TECHNISCHE UNIVERSITÄT MÜNCHEN
Lehrstuhl für Angewandte Mechanik

Modellierung und Steuerung von strukturelastischen Robotern

Matthias J. Reiner

Vollständiger Abdruck der von der Fakultät für Maschinenwesen
der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr.-Ing. Gunther Reinhart
Prüfer der Dissertation:	
	1. Univ.-Prof. Dr.-Ing. Dr.-Ing. habil. Heinz Ulbrich
	2. Hon.-Prof. Dr.-Ing. Martin Otter

Die Dissertation wurde am 12.04.2010 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 23.09.2010 angenommen.

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter in der Abteilung für Systemdynamik und Regelungstechnik am Institut für Robotik und Mechatronik des Deutschen Zentrums für Luft- und Raumfahrt (DLR) in Oberpfaffenhofen.
Herrn Dr.-Ing. Johann Bals, Leiter der Abteilung für Systemdynamik und Regelungstechnik, sowie Herrn Hon.-Prof. Dr.-Ing. Martin Otter danke ich für die gute Zusammenarbeit sowie für die Betreuung der Arbeit am DLR.
Herrn Univ.-Prof. Dr.-Ing. Dr.-Ing. habil. Heinz Ulbrich danke ich für die Betreuung der Arbeit an der Technischen Universität München und die Übernahme des Erstgutachtens.
Bei Herrn Prof. Otter möchte ich mich zudem für die Erstellung des Zweitgutachtens bedanken.
Weiterhin danke ich Herrn Univ.-Prof. Dr.-Ing. Gunther Reinhart für die Übernahme des Prüfungsvorsitzes.
Darüber hinaus gilt mein Dank den Robotik-Teamkollegen Matthias Löhning, Sebastian Maier und Florian Saupe für die reibungslose Zusammenarbeit. Bei Dr. Schnepper, Dr. Joos, Dr. Heckmann und Dr. Bünte vom DLR möchte ich mich für die eingebrachte Expertise bedanken.
Der Firma KUKA Roboter möchte ich ebenfalls für die gute Zusammenarbeit danken, dies gilt insbesondere für Dr. Thümmel, Dr. Kurze und Dr. Weiß, die durch Ihre Unterstützung zum Gelingen der vorliegenden Arbeit beigetragen haben.
Nicht zuletzt gilt mein Dank meinen Eltern, die mich immer bestens unterstützt haben.

Matthias Reiner

Kurzfassung

In der vorliegenden Arbeit wird ein vollständiges Konzept zur Modellierung und Steuerung von Robotern mit Elastizitäten in den Antriebssträngen und Strukturteilen vorgestellt und experimentell verifiziert.
Hierzu werden objektorientiert aufgebaute, nichtlineare Modelle in unterschiedlichen Detaillierungsgraden in der Modellierungssprache *Modelica* erstellt und die Parameter der Modelle in einem mehrstufigen Identifikationsprozess bestimmt.
Eine systemtheoretische Untersuchung der erstellten Modelle zeigt, dass eine direkte Invertierung der Modelle – aufgrund der instabilen Nulldynamik – nicht möglich ist, weshalb unterschiedliche Ansätze zur approximativen Invertierung entwickelt und untersucht werden.
Mit Hilfe einer neu entwickelten, echtzeitfähigen Invertierungsmethode werden approximative strukturelastische inverse Modelle von zwei unterschiedlichen Robotern, mit 16 und 210 kg maximaler Traglast, erstellt und zur Steuerung der Roboter verwendet. Hierbei zeigt sich eine deutliche Verbesserung des Fahrverhaltens der Roboter im Experiment.
Zusätzlich werden auf Basis der erstellten inversen Modelle Trajektorien-Optimierungen durchgeführt, wodurch zeit- bzw. energieoptimale Trajektorien berechnet werden können, in welchen die Strukturelastizitäten sowie Antriebsstrangdynamik und Antriebsbegrenzungen berücksichtigt werden können.

Abstract

In this work a comprehensive approach to modeling and control of robots with elasticities in the drive trains and structural parts is presented and experimentally verified. For this purpose object-oriented, non-linear models in different level of detail are developed in the modeling language *Modelica* and the parameters of the models are obtained in a multilevel identification process.
A system theoretical study of the generated models shows that a direct inversion of the models – due to the unstable zero dynamics – is not possible, so different approaches for the approximate inversion are developed and investigated.
Using a newly developed, real-time inversion method approximate inverse models considering structural elasticity of two different industrial robots, with 16 and 210 kg (max.) payload, are created and are used for the control of the robots. The new control leads to a considerable improvement of the driving characteristics of the robot in the experiment.
In addition, based on the created inverse models, trajectory optimizations are done, allowing time- and energy-optimal trajectories to be calculated in which the complete elasticities and powertrain dynamics and limitations can be considered.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

1.1 Problemstellung

Von Seiten der Industrie werden stetig steigende Anforderungen bezüglich der Genauigkeit und Geschwindigkeit von Robotern gestellt. Zudem sehen sich Roboterhersteller, aufgrund der globalen Konkurrenz, einem steigenden Kostendruck ausgesetzt.

Um die Geschwindigkeit der Roboter zu erhöhen - was kleinere Taktzeiten in der Produktion bedeutet - müssen bei der Konstruktion leichtere Materialen und weniger massive Bauformen verwendet werden.

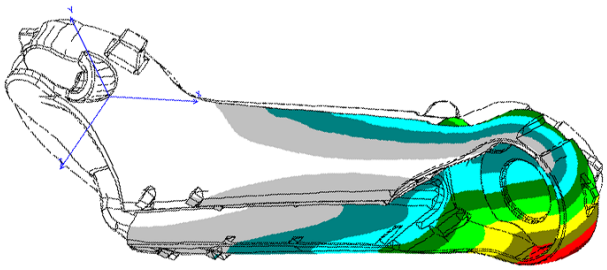


Abbildung 1.1: Beispiel für eine Verformung der Schwinge eines Roboters unter hohen Kräften.

Die Präzision der Roboter, auch bei hohen Geschwindigkeiten und Beschleunigungen, kann dadurch nicht mehr durch eine möglichst robuste Bauweise gewährleistet werden, sondern muss vor allem durch den gezielten Einsatz von fortschrittlichen Steuerungs- und Regelungsalgorithmen erreicht werden.

Vielversprechend sind in diesem Zusammenhang nichtlineare, modellbasierte Verfahren, die wegen ihrer Leistungsfähigkeit zunehmend zum Einsatz kommen. Durch die steigenden Anforderungen müssen für diese Algorithmen möglichst genaue Modelle verwendet werden, in welchen nicht nur die Elastizität der Antriebsstränge, sondern auch die Elastizität der Struktur berücksichtigt wird. Hierzu zählen insbesondere Torsions- und Biegeschwingungen der Bauteile, sowie Kippsteifigkeiten der Lagerungen der Gelenke. Diese – in dieser Arbeit als Strukturelastizität bezeichneten – Effekte begrenzen die erreichbare Genauigkeit und das Fahrverhalten des Roboters, wenn sie nicht

berücksichtigt werden. Abb. 1.1 zeigt ein Beispiel einer verformten Schwinge (Bauteil zwischen Achse 2 und 3) eines Roboters.

Auch wenn heutige Roboter noch relativ starr aufgebaut sind, kann eine Kompensation dieser strukturelastischen Effekte das Fahrverhalten der Roboter bereits deutlich verbessern. Für zukünftige Robotergenerationen wird dies immer entscheidender werden um wettbewerbsfähig zu bleiben.

Durch die kostenbedingte Forderung nach möglichst wenig Sensorik ist der Einsatz einer modellbasierten Vorsteuerung zweckmäßig, welche den Regler - dem nur sehr wenige Messgrößen zur Verfügung stehen - deutlich entlastet.

Um das zu erreichen muss der Roboter mathematisch modelliert werden. Ein gewonnenes Modell muss anhand von Messungen an den realen Roboter angepasst werden.

In dieser Arbeit werden neue Methoden entwickelt und experimentell verifiziert um strukturelastische Modelle für Roboter zu identifizieren und die erstellten nichtlinearen Modelle zu invertieren. Die hieraus abgeleiteten Vorsteuerungen sind so konzipiert, dass sie in Echtzeit im kompletten Arbeitsraum des Roboters eingesetzt werden können.

Experimente mit zwei unterschiedlichen Robotern zeigen, dass durch den Einsatz dieser inversen Modelle, in der Robotervorsteuerung, die Vibrationen des Roboters und das Trajektorienfolgeverhalten deutlich verbessert werden können.

1.2 Einführendes Beispiel zur Invertierung

Um einige grundlegende Verfahren zu verdeutlichen, die in dieser Arbeit verwendet werden, soll ein einfaches nichtlineares System untersucht werden. Das Beispiel soll zeigen wie ein nichtlineares System, welches als DAE[1] vorliegt invertiert werden kann. Hierbei soll auch auf die wesentlichen Problemstellungen bei dieser Art der Invertierung aufmerksam gemacht werden.

Bei dem Beispielsystem handelt es sich um einen Torsionsschwinger mit drei Trägheiten, welche über nichtlineare Federn und lineare Dämpfer verkoppelt sind. Ziel ist es, die inverse Dynamik des Systems zu berechnen, wodurch es möglich ist den Verlauf des Moments $\tau_1(t)$, welches an der ersten Trägheit Θ_1 angreift, so zu bestimmen, dass sich die dritte Trägheit Θ_3 entsprechend einer Trajektorien-Vorgabe ($\varphi_3(t)$) bewegt. Die inverse Dynamik kann daher als Vorsteuerung[2] für dieses System verwendet werden.

Die Nichtlinearität der Federn wird über Gleichung (1.1) beschrieben.

Über den Parameter γ kann eine Versteifung der Feder proportional zur quadratischen Verformung $(\Delta\varphi)^2$ eingestellt werden.

$$c(\Delta\varphi) = c_0 \left(1 + \gamma \left(\Delta\varphi\right)^2\right) \tag{1.1}$$

Durch einen Freischnitt des Systems (Abb. 1.2) lassen sich die Bewegungsgleichungen

[1] Englisch: Differential Algebraic Equations. Differential-algebraisches Gleichungssystem.

