelt).

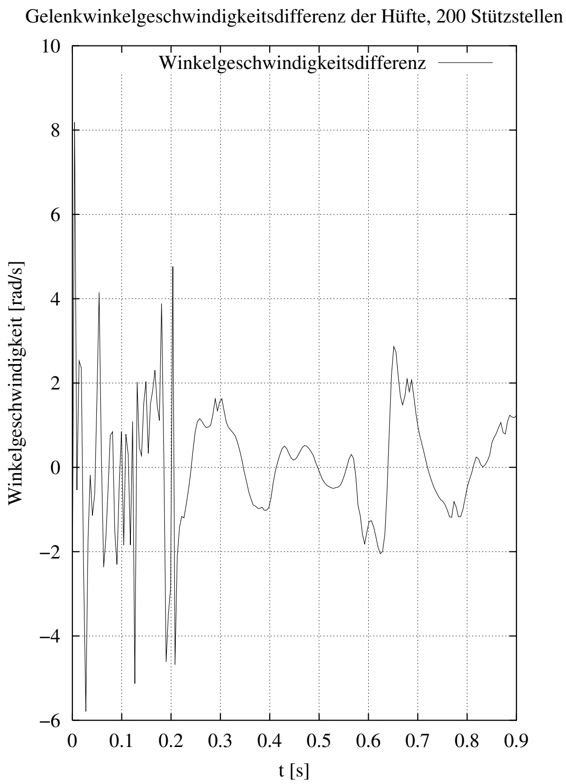


Abbildung 5.53: Differenz zwischen Ist- und Soll-Winkelgeschwindigkeit bei der Rotation des Hüftgelenks.

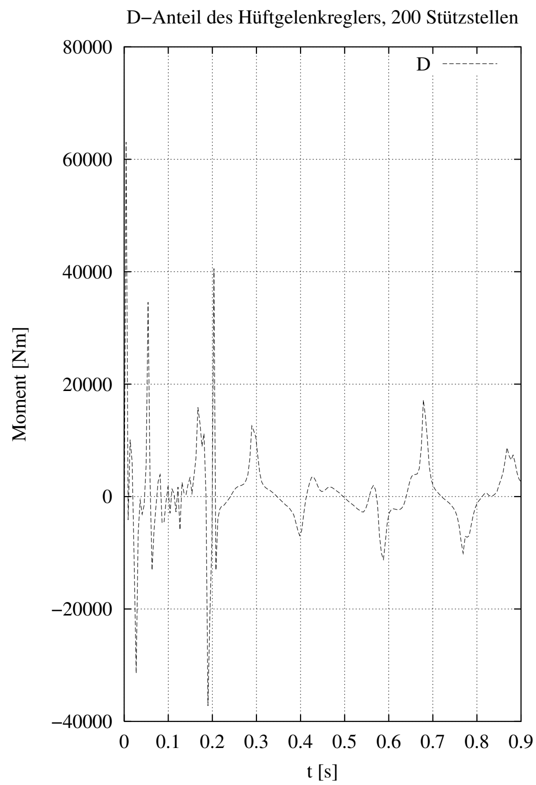


Abbildung 5.54: P- und D-Anteil des Hüftgelenkreglers.

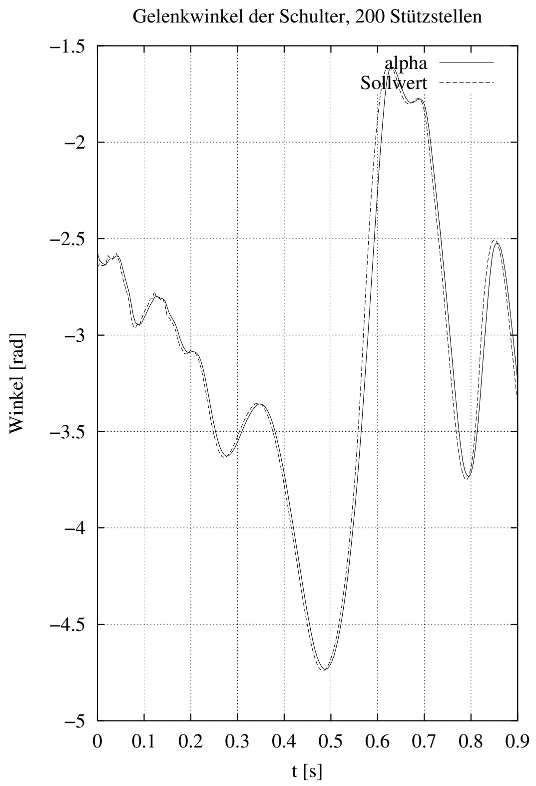


Abbildung 5.55: Gelenkwinkel der Schulter (durchgezogen) und der zugehörige Sollwert (gestrichelt).

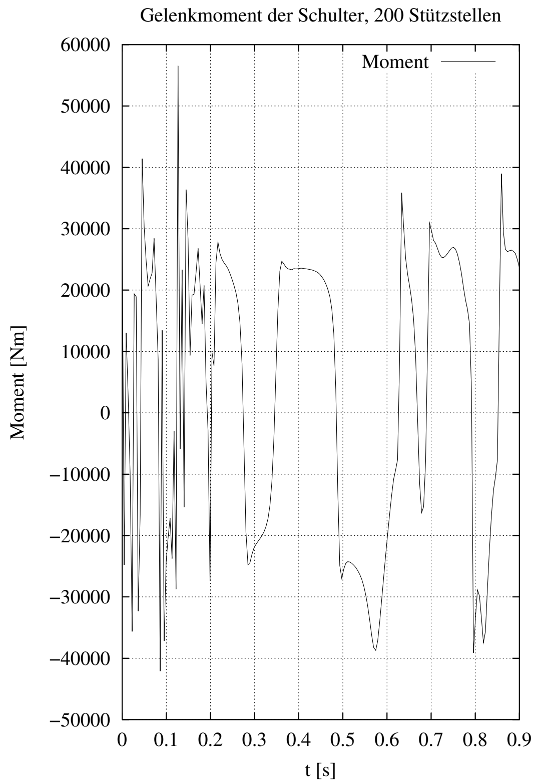


Abbildung 5.56: Moment des Reglers auf das Schultergelenk.

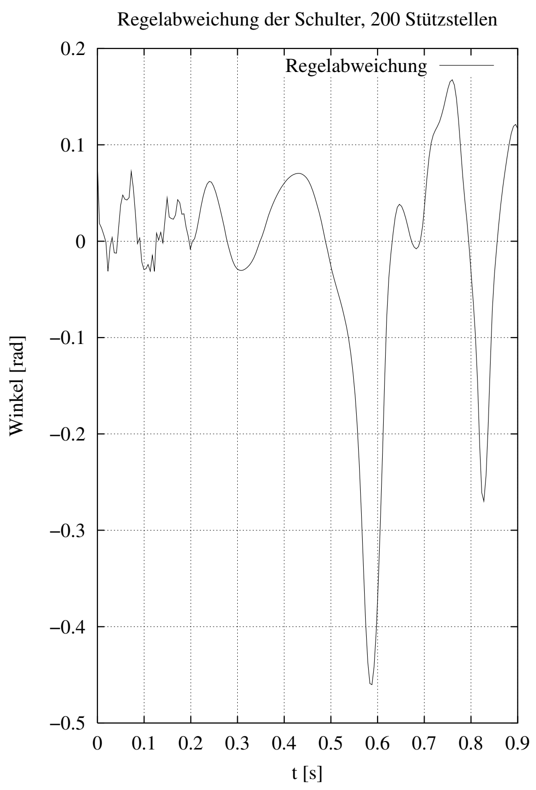


Abbildung 5.57: Regelabweichung des Schulterwinkels.