[2] In der Praxis ist es sinnvoll eine Vorsteuerung mit einem Regler zu kombinieren um Störungen auszugleichen und Modellierungsfehler zu kompensieren. Dieses Konzept ist als Regelung mit zwei Freiheitsgraden bekannt (mehr dazu in Abschnitt 5.1).

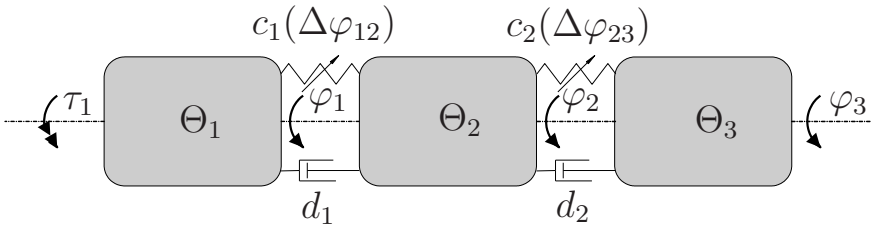


Abbildung 1.2: Beispiel zur Invertierung nichtlinearer Systeme: Torsionsschwinger mit drei Trägheiten. Drei Trägheiten $(\Theta_1, \Theta_2, \Theta_3)$ gekoppelt über nichtlineare Torsionsfedern $(c_1(\Delta\varphi_{12}), c_2(\Delta\varphi_{23}))$ und lineare Dämpfer (d_1, d_2). Die Koordinaten für die Rotation der drei Trägheiten sind $\varphi_1, \varphi_2, \varphi_3$. Ziel ist es für eine gewünschte Trajektorien-Vorgabe von $\varphi_3(t)$ das entsprechende Moment $\tau_1(t)$, welches an der ersten Trägheit Θ_1 angreift, zu bestimmen.

ermitteln.

$$\tau_1 = \Theta_1\ddot{\varphi}_1 + c_{0,1}(1 + (\varphi_2 - \varphi_1)^2\gamma_1)(\varphi_2 - \varphi_1) - d_1(\dot{\varphi}_2 - \dot{\varphi}_1) \tag{1.2a}$$

$$\begin{aligned} 0 = {} & \Theta_2\ddot{\varphi}_2 + c_{0,2}(1 + (\varphi_3 - \varphi_2)^2\gamma_2)(\varphi_3 - \varphi_2) - d_2(\dot{\varphi}_3 - \dot{\varphi}_2) \\ & + c_{0,1}(1 + (\varphi_2 - \varphi_1)^2\gamma_1)(\varphi_2 - \varphi_1) + d_1(\dot{\varphi}_2 - \dot{\varphi}_1) \end{aligned} \tag{1.2b}$$

$$0 = \Theta_3\ddot{\varphi}_3 - c_{0,2}(1 + (\varphi_3 - \varphi_2)^2\gamma_2)(\varphi_3 - \varphi_2) + d_2(\dot{\varphi}_3 - \dot{\varphi}_2) \tag{1.2c}$$

Um das inverse System lösen zu können muss der neue „Eingang“ des Systems φ_3 viermal differenzierbar sein, um die Gleichungen des inversen Systems aufstellen zu können. Die Differenzierung ist nötig um die gewünschten Ausgangsgrößen aus den Eingangsgrößen bestimmen zu können. Die Ausgangsgrößen müssen daher explizit mit Hilfe der gegebenen Eingangsgrößen darstellbar sein (weitere Details hierzu finden sich in Kapitel 3).

Durch differenzieren von Gl. (1.2c) erhält man:

$$\begin{aligned} 0 = {} & 2c_{0,2}(\varphi_3 - \varphi_2)(\dot{\varphi}_3 - \dot{\varphi}_2)\gamma_2(\varphi_3 - \varphi_2) \\ & + c_{0,2}(1 + (\varphi_3 - \varphi_2)^2\gamma_2)(\dot{\varphi}_3 - \dot{\varphi}_2) + d_2(\ddot{\varphi}_3 - \ddot{\varphi}_2) + \Theta_3\varphi_3^{(3)} \end{aligned} \tag{1.3}$$

Durch erneutes differenzieren von Gl. (1.3):

$$\begin{aligned} 0 = {} & 2(c_{0,2}((\dot{\varphi}_3 - \dot{\varphi}_2)(\dot{\varphi}_3 - \dot{\varphi}_2) + (\varphi_3 - \varphi_2)(\ddot{\varphi}_3 - \ddot{\varphi}_2))\gamma_2(\varphi_3 - \varphi_2) \\ & + c_{0,2}(\varphi_3 - \varphi_2)(\dot{\varphi}_3 - \dot{\varphi}_2)\gamma_2(\dot{\varphi}_3 - \dot{\varphi}_2)) \\ & + 2c_{0,2}(\varphi_3 - \varphi_2)(\dot{\varphi}_3 - \dot{\varphi}_2)\gamma_2(\dot{\varphi}_3 - \dot{\varphi}_2) \\ & + c_{0,2}(1 + (\varphi_3 - \varphi_2)^2\gamma_2)(\ddot{\varphi}_3 - \ddot{\varphi}_2) \\ & + d_2(\varphi_3^{(3)} - \varphi_2^{(3)}) + \Theta_3\varphi_3^{(4)} \end{aligned} \tag{1.4}$$

Durch differenzieren von Gl. (1.2b) erhält man:

$$\begin{aligned} 0 = {} & -2c_{0,1}(\varphi_2 - \varphi_1)(\dot{\varphi}_2 - \dot{\varphi}_1)\gamma_1(\varphi_2 - \varphi_1) \\ & - c_{0,1}(1 + (\varphi_2 - \varphi_1)^2\gamma_1)(\dot{\varphi}_2 + \dot{\varphi}_1) + d_1(\ddot{\varphi}_2 - \ddot{\varphi}_1) \\ & + 2c_{0,2}(\varphi_3 - \varphi_2)(\dot{\varphi}_3 - \dot{\varphi}_2)\gamma_2(\varphi_3 - \varphi_2) \\ & + c_{0,2}(1 + (\varphi_3 - \varphi_2)^2\gamma_2)(\dot{\varphi}_3 - \dot{\varphi}_2) - \Theta_2\varphi_2^{(3)} + d_2(\ddot{\varphi}_3 - \ddot{\varphi}_2) \end{aligned} \tag{1.5}$$

Gleichung (1.2a), (1.4) und (1.5) bilden nun die Bewegungsgleichungen für das inverse nichtlineare System.

Für die explizite Lösung der Gleichungen bietet es sich an, die Gleichungen auf eine nichtlineare Zustandsform nach Gleichung (1.6) zu transformieren.

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) \tag{1.6a}$$

$$\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u}, t) \tag{1.6b}$$

Für das Beispielsystem kann der Zustandsvektor $\boldsymbol{x}$, der Eingangsvektor $\boldsymbol{u}$ und der Ausgang y nach Gl. (1.7) gewählt werden:

$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} \ddot{\varphi}_2 \\ \dot{\varphi}_2 \\ \varphi_2 \\ \dot{\varphi}_1 \\ \varphi_1 \end{pmatrix}; \boldsymbol{u} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{pmatrix} = \begin{pmatrix} \varphi_3 \\ \dot{\varphi}_3 \\ \ddot{\varphi}_3 \\ \varphi_3^{(3)} \\ \varphi_3^{(4)} \end{pmatrix}; y = \tau_1 \tag{1.7}$$

Mit dieser Wahl können die Gleichungen (1.2a), (1.4) und (1.5) auf die Form $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t)$ und $\boldsymbol{g}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u}, t)$ transformiert werden was auf Gl. (1.8) führt.

Dieses Gleichungssystem kann durch einen geeigneten Integrationsalgorithmus (wie z. B. DASSL [1]) numerisch gelöst werden, wodurch die inverse Dynamik des Torsionsschwingers berechnet werden kann.

$$\begin{aligned}
\dot{x}_1 = \frac{1}{d_2}\Big(&-12c_{0,2}\gamma_2 u_2 x_2 u_1 + 6c_{0,2}\gamma_2 u_2^2 u_1 - 3c_{0,2}\gamma_2 x_3^2 x_1 \\
&+3c_{0,2}\gamma_2 x_3^2 u_3 - 3c_{0,2}\gamma_2 u_1^2 x_1 + 3c_{0,2}\gamma_2 u_1^2 u_3 - 6c_{0,2}\gamma_2 x_2^2 x_3 \\
&+6c_{0,2}\gamma_2 x_2^2 u_1 - 6c_{0,2}\gamma_2 u_2^2 x_3 \\
&+\Theta_3 u_5 + d_2 u_4 - c_{0,2} x_1 + c_{0,2} u_3 - 6c_{0,2}\gamma_2 u_1 u_3 x_3 \\
&+12c_{0,2}\gamma_2 u_2 x_2 x_3 + 6c_{0,2}\gamma_2 u_1 x_1 x_3\Big)
\end{aligned} \tag{1.8a}$$