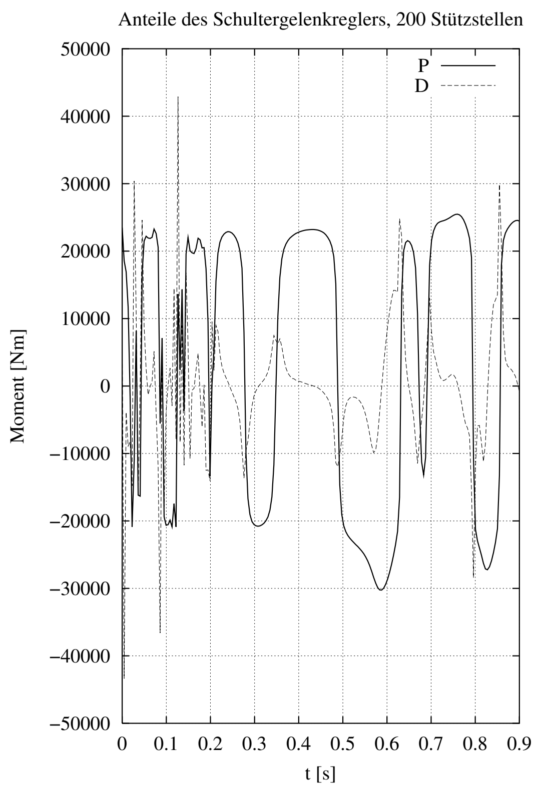


Abbildung 5.58: P- und D-Anteil des Schultergelenkreglers.

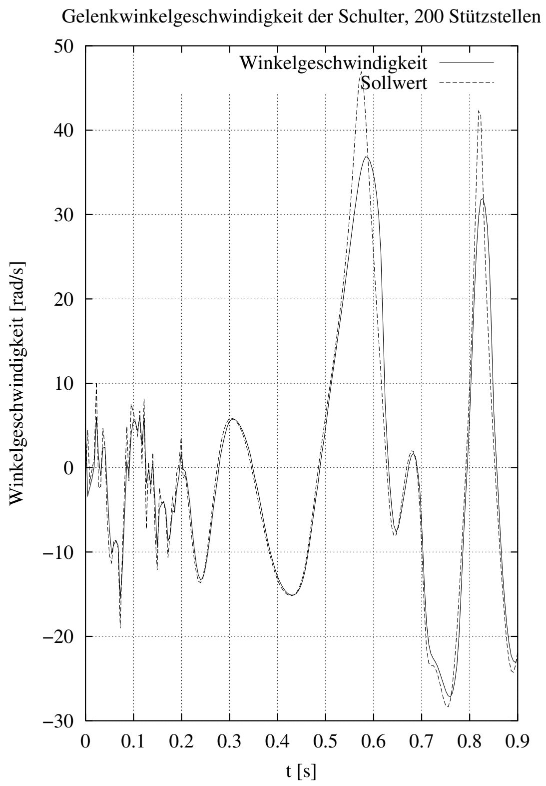


Abbildung 5.59: Gelenkwinkelgeschwindigkeit der Schulter (durchgezogen) und der zugehörige Sollwert (gestrichelt).

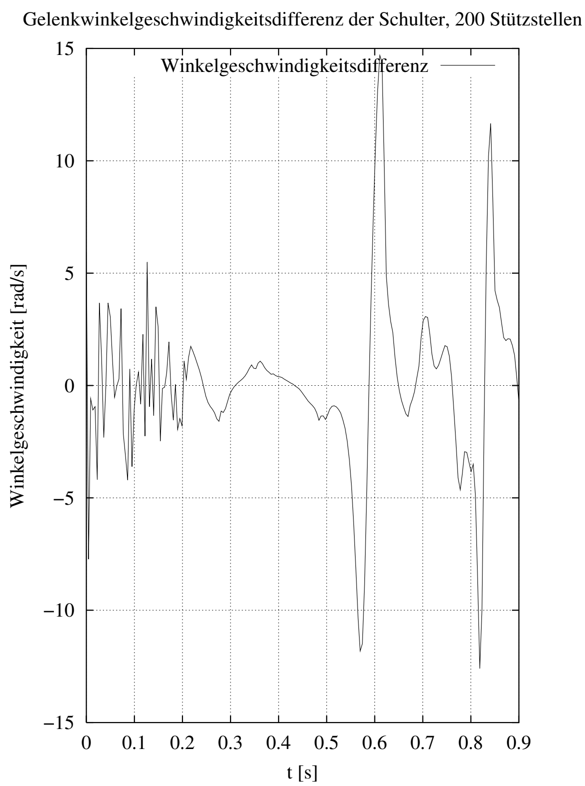


Abbildung 5.60: Differenz zwischen Ist- und Soll-Winkelgeschwindigkeit bei der Rotation der Schulter.

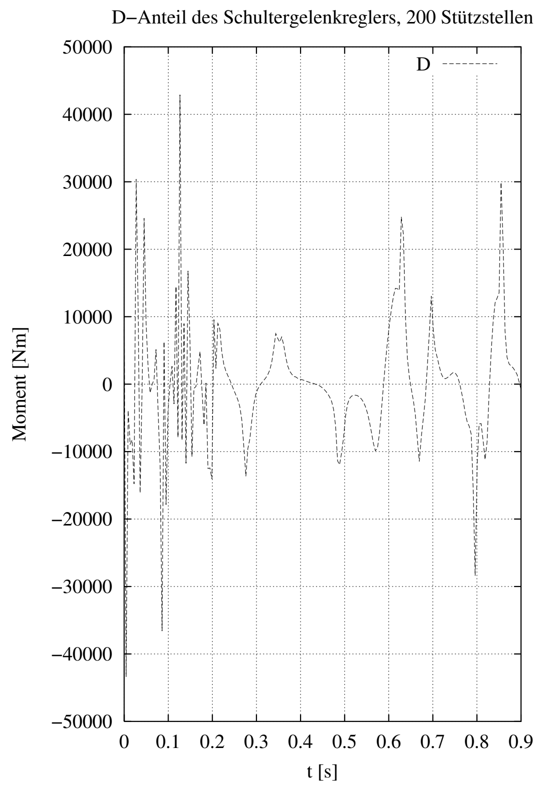


Abbildung 5.61: D-Anteil des Schultergelenkreglers.

# Kapitel 6

# Fazit und Ausblick

In der vorliegenden Arbeit wurde versucht, die Bewegung eines Flick-Flacks mit der Methode der Pseudoinversen Dynamik zu analysieren, um daraus die Gelenkmomente bestimmen zu können. Ausgehend von einer Videobildanalyse wurde ein achtgliedriges, ebenes MKS-Modell erstellt. Der Regelalgorithmus basiert auf PID-Reglern, deren Anteile mit nichtlinearen Eigenschaften versehen sind. Die so aus den Simulationen berechneten Gelenkmomente sind unphysiologisch hoch und besitzen hochfrequente Schwingungen, die von den fehlerbehafteten Messdaten auf das Modell übertragen werden. Selbst wenn die Bewegungen einer „idealen“ Vorwärtssimulation vorgegeben werden, bleiben die hohen, im Bereich von $10^4$ Nm liegenden, Momente bestehen. Der gewählte Ansatz ist instabil, er führt zu riesigen Momenten und ist somit nicht geeignet, um eine Bewegung auf Basis experimenteller, kinematischer Daten zu analysieren. Es bleibt die Frage zu klären, ob dieses Verfahren vielleicht durch eine optimierte Parameterwahl, auch bei guter Bewegungsreproduktion physiologischere Momente liefern kann. Man könnte auch einen D-Anteil (Abbildung 5.6) ausprobieren, der den Verlauf eines Doppelhöckers hat, so dass die Dämpfung nahe der Sollstellung wieder abnimmt. Es bleibt jedoch grundsätzlich fragwürdig, ob sich dieser technische Ansatz für die inversdynamische Analyse eines biologischen Systems eignen kann, da in ihm keinerlei physiologische Annahmen gemacht werden können.

```
C_____AUSLIEFERUNGSVERSION__:_s_8005
      SUBROUTINE ufel21( time     ! i  : Current Simulation Time [s]                       t
     +                 , jst      ! i  : MBS Joint State Postitions                        zg(6,ngmax)
     +                 , jstd     ! i  : MBS Joint State Velocities                        zgp(6,ngmax)
     +                 , jndim    ! i  : Max Number of Joints                              ngmax
     +                 , fst      ! i  : MBS Force States                                  zf(nzfmax,nfdmax)
     +                 , stndim   ! i  : Max Number of Force States per Element            nzfmax
     +                 , fdndim   ! i  : Max Number of Force Elements with Eigendynamics   nfdmax
     +                 , st       ! i  : (Integrated) Force States of this Element         zfk(nzfmax)
     +                 , mesm     ! i  : Measurement Matrix                                mess(3,nsmmax)
     +                 , mesmdim  ! i  : Max Number of Columns of mess                     nsmmax
     +                 , par      ! i/o: Force Parameters / Work Array of this Element     fparam(nfpmax)
     +                 , pardim   ! i  : Max Number of Force Parameters per Element        nfpmax
     +                 , ifvpdim  ! i  : Max Number of Value Pairs of Input Functions      nwpmax
     +                 , ifndim   ! i  : Max Number of Input Functions                     nfkmax
     +                 , ifpairs  ! i  : Number of Value Pairs of Input Functions          nwp(nfkmax)
     +                 , ifx      ! i  : Independent x-Values of Input Functions           ufkt(nwpmax,nfkmax)
     +                 , ify      ! i  : Dependent y-Values of Input Functions             afkt(nwpmax,nfkmax)
     +                 , ifs      ! i  : Spline Coefficients of Input Functions            sfkt(nwpmax,3,nfkmax)
     +                 , uin      ! i  : U Input Vector                                    u(numax)
     +                 , uindim   ! i  : Max Number of rows of U Vector                    numax
     +                 , rtinf    ! i/o: Root Info Vector                                  rinfo(*)
     +                 , id       ! i  : Force Element ID                                  nrf
     +                 , wv       !   o: Work Values                                       afel(nafmax)
     +                 , wvdim    ! i  : Max Number of Work Values per Element             nafmax
     +                 , std      !   o: Derivatives of Force States of this Element       zfkp(nzfmax)
     +                 , force    !   o: Force Vector acting on from-Marker Mk w.r.t. Rk   force(3)
     +                 , torque   !   o: Torque Vector acting on from-Marker Mk w.r.t. Rk  torque(3)
     +                 , *
     +                 )
************************************************************************
* ----------------------------------------------------------------------
* ------------------------- U F E L 2 1  ------------------------------
* ----------------------------------------------------------------------
*
*  State |  Name        |
* ------------------------------------------------------------------
*  Mille | Fritzsche     | Author
*  Mille | Keppler       | Co Autor
*  Mille | Mutschler     | CoCo Autor
*    -   |               |
*
* Purpose:  User defined subroutine for the computation of applied
* ========  forces and torques of force element type:
*
*           +------------------------------------------------------+
*           | Type 21: Nichtlinearer Kraftregler                   |
*           +------------------------------------------------------+
*
*           The applied forces and torques are acting between body k
*           and body l i.e. from body k to body l. The interactions
*           are resolved in the body fixed reference system Rk of body k.
*
*           Language: FORTRAN 90
*
*
* User defined Input Parameters:
* ==============================
*
*   Force States st:
*
*
```

```
*   Measurements mesm:
*
*
*   Force Parameters par:
*
*
*   Root Function Logics rtinf:
*
*
*
* User defined Output Parameters:
* ===============================
*
*   Work Values wv:
*
*
*
* General Remarks:
* ================
*
* - The kinematical measurements of system (bj) upon body j relative to
*   the system (bi) upon body i are resolved in the body fixed reference
*   system of body i.
*
* - The force element has no mass. Therefore, the cut reactions are
*   acting upon bodies k and l with different sign.
*   By this subroutine the cut reactions are computed acting at a marker
*   upon body k. They have positive sign if they are acting in the
*   direction of the normalized vector from body k to body l. This means
*   that stress is positive. The cut reactions are resolved in the
*   body fixed reference system of body k.
*
* - If only a force is caused by the force element, the vector TORQUE
*   doesn't have to be set. This holds vice versa too.
*
*
* Additional User Remarks:
* ========================
*
*
*
************************************************************************

      IMPLICIT none

C ----------------------------------------------------------------------
C Declaration of global Parameters
C ----------------------------------------------------------------------

      INCLUDE 'paramet.ins.f'   ! SIMPACK MaxDims
      INCLUDE 'vinput.ins.f'    ! c_nrf
      INCLUDE 'vpeeing.ins.f'   ! kefel
      INCLUDE 'vsysbes.ins.f'   ! fparam
      INCLUDE 'vausgan.ins.f'   ! afel

C ----------------------------------------------------------------------
C Declaration of Parameters in Parameter List
C ----------------------------------------------------------------------

      INTEGER  jndim
      INTEGER  stndim
      INTEGER  fdndim
```

```
      INTEGER  mesmdim
      INTEGER  pardim
      INTEGER  ifvpdim
      INTEGER  ifndim
      INTEGER  uindim
      INTEGER  rtinf(*)