$$\dot{x}_2 = x_1 \tag{1.8b}$$

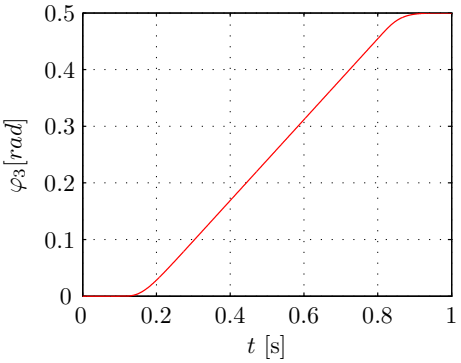
$$\dot{x}_3 = x_2 \tag{1.8c}$$

$$\begin{aligned}
\dot{x}_4 = \frac{1}{d_2 d_1}\Big(&\Theta_2 c_{0,2} u_3 - \Theta_2 c_{0,2} x_1 \\
&+\Theta_2 d_2 u_4 + \Theta_2 \Theta_3 u_5 - c_{0,2} u_2 d_2 + c_{0,2} x_2 d_2 \\
&+c_{0,1} x_2 d_2 - c_{0,1} x_4 d_2 + d_1 x_1 d_2 - d_2^2 u_3 + d_2^2 x_1 + 6\Theta_2 c_{0,2}\gamma_2 u_1 x_1 x_3 \\
&+12\Theta_2 c_{0,2}\gamma_2 u_2 x_2 x_3 - 6\Theta_2 c_{0,2}\gamma_2 u_1 u_3 x_3 - 12\Theta_2 c_{0,2}\gamma_2 u_2 x_2 u_1 \\
&+6c_{0,2}\gamma_2 u_1 x_3 u_2 d_2 - 6c_{0,2}\gamma_2 u_1 x_3 x_2 d_2 + 6c_{0,1}\gamma_1 x_3 x_5 x_4 d_2 \\
&-6c_{0,1}\gamma_1 x_3 x_5 x_2 d_2 - 6\Theta_2 c_{0,2}\gamma_2 u_2^2 x_3 + 6\Theta_2 c_{0,2}\gamma_2 x_2^2 u_1 - 6\Theta_2 c_{0,2}\gamma_2 x_2^2 x_3 \\
&+3\Theta_2 c_{0,2}\gamma_2 u_1^2 u_3 - 3\Theta_2 c_{0,2}\gamma_2 u_1^2 x_1 + 3\Theta_2 c_{0,2}\gamma_2 x_3^2 u_3 - 3\Theta_2 c_{0,2}\gamma_2 x_3^2 x_1 \\
&+6\Theta_2 c_{0,2}\gamma_2 u_2^2 u_1 + 3c_{0,1}\gamma_1 x_3^2 x_2 d_2 - 3c_{0,1}\gamma_1 x_3^2 x_4 d_2 + 3c_{0,1}\gamma_1 x_5^2 x_2 d_2 \\
&-3c_{0,1}\gamma_1 x_5^2 x_4 d_2 + 3c_{0,2}\gamma_2 u_1^2 x_2 d_2 - 3c_{0,2}\gamma_2 u_1^2 u_2 d_2 + 3c_{0,2}\gamma_2 x_3^2 x_2 d_2 \\
&-3c_{0,2}\gamma_2 x_3^2 u_2 d_2\Big)
\end{aligned} \tag{1.8d}$$

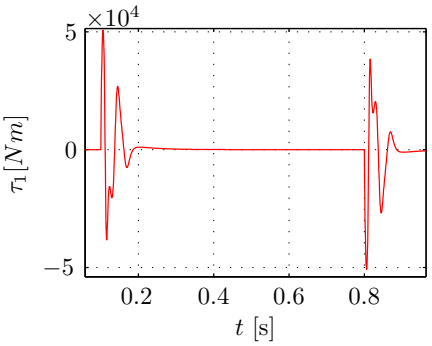
$$\dot{x}_5 = x_4 \tag{1.8e}$$

$$y = \Theta_1 \dot{x}_4 - c_{0,1}\left(1 + (x_3 - x_5)^2 \gamma_1\right)(x_3 - x_5) - d_1 (x_2 - x_4) \tag{1.8f}$$

Abb. 1.3 zeigt einen beispielhaften Verlauf des berechneten Moments τ_1 für eine vorgegebene, viermal stetig differenzierbare Trajektorie für φ_3. Die Differenzierbarkeit wird in diesem Beispiel durch einen kritisch gedämpften Filter (siehe Gl. (3.32)) vierter Ordnung gewährleistet.



(a) Beispielsystem: Verlauf von φ_3



(b) Beispielsystem: Verlauf von τ_1

Abbildung 1.3: Beispiel für einen berechneten Verlauf von τ_1 nach Gl. (1.8) für eine vorgegebene, viermal stetig differenzierbare Trajektorie von φ_3.

Anhand dieses einfachen Beispiels wird bereits deutlich, dass für eine Invertierung von nichtlinearen Systemen – nach dieser Methode – die Bewegungsgleichungen sowie die Solltrajektorie differenzierbar, sowie alle eingesetzten Funktionen eindeutig invertierbar sein müssen. Dies muss bereits bei der Modellierung (Kapitel 2) berücksichtigt werden. Unstetigkeiten müssen daher vermieden oder approximiert werden. Zudem wird bereits an diesem Beispiel deutlich, dass das Aufstellen der Gleichungen des inversen Systems für nichttriviale Beispiele nicht mehr von Hand durchführbar ist und rechnergestützte Hilfsmittel und Algorithmen verwendet werden müssen. Bei komplizierteren Systemen müssen Verfahren eingesetzt werden, welche die zu differenzierenden Gleichungen automatisch erkennen, um die inversen Systeme aufstellen zu können. Zudem muss die Stabilität des inversen Systems gegeben sein, was wie später genauer beschrieben werden wird, auf die Analyse der Nulldynamik der zu invertierenden Systems führt. Mit diesen Problemstellungen beschäftigt sich Kapitel 3.

1.3 Stand der Technik

In der Literatur findet sich eine große Anzahl an Publikationen zur Steuerung und Regelung von Robotern. Die unterschiedlichen Veröffentlichungen können dabei im Wesentlichen daran unterschieden werden, welche mechanischen Modelle zur Modellierung des Roboters verwendet werden, welche Komplexität die jeweiligen Modelle besitzen und welche Sensoren benutzt werden. Übliche Ansätze sind hierbei: Komplett starre Systeme, Roboter mit starren Gliedern und Elastizitäten in den Gelenken sowie strukturelastische Roboter mit elastischen Gelenken. Die Komplexität der Modelle kann durch die Anzahl der berücksichtigten Achsen des Manipulators sowie durch die modellierten mechanischen Effekte (wie z. B. Lose, Reibung oder nichtlineare Steifigkeiten) unterschieden werden. Auch die Identifikation von Robotern ist oftmals

Bestandteil der Veröffentlichungen, wobei hierbei zwischen linearen und nichtlinearen Modellen unterschieden werden muss.

Aus der Vielzahl der Veröffentlichungen sollen nun einige wesentliche Arbeiten kurz vorgestellt werden, welche sich mit der Thematik dieser Arbeit beschäftigen. Hauptgebiete dieser Arbeiten sind die Themen Modellierung, Steuerung und Regelung von (struktur-) elastischen Robotern.

In [2] wird ein zweiachsiger Laborroboter untersucht. Die Antriebe werden als lineare Federn modelliert, die Elemente als Euler-Balken. Zur Regelung des Systems mit einem PD-Regler auf Gelenkebene wird eine Winkelkorrektur durchgeführt, die über die Berechnung eines (linearen) mechanischen Ersatzsystems ohne dynamische Kräfte erfolgt. Die Verformungen der Struktur werden über Dehnmessstreifen gemessen und als zusätzliche Regelgröße verwendet. Für die Vorsteuerung wird ein, um diese Korrektur beaufschlagtes, Starrkörpersystem des Roboters verwendet.

In [3] wird ein zweiachsiger Laborroboter untersucht, welcher mit Hilfe von Balken modelliert wird. Zur Regelung werden zunächst über ein vereinfachtes linearisiertes Ersatzsystem Korrekturwinkel für die Gelenke berechnet. Diese werden zur Berechnung der Vorsteuergrößen mit Hilfe eines starren inversen Ersatzsystems verwendet und auf den Roboter geschaltet. Die Verformungen am Roboter werden gemessen (Dehnmessstreifen) und über PI-Regler kompensiert.

In [4] wird ein Laborroboter mit 6 Gelenken und zwei flexiblen Bauteilen untersucht und eine Vorsteuerung entworfen. Hierzu werden die Bewegungsgleichungen bezüglich einer Solltrajektorie linearisiert. Für die Verformungen wird ein quasistatischer Ansatz berechnet und eine Korrektur durchgeführt. Zusätzlich werden zur Steuerungen noch Verformungen der Schwinge mittels DMS gemessen und über PI-Regler zurückgeführt. Für die Invertierung der Massenmatrix wird ein $O(n)$ Algorithmus (siehe [5, 6]) verwendet.

Die Arbeit [7] verwendet Verfahren analog zu [4], erweitert diese jedoch um ein Kontaktproblem mit einer bewegten Ebene. Ziel ist es mit Hilfe einer Master-Slave Regelung ein Werkstück mit Hilfe von zwei Roboter zu führen. Hierbei wird der Master als starr angenommen und der elastische Laborroboter folgt dem Master um eine definierte Kontaktkraft aufzubringen. Hierfür wird zusätzlich zu der Vorsteuerung und Gelenks-PD Regelung (mit Messungen der Verformungen) eine übergeordnete Kraftregelung (bzw. Impedanzregelung) auf der Basis einer Kraftmessdose eingesetzt.

In [8] werden unterschiedliche mechanisch Ersatzmodelle für einzelne Balken und einen zweiachsigen Laborroboter untersucht. Dabei zeigt sich, dass die Nulldynamik der inversen Systeme instabil ist. Zur Lösung dieses Problems wird im linearen Fall eine nicht-kausale (rückwärts-) Integration vorgeschlagen, für den nichtlinearen Fall eine quasi-statische Korrektur nach [2] verwendet, welche noch zusätzlich zeitlich verschoben wird, um eine vorauseilende Korrektur zu erhalten. Die Gleichungen werden nach dieser Korrektur linearisiert und invertiert. Zur Kompensation werden zusätzlich die Verformungen am realen System gemessen und ausgeregelt.

In [9] wird ein SCARA[3]-Roboter untersucht. Die mechanische Modellierung erfolgt über ein gemischtes FEM/Starrkörper-Modell, welches linearisiert wird, wobei nur das

[3]Englisch: Selective Compliance Assembly Robot Arm.

letzte Strukturteil vor dem Endeffektor (TCP) des Roboters als elastisch angenommen wird. Für die Regelung des Systems wird ein hierarchisches Konzept verwendet. Zunächst werden Korrekturwinkel nach [3] berechnet. Zur Regelung werden zusätzliche Piezo-Elemente auf den Strukturteilen des Roboters angebracht, durch welche direkt (kolloziert) Schwingungen der Struktur gedämpft (und gemessen) werden können (sog. Dissipative Regler). Das so gedämpfte System wird über PD-Regler auf Gelenkebene ausgeregelt.

In [10] wird ein Roboter mit starrer Struktur und elastischen Antriebssträngen modelliert. Eine Untersuchung zeigt, dass für diese Modellstruktur keine instabile Nulldynamik auftritt. Über einen DAE-Ansatz wird dieses Modell invertiert und zur Vorsteuerung verwendet. Zusätzlich wird ein Zustandsregler entworfen in Kombination mit einem Beobachter der, mit Hilfe eines zusätzlichen Beschleunigungssensors am Roboter, die abtriebsseitigen Zustände schätzt.

In [11] wird ein strukturelastischer Roboter über ein Feder-Dämpfer Ersatzmodell modelliert. Dieses Modell wird über Linearisierungen im Frequenzbereich identifiziert. Ein vereinfachtes Modell (ebenes 2D-Modell), ohne instabile Nulldynamik, wird dann über einen DAE-Ansatz invertiert und zur Vorsteuerung eingesetzt. Zusätzlich wird ein Benchmark-Problem eines Viermassen-Schwingers untersucht, als Ersatzproblem für einen elastischen Roboter.

[12] beschäftigt sich mit der Identifikation von linearisierten Feder-Dämpfer Ersatzmodellen (Modelle wie in [11]) für einen elastischen Roboter. Hierbei werden verschieden Verfahren zur Schätzung und Identifizierung der Parameter dieser Modelle untersucht, sowie Untersuchungen bezüglich der Experimente – mit welchen die Daten zur Identifikation gewonnen werden – durchgeführt.

1.4 Beitrag und Aufbau der Arbeit

Diese Arbeit befasst sich mit der Modellierung und Steuerung von strukturelastischen Robotern. Hierzu werden zunächst verschiedene nichtlineare Modelle (DAE-Systeme) in der Modellierungssprache *Modelica* ([13]) erstellt welche, in unterschiedlichen Detaillierungsgraden, die wesentlichen dynamischen Eigenschaften des Roboters berücksichtigen.

Verschiedene Detaillierungsgrade der Modelle sind sinnvoll da, je nach Aufgabenstellung, unterschiedliche Rechenzeit zur Verfügung steht oder Echtzeitanforderungen erfüllt werden müssen.

Anhand von Messungen an zwei Robotern[4] werden die Parameter der Modelle über einen neuen, mehrstufigen Identifikationsprozess auf Basis einer Kombination von globaler und lokaler Optimierung und Sensordatenfusionierung ermittelt.

Untersuchungen der Modelle zeigen, dass eine instabile Nulldynamik – für Modelle mit Strukturelastizitäten – vorhanden ist, wenn Motorgrößen als Eingang und die Position der Last (bzw. des TCP[5]) als Ausgang gewählt werden.

Für die verschiedenen Modellierungsansätze werden neue approximative Invertierungs-

[4] KR16 mit max. 16 kg Traglast und KR210 mit max. 210 kg Traglast.
[5] Tool Center Point, Werkzeugspitze oder Endeffektor.

methoden untersucht und ausgearbeitet, mit welchen die erstellten Modelle – trotz instabiler Nulldynamik – invertiert werden können. Die erstellten inversen Modelle können direkt zur Vorsteuerung des Roboters eingesetzt werden. Für einfache Modellvarianten kann dies in Echtzeit durchgeführt werden. Zudem können, basierend auf den erstellten Modellen, zeit- oder energieoptimale Trajektorien berechnet werden.

Der neu entwickelte Ansatz erlaubt hierbei die komplette Berücksichtigung der Nichtlinearitäten und ist nicht auf eine Linearisierung der Gleichungen angewiesen. Insbesondere der Ansatz zur Kompensation der Strukturelastizitäten über ein Parallelmodell unterscheidet sich von den in der Literatur vorgeschlagenen Methoden, da er speziell auf die Modellstruktur angepasst ist und auch Dämpfung im System berücksichtigt.

Die erstellten inversen Modelle werden an den 6-freiheitsgradigen Robotern KR16 (Nutzlast: 16 kg) und KR210 (Nutzlast: 210 kg) unter realistischen Bedingungen (kein ebenes Problem oder Einschränkungen im Arbeitsraum) zur Vorsteuerung eingesetzt. Dabei werden deutliche Verbesserungen im Schwingungsverhalten erzielt. Zusätzlich werden auf Basis der erstellten inversen Modelle zeit- und energieoptimale Soll-Trajektorien berechnet, welche die kompletten Elastizitäten sowie Stellgrößen- und Zustandsbeschränkungen berücksichtigen.

Der Aufbau der Arbeit gliedert sich wie folgt:

In Kapitel 2 werden die einzelnen Komponenten der Modelle für die Roboter KR16 und KR210 vorgestellt.

Die grundsätzlichen Probleme bei der Invertierung, sowie die hierfür neu entwickelten Methoden und Verfahren beschreibt Kapitel 3.

Kapitel 4 beschreibt ein neues Verfahren zur Identifikation strukturelastischer Modelle. Das Verfahren wird zur Identifikation der Roboter KR16 und KR210 angewendet.

In Kapitel 5 werden unterschiedliche Anwendungsmöglichkeiten und Trajektorien-Optimierungen, basieren auf den erstellten inversen Modellen, vorgestellt.

Messungen und experimentelle Ergebnisse beim Einsatz der inversen Modelle an realen Roboter beschreibt Kapitel 6.

Eine Zusammenfassung der Ergebnisse bildet Kapitel 7.

Kapitel 2

Modellierung von strukturelastischen Robotern

In diesem Kapitel wird die Modellierung des Roboters erläutert. Bei den untersuchten Robotern handelt es sich um Manipulatoren mit sechs Freiheitsgraden. Weitere Details zu den untersuchten Robotern befinden sich im Anhang A.1.

Besonderes Augenmerk bei den Modellen liegt darauf, neben den elastischen Antriebssträngen, auch die Elastizität der Roboter-Strukturteile und Gelenklagerungen zu berücksichtigen.

Dies unterscheidet die Modellierung von vielen in der Literatur üblichen (z. B. [14, 15, 10]) Robotermodellen, bei welchen eine starre Struktur in Kombination mit elastischen Gelenken angenommen wird[1].

2.1 Baukastensystem zur Modellierung von Robotern

Ziel der Modellbildung ist es mathematische Modelle für die untersuchten Roboter zu gewinnen, welche die Dynamik der Systeme möglichst genau abbilden. Die Modelle sind hierbei objektorientiert aufgebaut, wobei sich die Aufteilung in einzelne Komponenten/Module nach physikalischen Eigenschaften des Roboters richtet.

Die einzelnen Komponenten der Modelle können, je nach Anwendungsfall und verfügbarer Rechenzeit, zu unterschiedlichen Gesamtmodellen zusammengesetzt werden. Abb. 2.1 zeigt das Schema dieses Baukastensystems.

Für exakte Referenzmodelle stehen dafür sehr komplexe Module zur Verfügung, welche für Synthesemodelle durch einfachere (approximative) Module ersetzt werden können. Für Synthesemodelle, welche zur Steuerung und Regelung eingesetzt werden, dürfen die Module eine gewisse Komplexität nicht überschreiten, da sonst die verfügbare Rechenzeit nicht ausreicht. Hier muss ein Kompromiss zwischen Genauigkeit und benötigter Rechenzeit gefunden werden.

[1] Englische Abkürzung: Rigid Link Flexible Joint Robot (RLFJ)

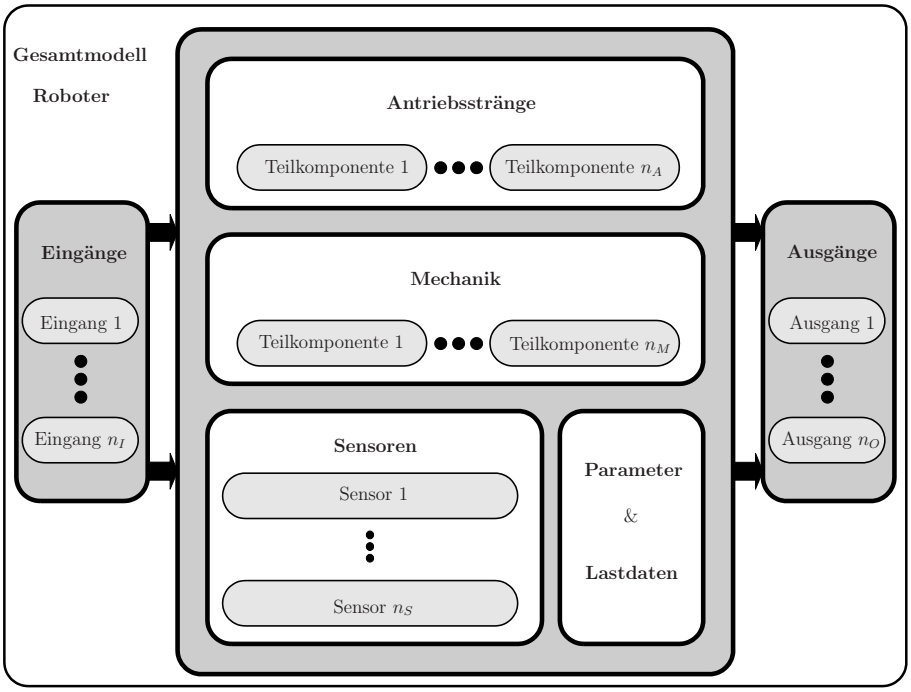


Abbildung 2.1: Baukastensystem: Gesamtmodelle des Roboters können, je nach Anwendungsfall und verfügbarer Rechenzeit, aus Teilkomponenten zusammengestellt werden. Auch die Ein- und Ausgänge können dabei als Komponenten des Modells betrachtet werden. Durch eine Änderung dieser können vorwärts und inverse (unter gewissen Voraussetzungen) Modelle generiert werden. Zusätzliche Sensoren im Modell und unterschiedliche (Nutz-) Lasten des Roboters können ebenfalls als Komponente des Baukastens aufgefasst werden. Ziel bei der Modellierung ist es, möglichst alle Teilkomponenten einfach austauschbar zu halten und sie so zu modellieren, dass auch inverse Modelle abgeleitet werden können.

Da je nach benötigter Aufgabe, bzw. regelungstechnischem Ansatz, unterschiedliche Modelle nötig sind, werden im Folgenden die einzelnen Komponenten der Modelle vorgestellt, aus denen je nach Einsatz die unterschiedlichen Gesamtmodelle (z. B. Referenz- oder Synthesemodelle) zusammen gestellt werden können.

Die Modellierung der einzelnen Komponenten erfolgt in der Modellierungssprache *Modelica* [13] und dem *Modelica* -Simulationswerkzeug *Dymola* [16].