      INTEGER  id
      INTEGER  wvdim

      DOUBLE PRECISION  time
      DOUBLE PRECISION  jst(6,jndim)
      DOUBLE PRECISION  jstd(6,jndim)
      DOUBLE PRECISION  fst(stndim,fdndim)
      DOUBLE PRECISION  st(stndim)
      DOUBLE PRECISION  mesm(3,mesmdim)
      DOUBLE PRECISION  par(pardim)
      DOUBLE PRECISION  ifpairs(ifndim)
      DOUBLE PRECISION  ifx(ifvpdim,ifndim)
      DOUBLE PRECISION  ify(ifvpdim,ifndim)
      DOUBLE PRECISION  ifs(ifvpdim,3,ifndim)
      DOUBLE PRECISION  uin(uindim)
      DOUBLE PRECISION  wv(wvdim)
      DOUBLE PRECISION  std(stndim)
      DOUBLE PRECISION  force(3)
      DOUBLE PRECISION  torque(3)

C ----------------------------------------------------------------------
C Declaration of Functions
C ----------------------------------------------------------------------

      DOUBLE PRECISION  inpfct  ! Evaluation of Input Functions y(x):
      EXTERNAL          inpfct  ! y = inpfct( InpFctID , x )
C
C ----------------------------------------------------------------------
C Declaration of Local Variables
C ----------------------------------------------------------------------

      INTEGER  i,j,InpFctID(3), offset9



      DOUBLE PRECISION  zero, eins,
     +      work(3),mywk1(3)
     +  ,ra(6) ,va(6), mywk2(3)
     +     ,cosbe, cosga, cosal , al ,be , ga ,sinal,sinbe,singa
     +     ,Eingabe(20),R_rel(3),V_rel(3),Phi_rel(3),PPhi_rel(3)
     +     ,Plotwert(20),ableit(3), dt,
     +      time_abschalt, time_2_abschalt


      PARAMETER( zero = 0.0d+00 , eins = 1.0d+00)

DOUBLE PRECISION  inpfkt(3),abstand(3),
     +      pconst, patansteig, pregler(3),pdatan(3), dnl(3), dconst,
     +      dgrunddaempf, dbreite, iconst, tau, ivergessen, iatansteig,
     +      idatan(3), Arbeit(3), AAQ(3)
C ----------------------------------------------------------------------
C Defaults
C ----------------------------------------------------------------------
 offset9 = 98
c       par(offset9-7/-9)gesavete Inputfunktionen
c       par(offset9-19)  gesavete Zeit
```

```
c       par(offset9-20)  die nächsten freien
c   Initialisierung !  muss relativ am Anfang stehen und wird nur beim allerersten Aufruf durchlaufen

IF( int( par(offset9-0) ) .EQ. 0 ) THEN
par(offset9-0) = 1.0 !  muß 1 sein
par(offset9-1) = 0.0
par(offset9-2) = 0.0
par(offset9-3) = 0.0
par(offset9-4) = 0.0
par(offset9-5) = 0.0
par(offset9-6) = 0.0
par(offset9-7) = 0.0
par(offset9-8) = 0.0
par(offset9-9) = 0.0
par(offset9-10) = 0.0
par(offset9-11) = 0.0
par(offset9-12) = 0.0
par(offset9-13) = 0.0
par(offset9-14) = 0.0
par(offset9-15) = 0.0
par(offset9-16) = 0.0
par(offset9-17) = 0.0
par(offset9-18) = 0.0
par(offset9-19) = 0.0
par(offset9-20) = 0.0

ENDIF


      force(1)  = zero
      force(2)  = zero
      force(3)  = zero

      torque(1) = zero
      torque(2) = zero
      torque(3) = zero

      do i=1, wvdim
         wv(i) = zero
      end do




C -----------------------------------------------------------------------
C Belegung
C -----------------------------------------------------------------------

C Maximalmoment = PAR()



pconst = 1.0*PAR(5)
patansteig = PAR(6)

dconst = PAR(7)
      dgrunddaempf = PAR(8)
      dbreite = PAR(9)

iconst = PAR(10)
tau =  PAR(11)
```

```
      ivergessen   =  PAR(12)
iatansteig =  PAR(13)

      if(  tau < 0.000000001 ) THEN
  print*,'Tierischer Fehler tau fast 0'
  tau = 0.3
ENDIF




c do i = 1,3
c
c abstand(i) = inpfkt(i) - ra(i)
c
c pdatan(i)= datan(patansteig*abstand(i))
c pregler(i) = pconst*(abstand(i)+pdatan(i))
c
c dnl(i) = dconst * (1.0 / ((abstand(i)*abstand(i))/dbreite + 1.0) +
c     +                  dgrunddaempf)
c
c
c         idatan(3) = datan(iatansteig*abstand(i))
c     std(i) = (abstand(i) + idatan(3) ) -
c     +       ivergessen * dabs(va(i)) * st(i+3)    ! - 0.6 * st(1)
c
c     std(i+3) = - (st(i+3))/tau  + st(i)
c
c      end do
c
c
c do i = 1,3
c
c force(i) = pregler(i) + dnl(i)*va(i) + iconst*st(i)
c
c     torque(i) = pregler(i+3) + dnl(i+3)*va(i+3) + iconst*st(i+3)
c
c     end do



C ----------------------------------------------------------------------
C Computations
C ----------------------------------------------------------------------

      DO  i = 1,3
                  ra(i) = mesm(1,i) * mesm(1,7) + mesm(2,i) * mesm(2,7)
     +                   + mesm(3,i) * mesm(3,7)
ENDDO





sinbe =   mesm(1,6)
      cosbe =   DSQRT( eins - sinbe* sinbe )
    singa = - mesm(1,5) / cosbe
          cosga =   mesm(1,4) / cosbe
          cosal =   mesm(3,6) / cosbe
          sinal = - mesm(2,6) / cosbe
          al    =   ATAN2( sinal , cosal )
          be    =   ATAN2( sinbe , cosbe )
```

```
          ga    =   ATAN2( singa , cosga )
    ra(4)= al
ra(5)= be
ra(6)= ga

C     Soll den RelativGeschwindigkeitsvektor darstellen im System Ki
      DO i = 1,3
                 va(i)  = mesm(1,i) * mesm(1,8) + mesm(2,i) * mesm(2,8)
     +                   + mesm(3,i) * mesm(3,8)
                 va(i+3)= mesm(1,i) * mesm(1,9) + mesm(2,i) * mesm(2,9)
     +                   + mesm(3,i) * mesm(3,9)
      ENDDO




      do i = 1,3
InpFctID(i)= NINT( PAR(i) )
inpfkt(i) = inpfct( InpFctID(i) , time )
end do



      dt = par(offset9-19) - time
      do i = 1,3

  if( dabs( dt ) > 1E-12  ) then
c          ableit(i) =( inpfct( InpFctID(i) , time + dt) - inpfkt(i) ) / dt
          ableit(i) = ( par(offset9-6-i) - inpfkt(i) ) / dt
  else
          ableit(i) = 0.0
  endif
end do






do i = 1,3

abstand(i) = ra(i) - inpfkt(i)


pdatan(i)= datan(patansteig*abstand(i))
pregler(i) = pconst*(abstand(i)+pdatan(i))

dnl(i) = dconst * (1.0 / ((abstand(i)*abstand(i))/dbreite + 1.0) +
     +                dgrunddaempf)


          idatan(i) = datan(iatansteig*abstand(i))
    std(i) = (abstand(i) + idatan(i) ) -
     +       ivergessen * dabs(va(i)-ableit(i)) * st(i+3)    ! - 0.6 * st(1)

    std(i+3) = - (st(i+3))/tau  + st(i)
      end do

      INCLUDE 'time1.inc.f'

do i = 1,3
```

```
          if ( time > time_2_abschalt) then
 force(i) = 0.0
    else
 force(i) = (pregler(i) + dnl(i)*(va(i)-ableit(i)) + iconst*st(i))
    end if

C     torque(i) = (pregler(i+3) + dnl(i+3)*va(i+3) + iconst*st(i+3))
c          print*,'arg',i,'  : ',inpfkt(i) ,ra(i), abstand(i)
std(i+6) = force(i)*va(i)
    Arbeit(i) = st(i+6)

    AAQ(i) = ABS(abstand(i))
      end do



****************************************************
******* GOT ALL DATAS ....     **********
****************************************************

R_rel(1)    = ra(1)
R_rel(2)    = ra(2)
R_rel(3)    = ra(3)
V_rel(1)    = va(1)
V_rel(2)    = va(2)
V_rel(3)    = va(3)
Phi_rel(1)  = ra(4)
Phi_rel(2)  = ra(5)
Phi_rel(3)  = ra(6)
PPhi_rel(1) = va(4)
PPhi_rel(2) = va(5)
PPhi_rel(3) = va(6)
********************************************************************************
********************************************************************************
******* TEMPLATE : DERIVES DATA FROM SIMPACK-MATRICES **********
******* END      **********
********************************************************************************
********************************************************************************
*
*
***************************************************************************************************
* USERPART !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
***************************************************************************************************
C ----------------------------------------------------------------------