Modelica ist eine sehr flexibel aufgebaute Modellierungssprache in der objektorientiert programmiert werden kann. Über Konnektoren können automatisch Bilanzgleichungen für Fluss- und Potentialvariablen erstellt werden. Verbindungen der einzelnen Komponenten beschreiben daher nicht einfach eine „Informationsflussrichtung“, sondern erzeugen neue Gleichungen.

Im Falle eines mechanischen Flansch-Konnektors werden so z. B. zwei Gleichungen (2.1) erstellt, welche beschreiben, dass die Geschwindigkeit für beide verbundenen Konnektoren (*flange_a* und *flange_b*) identisch ist und dass die Summe der eingeleiteten Momente verschwindet (Aktio gleich Reaktio).

$$flange\_a.phi = flange\_b.phi \tag{2.1a}$$

$$flange\_a.tau + flange\_b.tau = 0 \tag{2.1b}$$

Ein ausführliche Beschreibung der Sprache findet sich in [17].

2.2 Motor und Reibung

In diesem Abschnitt sollen Modelle für die Motoren und Reibung innerhalb der Antriebsstränge des Roboters vorgestellt werden.

Die Motoren der Antriebsstränge haben – außer bei extrem geringen Geschwindigkeiten – keinen wesentlichen Einfluss auf das dynamische Verhalten des Roboters und können relativ schlicht modelliert werden. Für ein mechanisch orientiertes Modell genügt es daher den Motor lediglich als eine Trägheit Θ_m der entsprechenden Achse zu modellieren, zu welcher auch die Trägheit der Motorwelle addiert werden kann. Das Motormoment τ_m kann dabei als proportional zum Motorstrom I_m angenommen werden.

Dies ist damit zu begründen, dass die Dynamik des Motors gegenüber der deutlich langsameren Dynamik der Mechanik vernachlässigbar ist (Unterschied der dynamischen Zeitkonstanten ist groß). Soll die Dynamik berücksichtigt werden, bietet es sich an den Motor als Verzögerungsglied erster Ordnung (VZ1) zu modellieren (2.2).

$$\tau_m + T_m \dot{\tau}_m = k_T I_m \tag{2.2}$$

Lediglich bei extrem langsamen Bewegungen ist es nötig den Motor genauer zu modellieren, da hierbei das sogenannte „Cogging“ auftreten kann. Unter diesem Begriff versteht man Reluktanzkräfte aufgrund magnetischer Unsymmetrien, welche durch die endliche Anzahl an Polpaaren sowie die geometrischen und magnetischen Toleranzen der Fertigung verursacht werden. Dieser Effekt soll hier aber nicht weiter berücksichtigt werden. Für weitere Details und Modellierungsansätze hierzu sei z. B. auf [18] und [19] verwiesen.

Die Reibung hat einen sehr großen Einfluss auf die Dynamik des Roboters. Reibung entsteht an verschiedenen Stellen am Roboter. Der größte Teil stammt hierbei aus der Reibung aus dem Antriebsstrang, wobei die Reibung in Getriebe und Motor den Großteil ausmachen. Zusätzliche Quellen für Reibungsverluste sind die Lager des Roboters, sowie deren Schmierung. Die Modellierung der Reibung ist ein sehr komplexes Feld und unterschiedliche Effekte, die zur Gesamtreibung führen, sind nur schwer einzeln messbar. Eine Übersicht zur Modellierung von Reibung bieten etwa [20],[21] und [22].

Beim Roboter hängt die Reibung im Wesentlichen von der Temperatur von Motor und Getriebe sowie der Winkelgeschwindigkeit $\dot{\varphi}$ im Antriebsstrang ab. Auch die Belastung des Antriebsstrangs spielt dabei eine Rolle (Lastabhängigkeit). Zudem verändert sich die Reibung eines Roboters mit dem Alter der Bauteile und ihrer Abnutzung, sowie dem Zustand der Schmierstoffe (Öle und Fette).

Es kann ein nahezu beliebiger Aufwand bei der Modellierung von Reibung betrieben werden. In dieser Arbeit wurden nur relativ einfache Reibmodelle untersucht, die aber die wesentlichen Effekte der Reibung auf die Roboterdynamik möglichst exakt wiedergeben.

Zur Vereinfachung wird daher die Annahme getroffen, dass die komplette Reibung des Roboters in eine antriebsseitige Reibungskomponente am Motor τ_{rm}, sowie eine abtriebsseitige Reibungskomponente τ_{ra} nach dem Getriebe am Roboter aufgeteilt werden kann. Aufgrund der hohen Übersetzungsverhältnisse der Getriebe eines Industrieroboters und die dadurch höheren Winkelgeschwindigkeiten an der Antriebsseite gilt zudem

$\tau_{rm} \gg \tau_{ra}$ (unter der Annahme, dass beide Größen über die Getriebeübersetzung auf die Abtriebsseite des Getriebes umgerechnet werden). Die vier wesentlichen Kompo-

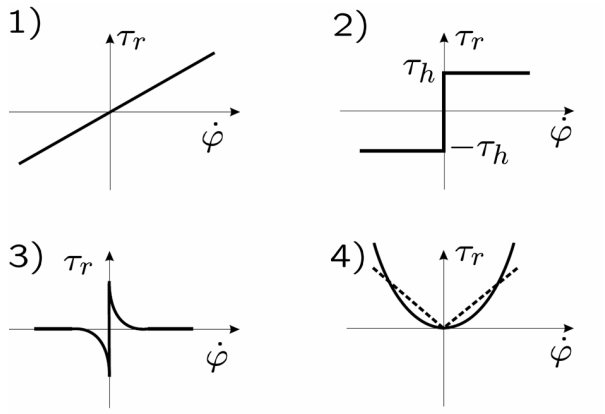


Abbildung 2.2: Die vier wichtigsten Reibungskomponenten (modifiziert aus [15]).

nenten, aus denen sich die Reibung zusammen setzt, sind in Abbildung 2.2 dargestellt.

1. Viskose Reibung (Gleitreibung)
2. Haftreibung
3. Stribeck-Effekt (Stick-Slip-Effekt)
4. Lastabhängige Reibung

Diese Komponenten überlagern sich und werden zusätzlich noch stark von der Temperatur beeinflusst. Die lastabhängige Reibung macht hierbei nur einen sehr kleinen Anteil aus und kann im Allgemeinen vernachlässigt werden. Der Stribeck-Effekt ist bei den untersuchten Robotern ebenfalls nur sehr schwach ausgeprägt.

Zur Modellierung der Reibung wurden verschiedene Varianten untersucht. Eine Variante ist die Näherung der Reibungskennlinie durch Gleichung (2.3). Hierbei ist τ_r das resultierende Reibmoment aufgrund der Winkelgeschwindigkeit $\dot{\varphi}$. τ_h ist ein Parameter für die Größe der Haftreibung, τ_v ein Parameter für die Gleitreibung und $s \gg 1$ ein Parameter für die Steilheit der Reibkennlinie für kleine Werte von $\dot{\varphi}$. Die e-Funktion wird verwendet, damit die Funktion differenzierbar bleibt.

$$\tau_r(\dot{\varphi}) = \tau_v \dot{\varphi} + \frac{2\tau_h}{1 + e^{-s\dot{\varphi}}} - \tau_h \tag{2.3}$$

Die Haftphase wird hierbei nur sehr vereinfacht modelliert, allerdings mit dem Vorteil, dass die Kennlinie invertierbar ist, da sie keinen Sprung bei $\dot{\varphi} = 0$ aufweist.

Soll auch der Stribeck-Effekt berücksichtigt werden bietet sich einen Erweiterung von Gl. (2.3) an. Durch einen zusätzlichen Term (wie in [10] vorgeschlagen) führt dies zu Gl. (2.4), worin die Parameter a und c zur Dimensionierung der Stribeck-Kurve (Stick-Slip-Effekt) dienen.

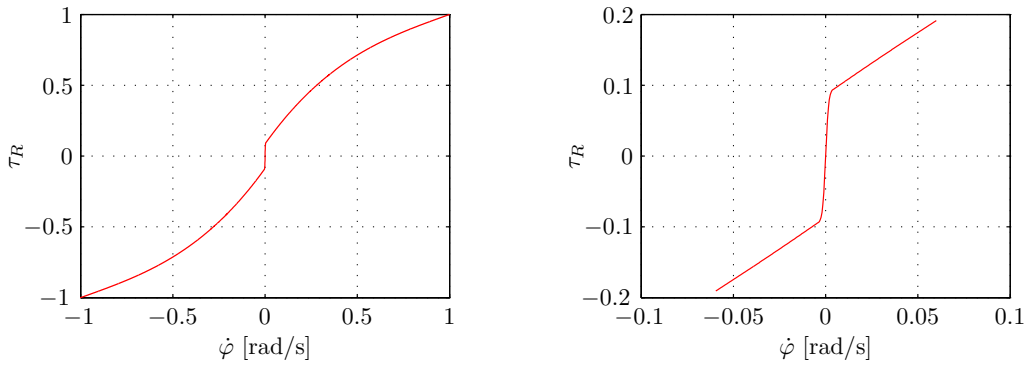
$$\tau_r(\dot{\varphi}) = \tau_v \dot{\varphi} + \frac{2\tau_h}{1 + e^{-s\dot{\varphi}}} - \tau_h + \frac{\dot{\varphi}}{a\dot{\varphi} + c} \tag{2.4}$$

Soll die Haftphase genauer modelliert werden, ist eine Zustandsumschaltung nötig. Während der Haftphase ergibt sich das Reibmoment dann aus einer Momentenbilanz zwischen Flansch A und Flansch B unter der Nebenbedingung, dass $\dot{\omega}_A = \ddot{\varphi}_A = 0$ gilt. Übersteigt das berechnete Moment die Haftreibung τ_h kann die Reibung $\tau_r = \tau_r(\dot{\varphi})$ wieder aus der Geschwindigkeit des antreibenden Flansches A berechnet werden. Dieses Verfahren wurde, wie in [23] beschrieben, in *Modelica* implementiert. Nachteilig hieran ist allerdings, dass dieses Reibmodell dann nicht mehr exakt invertiert werden kann, was eine Grundvoraussetzung für ein inverses Modell ist, welches für eine Steuerung benötigt wird.

Nach der Analyse von Reibmessungen wurde ein erweitertes Modell für die Reibung aufgestellt, in dem die Kennlinie in verschiedene Abschnitte eingeteilt wird. Der Nulldurchgang wird hierbei ebenfalls durch eine e-Funktion beschrieben. Im Bereich von höheren Geschwindigkeiten beschreibt ein Polynom (mit Koeffizientenvektor $\boldsymbol{p}_\tau$ mit Elementen $\boldsymbol{p}_{\tau,i}$, $i = 1, 2, \dots$) die Kennlinie, das so gewählt wird, dass sich kontinuierliche Übergänge ergeben. Im Bereich sehr hoher Geschwindigkeiten geht die Kennlinie in eine Gerade über (siehe Gleichung (2.5).

$$\tau(\dot{\varphi}) = \begin{cases} \tau_{lin} + d\tau_{lin}(\dot{\varphi} - 1), & \dot{\varphi} > 1 \\ \boldsymbol{p}_{\tau,1}\dot{\varphi}^3 + \boldsymbol{p}_{\tau,2}\dot{\varphi}^2 + \boldsymbol{p}_{\tau,3}\dot{\varphi} + \boldsymbol{p}_{\tau,4}, & \dot{\varphi}_{step} \leq \dot{\varphi} \leq 1 \\ \frac{2R_h}{1+e^{-s\dot{\varphi}}} - R_h, & -\dot{\varphi}_{step} < \dot{\varphi} < \dot{\varphi}_{step} \\ \boldsymbol{p}_{\tau,1}\dot{\varphi}^3 - \boldsymbol{p}_{\tau,2}\dot{\varphi}^2 + \boldsymbol{p}_{\tau,3}\dot{\varphi} - \boldsymbol{p}_{\tau,4}, & -\dot{\varphi}_{step} \geq \dot{\varphi} \geq -1 \\ -\tau_{lin} + d\tau_{lin}(\dot{\varphi} + 1), & \dot{\varphi} < -1 \end{cases} \tag{2.5}$$

Dieses Modell ist mathematisch sehr gut handhabbar und es lässt sich damit eine sehr



(a) Reibmodell - Verlauf

(b) Reibmodell - Detail

Abbildung 2.3: Reibmodell für den Roboter. Das Reibmoment ist bezüglich des Reibmoments bei $\dot{\varphi} = 1$ [rad/s] normiert. Abb. 2.3(b) zeigt die Approximation des Nulldurchgangs der Reibkennlinie.

gute Übereinstimmungen mit den Messungen erreichen. Der Stribeck-Effekt wurde in diesem Modell nicht berücksichtigt, da er nicht (relevant) messbar war. Abbildung 2.3 zeigt die in Gleichung (2.5) beschriebene Funktion.

Die Identifizierung der Temperatur- und Lastabhängigkeit der Reibung der einzelnen Achsen des Roboters ist nur durch sehr aufwändige Messreihen möglich. Problematisch für den Einsatz einer temperaturabhängigen Reibungskennlinie als Teil einer Vorsteuerung ist, dass es bei den untersuchten Robotern keinen Temperatursensor im Getriebe

am Roboter gibt, sondern lediglich einen Temperatursensor am Motor des Roboters, wodurch keine genauen Temperaturwerte für die Reibung zur Verfügung stehen.

Für die Modelle wurde daher eine temperaturunabhängige Reibungskennlinie verwendet da, durch die Ungenauigkeit der Temperaturmessungen in Versuchen, keine deutliche Verbesserung durch den Einsatz einer temperaturabhängigen Reibkennlinie erreicht werden konnte.

Alternativ kann die Reibung auch online bestimmt werden, was in [24] untersucht wurde. Hierbei wird die Reibung direkt am Roboter über Neuronale Netze geschätzt. Die Neuronalen Netze wirken hierbei als Funktionsapproximatoren, wobei der Fehler zwischen Vorsteuermoment (ohne Reibung) und tatsächlichem Moment minimiert wird. Nachteilig hieran ist, dass der Roboter zuerst eine gewisse Lernphase durchlaufen muss, bevor ein gutes Reibmodell vorhanden ist und sich alle Fehler der Vorsteuerung und Regelung im Reibmodell wiederfinden. Vorteilhaft ist bei dieser Vorgehensweise, dass sowohl Temperatureffekte, als auch Alterungserscheinungen mitgelernt werden können.

Dieses Verfahren stellt eine Alternative zu einer statischen Reibkennlinie dar, es muss jedoch zusätzliche Rechenzeit zur Berechnung der Neuronalen Netze vorhanden sein und die Stabilität des verwendeten Lernalgorithmus muss gewährleistet sein.

2.3 Getriebe

Bei den untersuchten Robotern werden Getriebe mit hohen Übersetzungen eingesetzt. Die Dynamik der Getriebe ist sehr wichtig für die Dynamik des gesamten Roboters. In der Literatur existieren sehr aufwändige Getriebemodelle (z. B. [22, 25, 26]), diese sind jedoch zu komplex um sie in einer Vorsteuerung sinnvoll einsetzen zu können. Zur Approximation eignen sich nichtlineare Feder-Dämpfer-Modelle in Kombination mit Modellen für Wirkungsgrad und Lose, wodurch die wesentlichen Eigenschaften der Getriebe gut beschrieben werden können. Für eine detailliertere Modellierung ist auch noch die Hysteresemodellierung möglich.

Auch die Reibung spielt bei den Getrieben eine wichtige Rolle. Sie kann analog zur Motorreibung (siehe Abschnitt 2.2) getrennt modelliert werden oder zur Motorreibung hinzugerechnet werden.

2.3.1 Getriebeübersetzung

Die Übersetzung i des Getriebes kann entweder durch eine Umrechnung der Winkelgeschwindigkeiten und Momente von Flansch A auf Flansch B erreicht werden oder vorab, durch Umrechnung der antriebsseitigen Trägheiten auf die Abtriebsseite des Getriebes und entsprechender Anpassung der Geschwindigkeiten, wobei dann im Modell $i = 1$ gesetzt werden kann.

Problematisch ist, dass die eingesetzten Getriebe keine perfekte Übersetzung erzeugen können, sondern immer mit Übersetzungsfehlern behaftet sind. Wie in [10] gezeigt wurde kann die Übersetzung durch die Gleichungen (2.6) modelliert werden. Die Gleichungen stellen eine Fourierreihe mit n_i ganzzahligen Frequenzen $\breve{i}$ dar, wobei i_0 für

die Basisübersetzung steht. $\hat{i}$ ist die Amplitude der jeweiligen Schwingung und $\varphi_{i,0}$ der Startwinkel.

$$i = i_0 + \sum_{j=1}^{n_i} \hat{i}_j \cos(\breve{i}_j \varphi_B + \bar{i}_j) \tag{2.6a}$$

$$\varphi_A = i_0 \varphi_B + \sum_{j=1}^{n_i} \frac{\hat{i}_j}{\breve{i}_j} \sin(\breve{i}_j \varphi_B + \bar{i}_j) + \varphi_{i,0} \tag{2.6b}$$

$$\tau_A = -\frac{\tau_B}{i} \tag{2.6c}$$

Mit Hilfe dieser Gleichungen ist es bereits für kleine Werte für n_i möglich die Drehungleichförmigkeit des Getriebes zu modellieren.

Für ein Modell, das zur Vorsteuerung eingesetzt werden soll, sind die Gleichungen (2.6) allerdings weniger geeignet. Die Startwinkel $\varphi_{i,0}$ und die Phasenverschiebung $\bar{i}_j$ können nach einer (erneuten) Justage des Roboters nur unter hohem Aufwand ermittelt werden, da sie von der Stellung der Zähne im Getriebe abhängen und diese, ohne zusätzliche absolute Sensorik, nicht ermittelt werden kann. Für Modelle, die zur Vorsteuerung einsetzt werden sollen, muss daher eine ideale Übersetzung $i = i_0$ im Getriebe angenommen werden (mit entsprechender Umrechnung der Trägheiten und Geschwindigkeiten auf die Abtriebsseite des Getriebes). Sollen Getriebeschwingungen dennoch berücksichtigt werden, müssen diese online ermittelt und eingerechnet werden.

2.3.2 Getriebesteifigkeit

Eine Möglichkeit, die Steifigkeit des Getriebes zu modellieren, ist sie als nichtlineare Feder anzunehmen. Gleichung (2.7) stellt eine Möglichkeit für die Darstellung der nichtlinearen Federkennlinie dar. Gleichung (2.7) ist differenzierbar und besitzt nur wenige Parameter. γ ist hierbei ein konstanter Versteifungsfaktor für die Federkennlinie c_0 ist ein Parameter für die Steifigkeit des Getriebes bei kleiner Last. $\varphi_{rel} = \varphi_B - \varphi_A$ steht dabei für den relativen Winkel zwischen Flansch A und B, wobei die Winkel, nach Konvention in dieser Arbeit, immer bezüglich der Abtriebsseite des Getriebes angegebenen werden (Umrechnung über die Getriebeübersetzung, dies gilt auch für Trägheiten auf der Motorseite des Getriebes).

$$\tau_\gamma(\varphi_{rel}) = \varphi_{rel} c_0 (1 + \gamma \varphi_{rel}^2) \tag{2.7}$$

Mit dieser Gleichung kann die Versteifung der Feder bei hoher Belastung, welche in Messungen festgestellt werden kann, gut modelliert werden. Nachteilig ist an der Modellierung nach Gl. (2.7), dass durch den quadratischen Term die Steifigkeit sehr stark anwachsen kann. Soll ein Maximum $c_{max} > c_0$ für die Steifigkeit der Feder gesetzt werden so kann Gl. (2.8) verwendet werden.

$$\tau_{exp}(\varphi_{rel}) = \varphi_{rel} \left(c_0 + (c_{max} - c_0) \left(1 - e^{-\gamma |\varphi_{rel}|}\right)\right) \tag{2.8}$$

Alternativ kann Gl. (2.7) durch die Verwendung einer atan2(.) Funktion[2] auf $c_{max} > c_0$ limitiert werden, was auf Gl. (2.9) führt (mit Parameter $s_{lim} \gg 1$ zum Einstellen der

[2] Anmerkung zur atan2-Funktion finden sich im Anhang unter A.5.