    wv(1)=ra(1)
wv(2)=ra(2)
wv(3)=ra(3)
c wv(4)=ra(4)
c wv(5)=ra(5)
c wv(6)=ra(6)
wv(7)=va(1)
wv(8)=va(2)
wv(9)=va(3)
c wv(10)=va(4)
c wv(11)=va(5)
c wv(12)=va(6)
wv(13)=inpfkt(1)
wv(14)=inpfkt(2)
wv(15)=inpfkt(3)
wv(16)=ableit(1)
```

```
wv(17)=ableit(2)
wv(18)=ableit(3)
c wv(19)=0
      wv(20)=abstand(1)    ! Regelabweichung
wv(21)=abstand(2)    ! Regelabweichung
wv(22)=abstand(3)    ! Regelabweichung
c wv(23)=abstand(4)    ! Regelabweichung
c wv(24)=abstand(5)    ! Regelabweichung
c wv(25)=abstand(6)    ! Regelabweichung
wv(26)=force(1)
wv(27)=force(2)
wv(28)=force(3)
wv(29)=pregler(1)
wv(30)=pregler(2)
wv(31)=pregler(3)
wv(32)=dnl(1)*(va(1)-ableit(1))
wv(33)=dnl(2)*(va(2)-ableit(2))
wv(34)=dnl(3)*(va(3)-ableit(3))
wv(35)=iconst*st(1)
wv(36)=iconst*st(2)
wv(37)=iconst*st(3)
c wv(38)=torque(1)
c wv(39)=torque(2)
c wv(40)=torque(3)
c wv(41)=pregler(4)
c wv(42)=pregler(5)
c wv(43)=pregler(6)
c wv(44)=dnl(4)*va(4)
c wv(45)=dnl(5)*va(5)
c wv(46)=dnl(6)*va(6)
c     wv(47)=iconst*st(4)
c wv(48)=iconst*st(5)
c     wv(49)=iconst*st(6)
wv(50)=Arbeit(1)
wv(51)=Arbeit(2)
wv(52)=Arbeit(3)
c wv(53)=Arbeit(4)
c wv(54)=Arbeit(5)
c wv(55)=Arbeit(6)
wv(56)=AAQ(1)
wv(57)=AAQ(2)
wv(58)=AAQ(3)
c wv(59)=AAQ(4)
c wv(60)=AAQ(5)
c wv(61)=AAQ(6)


mywk1(1) = force(1)
mywk1(2) = force(2)
mywk1(3) = force(3)
mywk2(1) = torque(1)
mywk2(2) = torque(2)
mywk2(3) = torque(3)

torque(1) = zero
torque(2) = zero
torque(3) = zero
force(1) = zero
force(2) = zero
force(3) = zero

    DO i = 1,3
```

```
  DO  j = 1,3
force(i)  = force(i)  + mesm(i,j) * mywk1(j)
torque(i) = torque(i) + mesm(i,j) * mywk2(j)
      ENDDO
    ENDDO


c Diese Werte werden beim nächsten Durchlauf gebraucht! Deshalb hier am Schluß
          par(offset9-7) = inpfkt(1)
          par(offset9-8) = inpfkt(2)
          par(offset9-9) = inpfkt(3)
          par(offset9-19) = time


c      continue


c      par(offset9-1) = par(offset9-1) + 1
c  if( int( par(offset9-1) ) > 20 ) THEN
c   par(offset9-1) = 0
c      print*,' '
c     DO i = 1,6
c      print*,'arg',i,'  : ',ra(i), abstand(i)
c ENDDO
c       print*,' '
c      endif

C ----------------------------------------------------------------------
C Return and Error Handling
C ----------------------------------------------------------------------

      RETURN

C Warning - continue Calculations
 9100 continue
      print*, '** Warning by User Force Element of Type 21'
      print*, '   ', trim(c_nrf(id))  ! Force Element Name
      print*, '   Unknown Warning'    ! Insert Description
      return

C Error - stop Calculations
 9200 continue
      print*, '** ERROR in User Force Element of Type 21'
      print*, '   ', trim(c_nrf(id))  ! Force Element Name
      print*, '   Unknown Error'      ! Insert Description
      return 1

      END


C_____AUSLIEFERUNGSVERSION__:_s_8005
      SUBROUTINE ufel22( time     ! i  : Current Simulation Time [s]                       t
     +                 , jst      ! i  : MBS Joint State Postitions                        zg(6,ngmax)
     +                 , jstd     ! i  : MBS Joint State Velocities                        zgp(6,ngmax)
     +                 , jndim    ! i  : Max Number of Joints                              ngmax
     +                 , fst      ! i  : MBS Force States                                  zf(nzfmax,nfdmax)
     +                 , stndim   ! i  : Max Number of Force States per Element            nzfmax
     +                 , fdndim   ! i  : Max Number of Force Elements with Eigendynamics   nfdmax
     +                 , st       ! i  : (Integrated) Force States of this Element         zfk(nzfmax)
     +                 , mesm     ! i  : Measurement Matrix                                mess(3,nsmmax)
     +                 , mesmdim  ! i  : Max Number of Columns of mess                     nsmmax
     +                 , par      ! i/o: Force Parameters / Work Array of this Element     fparam(nfpmax)
```

```
     +                , pardim   ! i  : Max Number of Force Parameters per Element       nfpmax
     +                , ifvpdim  ! i  : Max Number of Value Pairs of Input Functions     nwpmax
     +                , ifndim   ! i  : Max Number of Input Functions                    nfkmax
     +                , ifpairs  ! i  : Number of Value Pairs of Input Functions         nwp(nfkmax)
     +                , ifx      ! i  : Independent x-Values of Input Functions          ufkt(nwpmax,nfkmax)
     +                , ify      ! i  : Dependent y-Values of Input Functions            afkt(nwpmax,nfkmax)
     +                , ifs      ! i  : Spline Coefficients of Input Functions           sfkt(nwpmax,3,nfkmax)
     +                , uin      ! i  : U Input Vector                                   u(numax)
     +                , uindim   ! i  : Max Number of rows of U Vector                   numax
     +                , rtinf    ! i/o: Root Info Vector                                 rinfo(*)
     +                , id       ! i  : Force Element ID                                 nrf
     +                , wv       !   o: Work Values                                      afel(nafmax)
     +                , wvdim    ! i  : Max Number of Work Values per Element            nafmax
     +                , std      !   o: Derivatives of Force States of this Element      zfkp(nzfmax)
     +                , force    !   o: Force Vector acting on from-Marker Mk w.r.t. Rk  force(3)
     +                , torque   !   o: Torque Vector acting on from-Marker Mk w.r.t. Rk torque(3)
     +                , *
     +                )
************************************************************************
* ----------------------------------------------------------------------
* ------------------------- U F E L 2 0 ------------------------------
* ----------------------------------------------------------------------
*
*  State |  Name        |
* ------------------------------------------------------------------
*  Mille | Fritzsche     | Author
*  Mille | Keppler       | Co Autor
*  Mille | Mutschler     | CoCo Autor
*    -   |               |
*
* Purpose:  User defined subroutine for the computation of applied
* ========  forces and torques of force element type:
*
*          +-------------------------------------------------------+
*          | Type 22: Nichtlinearer Momentenregler                  |
*          +-------------------------------------------------------+
*
*          The applied forces and torques are acting between body k
*          and body l i.e. from body k to body l. The interactions
*          are resolved in the body fixed reference system Rk of body k.
*
*          Language: FORTRAN 90
*
*
* User defined Input Parameters:
* ==============================
*
*   Force States st:
*
*
*   Measurements mesm:
*
*
*   Force Parameters par:
*
*
*   Root Function Logics rtinf:
*
*
*
* User defined Output Parameters:
* ===============================
```

```
*
*   Work Values wv:
*
*
*
* General Remarks:
* ================
*
* - The kinematical measurements of system (bj) upon body j relative to
*   the system (bi) upon body i are resolved in the body fixed reference
*   system of body i.
*
* - The force element has no mass. Therefore, the cut reactions are
*   acting upon bodies k and l with different sign.
*   By this subroutine the cut reactions are computed acting at a marker
*   upon body k. They have positive sign if they are acting in the