Steilheit der Begrenzung).

$$\tau_{\gamma,lim}(\varphi_{rel}) = \varphi_{rel}\Bigg(c_\gamma - c_\gamma\left(\frac{\operatorname{atan2}\left(\left(c_\gamma - c_{max}\right)s_{lim}\right)}{\pi} + 1 - \frac{\operatorname{atan2}\left(c_\gamma s_{lim}\right)}{\pi}\right) + c_{max}\left(\frac{\operatorname{atan2}\left(\left(c_\gamma - c_{max}\right)s_{lim}\right)}{\pi} + \frac{1}{2}\right)\Bigg) \tag{2.9a}$$

$$\text{mit: } c_\gamma = c_0\left(1 + \gamma\varphi_{rel}^2\right) \tag{2.9b}$$

Die Getriebe-Dämpfung kann über den linearen Term $\tau_d = d\dot{\varphi}_{rel}$ modelliert werden. In Kombination mit dem Getriebemoment ergeben sich dann Feder-Dämpfer-Elemente.

2.3.3 Wirkungsgrad

Einen zusätzlichen Effekt des Getriebes stellt der kraftflussrichtungsabhängige Wirkungsgrad des Getriebes dar. Unter Kraftfluss wird hierbei das Produkt aus Moment und Winkelgeschwindigkeit $\tau_A\omega_A$ des angetriebenen Flansches A verstanden. Der aktuelle, drehzahlabhängige Wirkungsgrad $\hat{\eta}(\omega_A)$ des Getriebes kann dann, wie in [23] beschrieben, durch Gleichung (2.10) berechnet werden.

$$\hat{\eta}(\omega_A) = \begin{cases} \eta_1(|\omega_A|), & \text{wenn } \tau_A\omega_a > 0 \text{ oder } \tau_A = 0 \wedge \omega_A > 0 \\ \frac{1}{\eta_2(|\omega_A|)}, & \text{wenn } \tau_A\omega_a < 0 \text{ oder } \tau_A = 0 \wedge \omega_A < 0 \\ \text{so, dass } \dot{\omega}_A = 0, & \text{wenn } \omega_A = 0 \end{cases} \tag{2.10}$$

Für $\omega_A = 0$ findet hierbei, wie im Falle der Haftreibung eine Zustandsumschaltung statt, wobei sich eine neue Bedingung für das Gleichungssystem ergibt (Fall 3). Dies führt dazu, dass dieses Gleichungssystem, wie im Fall der exakten Modellierung der Haftreibung, nicht mehr invertierbar ist.

Um dieses Problem zu umgehen kann eine Approximation des Wirkungsgrads vorgenommen werden. Hierzu wird eine Näherung für den Nulldurchgang von ω_A vorgenommen.

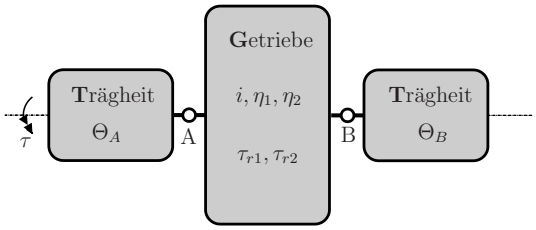
Die approximierte Berechnung für den Wirkungsgrad des Getriebes kann durch Gl. (2.11) beschrieben werden. $\xi_{\tau\omega}$ steht dabei für die approximative Kraftflussrichtung. $\tau_{r1} = \tau_{r1}(\omega_A)$ und $\tau_{r2} = \tau_{r2}(\omega_A)$ stellen die Reibung des Getriebes nach Gleichung (2.3) dar, wobei der Index 1 bzw. 2 für die positive bzw. negative (approx.) Kraftflussrichtung $\xi_{\tau\omega}$ steht. Der Parameter $s \gg 1$ dient zum Einstellen der Steilheit der Approximation beim Nulldurchgang von ω_A. η_1 und η_2 können hierbei in Abhängigkeit von $|\omega_A|$ gewählt werden (wobei $0 < \eta_i \leq 1$).

$$0 = \tau_B + i\left(\tau_A - \tau_{loss}\right) \tag{2.11a}$$

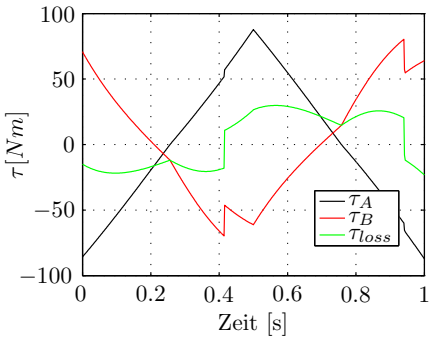
$$\xi_{\tau\omega} = \tfrac{\operatorname{atan2}(s\tau_A\omega_A)}{\pi} + \tfrac{1}{2} \tag{2.11b}$$

$$\tau_{loss} = \xi_{\tau\omega}\left(\left(1 - \eta_1\right)\tau_A + \tau_{r1}\right) + \left(1 - \xi_{\tau\omega}\right)\left(\left(1 - \tfrac{1}{\eta_2}\right)\tau_A + \tau_{r2}\right) \tag{2.11c}$$

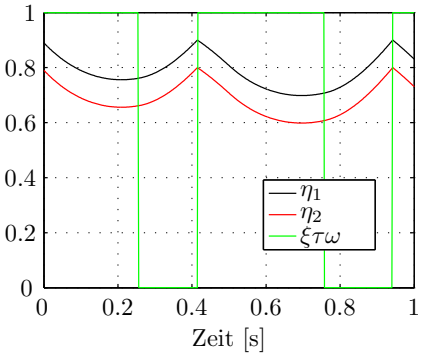
Abbildung 2.4 zeigt ein einfaches Beispielsystem und Simulationsergebnisse zur Anwendung des Wirkungsgrad-Modells. Das System besteht aus zwei Trägheiten (Θ_A, Θ_B) und dem Getriebemodell. An der ersten Trägheit Θ_A liegt ein rampenförmig verlaufendes Moment τ an. Zwischen den beiden Trägheiten, gekoppelt über Flansch A und B,



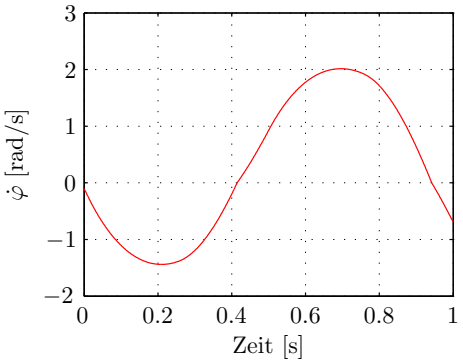
(a) Schematischer Aufbau der Simulation



(b) Momentverläufe



(c) Wirkungsgrade



(d) Geschwindigkeit Flansch A

Abbildung 2.4: Simulationsergebnisse zum approximierten Wirkungsgradmodell nach Gl. (2.11). Die Parameter des Systems sind: $\Theta_A = 1kg \cdot m^2; \Theta_B = 5kg \cdot m^2; i = 1; \eta_1 = 0.9 - 0.1|\omega_A|\frac{s}{rad}; \eta_2 = 0.8 - 0.1|\omega_A|\frac{s}{rad}; \tau_h = 5Nm; \tau_v = 5\frac{Nm}{rad}$

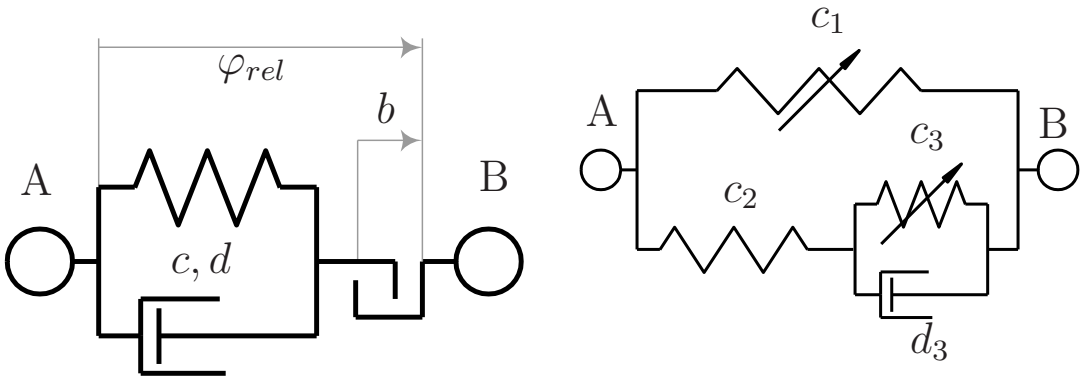
sitzt das Getriebemodell (ohne Elastizität) nach Gl. (2.11). Die Reibung des Getriebes ist nach Gl. (2.3) modelliert mit $\tau_{r1} = \tau_{r2} = \tau_r(\varphi_A, \tau_h, \tau_v)$.

2.3.4 Lose

Die Lose der verbauten Getriebe ist sehr klein, weshalb sie meist vernachlässigt werden kann. Soll sie dennoch mit modelliert werden, kann Gleichung (2.12) verwendet werden, wobei $c = c(\varphi)$ wieder analog zu Gleichung (2.7) modelliert werden kann. Abbildung 2.5(a) zeigt den schematischen Aufbau des Systems. $\varphi_{rel} = \varphi_B - \varphi_A$ steht dabei für den relativen Winkel zwischen Flansch A und B, b ist das Spiel des Getriebes, φ_0 der Startwinkel der Feder. Diese Gleichung ist allerdings nicht stetig differenzierbar aufgrund der Strukturumschaltung für $|\varphi_{rel}| < \frac{b}{2}$.

$$\tau = \begin{cases} c(\varphi_{rel} - \varphi_0 - \frac{b}{2}) + d\dot{\varphi}_{rel}, & \text{wenn } \varphi_{rel} > \frac{b}{2} \\ c(\varphi_{rel} - \varphi_0 + \frac{b}{2}) + d\dot{\varphi}_{rel}, & \text{wenn } \varphi_{rel} < -\frac{b}{2} \\ 0, & \text{wenn } -\frac{b}{2} \leq \varphi_{rel} \leq \frac{b}{2} \end{cases} \tag{2.12}$$

Ein alternativer approximativer Ansatz ist, das Getriebe als viskoselastische Feder-Dämpfer-Kombination zu modellieren (aus der Helikoptertechnik, siehe [27]). Abbil-



(a) Lose in Kombination mit Feder-Dämpfer-System.

(b) Viskoselastisches Feder-Dämpfer-Modell.