*   direction of the normalized vector from body k to body l. This means
*   that stress is positive. The cut reactions are resolved in the
*   body fixed reference system of body k.
*
* - If only a force is caused by the force element, the vector TORQUE
*   doesn't have to be set. This holds vice versa too.
*
*
* Additional User Remarks:
* ========================
*
*
*
************************************************************************

      IMPLICIT none

C -----------------------------------------------------------------------
C Declaration of global Parameters
C -----------------------------------------------------------------------

      INCLUDE 'paramet.ins.f'   ! SIMPACK MaxDims
      INCLUDE 'vinput.ins.f'    ! c_nrf
      INCLUDE 'vpeeing.ins.f'   ! kefel
      INCLUDE 'vsysbes.ins.f'   ! fparam
      INCLUDE 'vausgan.ins.f'   ! afel

C -----------------------------------------------------------------------
C Declaration of Parameters in Parameter List
C -----------------------------------------------------------------------

      INTEGER  jndim
      INTEGER  stndim
      INTEGER  fdndim
      INTEGER  mesmdim
      INTEGER  pardim
      INTEGER  ifvpdim
      INTEGER  ifndim
      INTEGER  uindim
      INTEGER  rtinf(*)
      INTEGER  id
      INTEGER  wvdim

      DOUBLE PRECISION  time
      DOUBLE PRECISION  jst(6,jndim)
      DOUBLE PRECISION  jstd(6,jndim)
```

```
      DOUBLE PRECISION  fst(stndim,fdndim)
      DOUBLE PRECISION  st(stndim)
      DOUBLE PRECISION  mesm(3,mesmdim)
      DOUBLE PRECISION  par(pardim)
      DOUBLE PRECISION  ifpairs(ifndim)
      DOUBLE PRECISION  ifx(ifvpdim,ifndim)
      DOUBLE PRECISION  ify(ifvpdim,ifndim)
      DOUBLE PRECISION  ifs(ifvpdim,3,ifndim)
      DOUBLE PRECISION  uin(uindim)
      DOUBLE PRECISION  wv(wvdim)
      DOUBLE PRECISION  std(stndim)
      DOUBLE PRECISION  force(3)
      DOUBLE PRECISION  torque(3)

C ----------------------------------------------------------------------
C Declaration of Functions
C ----------------------------------------------------------------------

      DOUBLE PRECISION  inpfct  ! Evaluation of Input Functions y(x):
      EXTERNAL          inpfct  ! y = inpfct( InpFctID , x )

C ----------------------------------------------------------------------
C Declaration of Local Variables
C ----------------------------------------------------------------------

      INTEGER  i,j,InpFctID(3), offset9



      DOUBLE PRECISION  zero, eins, mein_PI,
     +      work(3),mywk1(3)
     +  ,ra(6) ,va(6), mywk2(3)
     +     ,cosbe, cosga, cosal , al ,be , ga ,sinal,sinbe,singa
     +     ,Eingabe(20),R_rel(3),V_rel(3),Phi_rel(3),PPhi_rel(3)
     +     ,Plotwert(20),hilfs_1vektor(3), hilfs_2vektor(3)
     +     ,ableit(3), dt

      PARAMETER( zero = 0.0d+00 , eins = 1.0d+00)

DOUBLE PRECISION  inpfkt(3),abstand(3),
     +      pconst, patansteig, pregler(3),pdatan(3), dnl(3), dconst,
     +      dgrunddaempf, dbreite, iconst, tau, ivergessen, iatansteig,
     +      idatan(3), Arbeit(3), AAQ(3), maxsteig, Maximalmoment,
     +      time_abschalt, time_2_abschalt


C ----------------------------------------------------------------------
C Defaults
C ----------------------------------------------------------------------

      mein_PI = 2.0 * dacos(0.0d0)
 offset9 = 98
c       par(offset9-0)   Init ist beim start 0, muss dafür sorgen dass variablen initialisiert werden
c       par(offset9-1)   Zähler, wie oft dieses Forzeelement aufgerufen wird
c
c       par(offset9-4)   Ab hier die Zahlen fürs durchdrehen ( 6 stück ) Die im letzten Durchlauf berechneten Werte
c       par(offset9-10)  Ab hier die Zahlen fürs durchdrehen ( 6 stück ) Vielfache von 2-Pi Sprüngen
c       par(offset9-16)  Ab hier die gespeicherten Werte für hilfs_2vektor (3 Stück)
c       par(offset9-19)  gesavete Zeit
c       par(offset9-20)  die nächsten freien
c   Initialisierung !  muss relativ am Anfang stehen und wird nur beim allerersten Aufruf durchlaufen
```

```
IF( int( par(offset9-0) ) .EQ. 0 ) THEN
par(offset9-0) = 1.0 !  muß 1 sein
par(offset9-1) = 0.0
par(offset9-2) = 0.0
par(offset9-3) = 0.0
par(offset9-4) = 0.0
par(offset9-5) = 0.0
par(offset9-6) = 0.0
par(offset9-7) = 0.0
par(offset9-8) = 0.0
par(offset9-9) = 0.0
par(offset9-10) = 0.0
par(offset9-11) = 0.0
par(offset9-12) = 0.0
par(offset9-13) = 0.0
par(offset9-14) = 0.0
par(offset9-15) = 0.0
par(offset9-16) = 0.0
par(offset9-17) = 0.0
par(offset9-18) = 0.0
par(offset9-19) = 0.0
par(offset9-20) = 0.0

ENDIF

      force(1)  = zero
      force(2)  = zero
      force(3)  = zero

      torque(1) = zero
      torque(2) = zero
      torque(3) = zero

      do i=1, wvdim
         wv(i) = zero
      end do



C ----------------------------------------------------------------------
C Belegung
C ----------------------------------------------------------------------



pconst = 1.0*PAR(5)
patansteig = PAR(6)

dconst = PAR(7)
      dgrunddaempf = PAR(8)
      dbreite = PAR(9)

iconst = PAR(10)
tau =  PAR(11)
      ivergessen   =  PAR(12)
iatansteig =  PAR(13)


      if(  tau < 0.000000001 ) THEN
  print*,'Tierischer Fehler tau fast 0'
  tau = 0.3
```

```
ENDIF

Maximalmoment = PAR(20)
      maxsteig = PAR(21)



c do i = 1,6
c
c abstand(i) = inpfkt(i) - ra(i)
c
c pdatan(i)= datan(patansteig*abstand(i))
c pregler(i) = pconst*(abstand(i)+pdatan(i))
c
c dnl(i) = dconst * (1.0 / ((abstand(i)*abstand(i))/dbreite + 1.0) +
c     +                 dgrunddaempf)
c
c
c         idatan(6) = datan(iatansteig*abstand(i))
c     std(i) = (abstand(i) + idatan(6) ) -
c     +       ivergessen * dabs(va(i)) * st(i+6)    ! - 0.6 * st(1)
c
c     std(i+6) = - (st(i+6))/tau  + st(i)
c
c      end do
c
c
c do i = 1,3
c
c force(i) = pregler(i) + dnl(i)*va(i) + iconst*st(i)
c
c     torque(i) = pregler(i+3) + dnl(i+3)*va(i+3) + iconst*st(i+3)
c
c     end do



C ----------------------------------------------------------------------
C Computations
C ----------------------------------------------------------------------

      DO  i = 1,3
                  ra(i) = mesm(1,i) * mesm(1,7) + mesm(2,i) * mesm(2,7)
     +                   + mesm(3,i) * mesm(3,7)
ENDDO





sinbe =   mesm(1,6)
      cosbe =   DSQRT( eins - sinbe* sinbe )
    singa = - mesm(1,5) / cosbe
          cosga =   mesm(1,4) / cosbe
          cosal =   mesm(3,6) / cosbe
          sinal = - mesm(2,6) / cosbe
          al    =   ATAN2( sinal , cosal )
          be    =   ATAN2( sinbe , cosbe )
          ga    =   ATAN2( singa , cosga )
    ra(4)= al
ra(5)= be
ra(6)= ga
```

```
C     Soll den RelativGeschwindigkeitsvektor darstellen im System Ki
      DO i = 1,3
                 va(i)  = mesm(1,i) * mesm(1,8) + mesm(2,i) * mesm(2,8)
     +                   + mesm(3,i) * mesm(3,8)
                 va(i+3)= mesm(1,i) * mesm(1,9) + mesm(2,i) * mesm(2,9)
     +                   + mesm(3,i) * mesm(3,9)
      ENDDO


c par(offset9-4) = al
c        par(offset9-5) = be
c         par(offset9-6) = ga
c         par(offset9-7) = inpfkt(1)
c         par(offset9-8) = inpfkt(2)
c         par(offset9-9) = inpfkt(3)
c         par(offset9-10) = durchdreh von al

c      i=1
c  InpFctID(i)= NINT( PAR(i) )
c inpfkt(i) = inpfct( InpFctID(i) , time )

c      call icsevu_3(InpFctID
c     *            , ifx(1, ifndim)
c     *            , ify(1, ifndim)
c     *            , ifvpdim(ifndim)
c     *            , sfkt(1, 1, ifndim)
c     *            , ifvpdim
c     *            , r_x_mm
c     *            , f
c     *            , 1
c     *            , ierr)
c
c Parameterliste von icsevu_3
c id_if ................0 Id of input function
c       ufkt ................ values of undependant variables       (do not change)
c       afkt ................ values of dependant variables         (do not change)
c       nwp ................. number of value doubles               (do not change)
c       sfkt ................ spline coefficients for interpolation (do not change)
c       nwpmax .............. field dimension                       (do not change)
c       r(1) ................ independant variable
c       f ................... dependant variable
c       1 ................... dimension of r_x_mm and f             (do not change)
c       ierr ................ error code





      do i = 1,3
InpFctID(i)= NINT( PAR(i) )
inpfkt(i) = inpfct( InpFctID(i) , time )
end do


DO i = 1,3

  if( ra(i+3) * par(offset9 - 3 - i) < 0.0   ) then   ! Vorzeichenwechsel ??? zwischen alten und neuen
     if( dabs( ra(i+3) - par(offset9 - 3 - i) ) > 4.71238898) then   ! Sprung wirklich akzeptiert
       if(  ra(i+3) < 0.0 ) then   ! Sprung wirklich akzeptiert
                par(offset9 - 9 - i) = par(offset9 - 9 - i) + 2.0 * mein_PI
       else
```

```
                par(offset9 - 9 - i) = par(offset9 - 9 - i) - 2.0 * mein_PI
       endif

     else

   if( dabs( ra(i+3) - par(offset9 - 3 - i) ) > 2.99999 ) then   ! Sprung wirklich akzeptiert
         if(  ra(i+3) < 0.0 ) then   ! Sprung wirklich akzeptiert
                 par(offset9 - 9 - i) = par(offset9 - 9 - i) +  mein_PI
         else
                 par(offset9 - 9 - i) = par(offset9 - 9 - i) -  mein_PI
         endif

       endif

 endif

  endif
        hilfs_1vektor(i) = ra(i+3) + par(offset9 - 9 - i)

  if( inpfkt(i) * par(offset9 - 6 - i) < 0.0   ) then   ! Vorzeichenwechsel ??? zwischen alten und neuen
     if( dabs( inpfkt(i) - par(offset9 - 6 - i) ) > 4.71238898) then   ! Sprung wirklich akzeptiert
       if(  inpfkt(i) < 0.0 ) then   ! Sprung wirklich akzeptiert
                par(offset9 - 12 - i) = par(offset9 - 12 - i) + 2.0 * mein_PI
       else
                par(offset9 - 12 - i) = par(offset9 - 12 - i) - 2.0 * mein_PI
       endif

     endif
  endif
        hilfs_2vektor(i) = inpfkt(i) + par(offset9 - 12 - i)

ENDDO



      dt = par(offset9-19) - time
      do i = 1,3

  if( dabs( dt ) > 1E-12  ) then
c          ableit(i) =( inpfct( InpFctID(i) , time + dt) - inpfkt(i) ) / dt
          ableit(i) = ( par(offset9-15-i) - hilfs_2vektor(i) ) / dt
  else
          ableit(i) = 0.0
  endif
end do





do i = 1,3

abstand(i) = hilfs_1vektor(i) - hilfs_2vektor(i)
c abstand(i) = ra(i+3) - inpfkt(i)

pdatan(i)= datan(patansteig*abstand(i))
pregler(i) = pconst*(abstand(i)+pdatan(i))

dnl(i) = dconst * (1.0 / ((abstand(i)*abstand(i))/dbreite + 1.0) +
     +                dgrunddaempf)
```

```
          idatan(i) = datan(iatansteig*abstand(i))
    std(i) = (abstand(i) + idatan(i) ) -
     +       ivergessen * dabs( (va(i+3)-ableit(i))  ) * st(i+3)    ! - 0.6 * st(1)

    std(i+3) = - (st(i+3))/tau  + st(i)
      end do

      INCLUDE 'time1.inc.f'


    do i = 1,3

C force(i)  = (pregler(i) + dnl(i)*va(i) + iconst*st(i))

          if ( time > time_abschalt) then
    torque(i) = 0.0
    else
           if ( time > time_2_abschalt) then
      torque(i) = pregler(i) + dnl(i)*(va(i+3)-ableit(i))
     else
      torque(i) = pregler(i) + dnl(i)*(va(i+3)-ableit(i)) + iconst*st(i)

 end if

end if

C     torque(i) = Maximalmoment*(  2/mein_PI * DATAN( maxsteig*
C     +        (pregler(i) + dnl(i)*(va(i+3)-ableit(i)) + iconst*st(i)) )  )

std(i+6) = torque(i)*va(i+3)
    Arbeit(i) = st(i+6)

AAQ(i) = ABS(abstand(i))

end do



****************************************************
******* GOT ALL DATAS ....     **********
****************************************************

R_rel(1)    = ra(1)
R_rel(2)    = ra(2)
R_rel(3)    = ra(3)
V_rel(1)    = va(1)
V_rel(2)    = va(2)
V_rel(3)    = va(3)
Phi_rel(1)  = ra(4)
Phi_rel(2)  = ra(5)
Phi_rel(3)  = ra(6)
PPhi_rel(1) = va(4)
PPhi_rel(2) = va(5)
PPhi_rel(3) = va(6)
********************************************************************************
********************************************************************************
******* TEMPLATE : DERIVES DATA FROM SIMPACK-MATRICES **********
******* END      **********
********************************************************************************
********************************************************************************
*
```

```
*
*************************************************************************************************
* USERPART !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
*************************************************************************************************
C ----------------------------------------------------------------------

          wv(1)=hilfs_1vektor(1)
          wv(2)=hilfs_2vektor(1)

wv(4)=ra(4)
wv(5)=ra(5)
wv(6)=ra(6)
c wv(7)=va(1)
c wv(8)=va(2)
c wv(9)=va(3)
wv(10)=va(4)
wv(11)=va(5)
wv(12)=va(6)
wv(13)=inpfkt(1)
wv(14)=inpfkt(2)
wv(15)=inpfkt(3)
wv(16)=ableit(1)
wv(17)=ableit(2)
wv(18)=ableit(3)
c wv(19)=0
      wv(20)=abstand(1)    ! Regelabweichung
wv(21)=abstand(2)    ! Regelabweichung
wv(22)=abstand(3)    ! Regelabweichung
wv(23)=(va(1+3)-ableit(1))
wv(24)=(va(2+3)-ableit(2))
wv(25)=(va(3+3)-ableit(3))
wv(26)=torque(1)
wv(27)=torque(2)
wv(28)=torque(3)
wv(29)=pregler(1)
wv(30)=pregler(2)
wv(31)=pregler(3)
wv(32)=dnl(1)*(va(1+3)-ableit(1))
wv(33)=dnl(2)*(va(2+3)-ableit(2))
wv(34)=dnl(3)*(va(3+3)-ableit(3))
wv(35)=iconst*st(1)
wv(36)=iconst*st(2)
wv(37)=iconst*st(3)
wv(38)=Arbeit(1)
wv(39)=Arbeit(2)
wv(40)=Arbeit(3)
wv(41)=AAQ(1)
wv(42)=AAQ(2)
wv(43)=AAQ(3)


mywk1(1) = force(1)
mywk1(2) = force(2)
mywk1(3) = force(3)
mywk2(1) = torque(1)
mywk2(2) = torque(2)
mywk2(3) = torque(3)

torque(1) = zero
torque(2) = zero
torque(3) = zero
force(1) = zero
```

```
force(2) = zero
force(3) = zero

    DO i = 1,3
  DO  j = 1,3
force(i)  = force(i)  + mesm(i,j) * mywk1(j)
torque(i) = torque(i) + mesm(i,j) * mywk2(j)
      ENDDO
    ENDDO

c Diese Werte werden beim nächsten Durchlauf gebraucht! Deshalb hier am Schluß
par(offset9-4) = al
          par(offset9-5) = be
          par(offset9-6) = ga
          par(offset9-7) = inpfkt(1)
          par(offset9-8) = inpfkt(2)
          par(offset9-9) = inpfkt(3)
    DO i = 1,3
           par(offset9-15-i)=hilfs_2vektor(i)
    ENDDO
          par(offset9-19) = time

c      continue


c      par(offset9-1) = par(offset9-1) + 1
c  if( int( par(offset9-1) ) > 20 ) THEN
c   par(offset9-1) = 0
c      print*,' '
c     DO i = 1,6
c      print*,'arg',i,'  : ',ra(i), abstand(i)
c ENDDO
c       print*,' '
c      endif

C ----------------------------------------------------------------------
C Return and Error Handling
C ----------------------------------------------------------------------

      RETURN

C Warning - continue Calculations
 9100 continue
      print*, '** Warning by User Force Element of Type 22'
      print*, '   ', trim(c_nrf(id))  ! Force Element Name
      print*, '   Unknown Warning'    ! Insert Description
      return

C Error - stop Calculations
 9200 continue
      print*, '** ERROR in User Force Element of Type 22'
      print*, '   ', trim(c_nrf(id))  ! Force Element Name
      print*, '   Unknown Error'      ! Insert Description
      return 1

      END
```

# Abbildungsverzeichnis

# Literaturverzeichnis


[1] **Ballreich, R. und Baumann, W.** (Hrsg.): *Grundlagen der Biomechanik des Sports*, zweite Aufl. Enke, Stuttgart, 1996.

[2] **Böhm, H.**: *Direkte Dynamik einer Riesenfelge im Rahmen von Mehrkörpersystemen*. Universität Tübingen, 1997. Diplomarbeit.

[3] **Günther, M.**: *Computersimulationen zur Synthetisierung des muskulär gesteuerten menschlichen Gehens unter Verwendung eines biomechanischen Mehrkörpermodells*. Universität Tübingen, 1997. Dissertation.

[4] **Goldstein, Herbert**: *Klassische Mechanik*. Aula-Verlag, Wiesbaden, 1991.

[5] **Gruber, K.**: *Entwicklung eines Modells zur Berechnung der Kräfte im Knie- und Hüftgelenk bei sportlichen Bewegungsabläufen mit hohen Beschleunigungen*. Universität Tübingen and ETH Zürich, 1987. Dissertation.

[6] **Gruber, K., Ruder, H., Denoth, J. und Schneider, K.**: *A comparative study of impact dynamics: wobbling mass modell versus rigid body models*. Journal of Biomechanics (1998).

[7] **Haug, Edward J.**: *Computer aided kinematics and dynamics of mechanical systems*, Bd. 1: Basic methods aus *Allyn and Bacon series in engineering*. Allyn and Bacon, 1989.

[8] **Hildebrand, Falk**: *Eine biomechanische Analyse der Drehbewegungen des menschlichen Körpers*, Bd. 1 aus *Schriftenreihe zur angewandten Trainingswissenschaft*. Meyer&Meyer, Aachen, 1997.

[9] **Krebs, T.**: *Untersuchung der Lokomotion kleiner Säuger mit inverser Dynamik und Computersimulationen biomechanischer Mehrkörpermodelle*. Universität Tübingen, 1995. Dissertation.

[10] **Krieg, M.**: *Simulation und Steuerung biomechanischer Mehrkörpersysteme*. Universität Tübingen, Juni 1992. Diplomarbeit.

[11] **Kuypers, Friedhelm**: *Klassische Mechanik*. Wiley-VCH, 1997.

[12] **Merz, Ludwig und Jaschek, Hilmar**: *Grundkurs der Regelungstechnik*, zehnte Aufl. Oldenburg, München , 1990.

[13] **Nigg, Benno M. und Herzog, Walter** (Hrsg.): *Biomechanics of the Musculoskeletal System*. Wiley, Chichester, 1995.

[14] **Platzer, Werner**: *dtv-Atlas der Anatomie*, 6. überarb. Aufl., Bd. 1: Bewegungsapparat. Thieme: Stuttgart, NewYork und Dt.Taschenbuch-Verl.: München, 1991.

[15] **Press, William H.**: *Numerical recipes in C*. Cambridge University Press, 1988.

[16] **Rauber/Kopsch**: *Anatomie des Menschen*. Thieme: Stuttgart, New York, 1987.

[17] **Tittel, Kurt**: *Beschreibende und funktionelle Anatomie des Menschen*, 12., völlig überarb. Aufl. Fischer: Jena, Stuttgart, 1994.

# Danksagung

Für ihre freundliche Hilfe möchte ich mich bei Prof. Ruder, Dr. Luding, Renata Gandini, Arnim Henze, Valentin Keppler, Helmut Mutschler und Anton Pichler bedanken.