Abbildung 2.5: Modellvarianten zur Getriebemodellierung.

dung 2.5(b) stellt den Aufbau dieses Systems dar. Die nichtlineare Feder $c_1 = c_1(\varphi_{rel})$ modelliert dabei die Steifigkeit der Struktur. Mit der weichen Feder c_2 kann die Verformung des Systems bei kleinen Lasten approximiert werden, wodurch die Auswirkung von Lose nachgebildet werden kann. Durch die Parallelschaltung einer Feder mit der Steifigkeit $c_3 = c_3(\varphi_{rel})$ und einem linearen Dämpfer mit Parameter d kann die viskoselastische Dämpfung modelliert werden. Die Steifigkeit c_3 ist hierbei deutlich weicher als c_1 zu wählen (daher $c_1 \gg c_3 > 0$). Das gesamte System ist durch seine Parametrisierungen sehr flexibel einstellbar und auch Grenzzyklen können damit simuliert werden. Nachteilig ist allerdings, dass die Parameter des Modells nur sehr schwer getrennt identifiziert werden können. Die sich ergebenden Gleichungen dieses Systems sind jedoch stetig differenzierbar und invertierbar.

2.3.5 Hysterese

Die in den Roboter verbauten Getriebe weisen im Allgemeinen nur eine sehr geringe Hysterese zwischen wirkender Kraft und Verformung auf. Sie kann daher für rechenzeitkritische Echtzeitmodelle vernachlässigt werden. Soll die Hysterese der Getriebe modelliert werden, so ist darauf zu achten, dass die Modellierung auch zur Ableitung von inversen Modellen geeignet ist und damit auch das Hysteresemodell selbst invertierbar ist. Hystereseeffekte treten nicht nur durch Materialeigenschaften, sondern auch durch die Kombination von Haftreibung und Elastizitäten auf.

In [28], [29] und [30] wird zur Modellierung von Hystereseeffekten von Piezokeramik eine operatorbasierte Modellbildung vorgeschlagen. Grundlage dieser operatorbasierten Modellierung ist der sog. Play-Operator $\mathfrak{P}_{r_h}$. Zeitdiskret ($k = 0, 1, 2, \dots$) kann dieser in der Form von Gl. (2.13) geschrieben werden.

$$y(k) = \mathfrak{P}_{r_h}[x, y_{\mathfrak{P}_0}](k) \tag{2.13a}$$

$$= \begin{cases} \max\{x(k) - r_h, \min\{x(k) + r_h, y(k-1)\}\} & , k > 0 \\ \max\{x(0) - r_h, \min\{x(0) + r_h, y_{\mathfrak{P}_0}\}\} & , k = 0 \end{cases} \tag{2.13b}$$

Hierin ist $y_{\mathfrak{P}_0}$ der Startwert für $k = 0$ und r_h der Parameter für die (halbe) Hysterese-

breite, wobei $r_h \geq 0$ gewählt werden muss. Der sog. Prandtl-Ishlinskii-Hystereseoperator $\mathfrak{H}$ entsteht durch eine gewichtete Summe über mehrere Play-Operatoren $\mathfrak{P}_{r_{hi}}$ (Gl. (2.14)).

$$\mathfrak{H}[x](k) = \sum_{i=0}^{n} w_{hi} \mathfrak{P}_{r_{hi}}[x, y_{\mathfrak{P}_{0i}}](k) \tag{2.14}$$

Wie in [28], [29] und [30] gezeigt wurde eignet sich dieser Operator um Hysterese zu modellieren, wenn eine geeignete Ordnung n des Operators gewählt wird. Die Parametervektoren $\boldsymbol{w}_h = (w_{h0}, w_{h1}, \ldots, w_{hn})^T$, $\boldsymbol{r}_h = (r_{h0}, r_{h1}, \ldots, r_{hn})^T$ und $\boldsymbol{y}_{\mathfrak{P}_0} = (y_{\mathfrak{P}_0 0}, y_{\mathfrak{P}_0 1}, \ldots, y_{\mathfrak{P}_0 n})^T$ müssen dabei über eine Optimierung anhand von Messwerten ermittelt werden (siehe auch Kapitel 4). Wie in [28] hergeleitet lässt sich der Prandtl-Ishlinskii-Hystereseoperator $\mathfrak{H}$ invertieren. Unter der Voraussetzung, dass die Bedingungen aus Gl. (2.15) erfüllt sind

$$r_{hi} \geq 0 \wedge w_{h0} > 0 \wedge w_{hi} \geq 0, \text{ für } i = 1, \ldots, n \tag{2.15}$$

existiert der inverse Prandtl-Ishlinskii-Hystereseoperator $\mathfrak{H}^{-1}$. Der inverse Operator ist wieder ein Prandtl-Ishlinskii-Hystereseoperator, allerdings mit geänderten Parametervektoren $\boldsymbol{w}'_h$, $\boldsymbol{r}'_h$ und $\boldsymbol{y}'_{\mathfrak{P}}$. Diese ergeben sich nach Gl. (2.16).

$$i = 0:$$

$$w'_{h0} = \frac{1}{w_{h0}}, r'_{h0} = 0, y'_{\mathfrak{P}_0 0} = \sum_{j=0}^{n} w_{hj} y_{\mathfrak{P}_0 j} \tag{2.16a}$$

$$i = 1, \ldots, n:$$

$$w'_{hi} = -\frac{w_{hi}}{\left(w_{h0} + \sum_{j=1}^{i} w_{hj}\right)\left(w_{h0} + \sum_{j=1}^{i-1} w_{hj}\right)} \tag{2.16b}$$

$$r'_{hi} = \sum_{j=0}^{i} w_{hj}(r_{hi} - r_{hj}), y'_{\mathfrak{P}_0 i} = \sum_{j=0}^{i} w_{hj} y_{\mathfrak{P}_0 i} + \sum_{j=i+1}^{n} w_{hj} y_{\mathfrak{P}_0 j} \tag{2.16c}$$

In [28] werden zusätzlich zum Prandtl-Ishlinskii-Hystereseoperator noch Superpositionsoperator und ein Kriechoperator vorgeschlagen. Der Superpositionsoperator kann in Kombination (Reihenschaltung) mit dem Prandtl-Ishlinskii-Hystereseoperator verwendet werden, um unsymmetrische Hystereseverläufe zu modellieren. Der Kriechoperator dient zu Modellierung von Kriecheffekten des modellierten Materials. Diese Effekte spielen bei mechanischen Getrieben nur eine sehr kleine Rolle, weshalb sie hier nicht weiter betrachtet werden.

Nachteilig an dem Prandtl-Ishlinskii-Hystereseoperator $\mathfrak{H}$ und seinem inversen Operator $\mathfrak{H}^{-1}$ ist, dass die Gleichungen nicht stetig differenzierbar sind und daher in komplexeren inversen Modelle – bei denen es nötig ist, das komplette sich ergebende Gleichungssystem differenzieren zu können (siehe Kapitel 3) – nicht eingesetzt werden können.

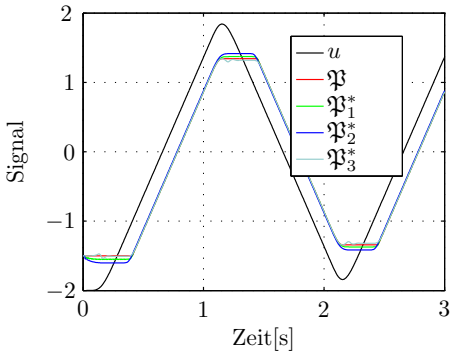
Um dieses Problem zu lösen kann eine kontinuierliche Approximation des diskreten Prandtl-Ishlinskii-Hystereseoperators verwendet werden. Diese soll hier mit $\hat{\mathfrak{H}}$ bezeichnet werden. Um $\hat{\mathfrak{H}}$ aufstellen zu können ist es nötig den Play-Operator zu approximieren. Der approximative Play-Operator soll $\mathfrak{P}^*$ genannt werden.

Die in Gleichung (2.13) verwendeten min- und max-Funktionen können stetig durch Gl. (2.17) approximiert werden.

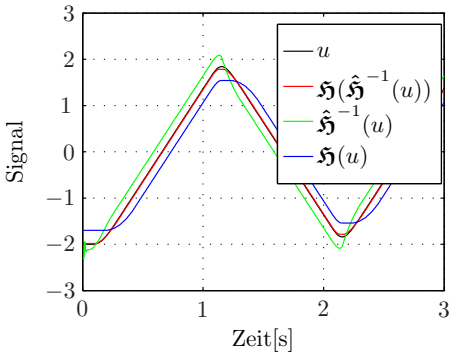
$$y_{max} = \max(u_1, u_2) \approx \frac{1}{\psi} \ln\left(e^{\psi u_1} + e^{\psi u_2}\right), \text{mit } \psi \gg 1 \tag{2.17a}$$

$$y_{min} = \min(u_1, u_2) = -\max(-u_1, -u_2) \approx -\frac{1}{\psi} \ln\left(e^{-\psi u_1} + e^{-\psi u_2}\right) \tag{2.17b}$$

Hierin ist $\psi \gg 1$ ein Parameter zum Einstellen der Güte der Approximation. Hohe Werte von ψ bedeuten zwar eine genauere Approximation, allerdings können durch zu hohe Werte von ψ numerische Probleme beim der Integration des DAE-Systems auftreten. Es muss für ψ daher ein Kompromiss eingegangen werden.



(a) Unterschiedlich parametrisierte stetige Approximationen $\mathfrak{P}^*$ des Play-Operators $\mathfrak{P}$.



(b) Kompensation von Hystereseeffekten mit stetiger (inverser) Approximation $\hat{\mathfrak{H}}^{-1}$ des Prandtl-Ishlinskii-Hystereseoperators $\mathfrak{H}$.

Abbildung 2.6: Simulationsergebnisse in *Dymola*. Durch das Vorschalten von $\hat{\mathfrak{H}}^{-1}$ kann die Hysterese, welche durch $\mathfrak{H}$ modelliert ist kompensiert werden. Die in der Simulation verwendeten Parameter listet Tabelle 2.6 auf. Das Eingangssignal $u(t)$ ist gefiltert, um die benötigten Differentiationen zu ermöglichen.

Approx.	**Parameter**
$\mathfrak{P}_1^*$	$T_s = 0.01$ s, $\psi = 50, n = 6$
$\mathfrak{P}_2^*$	$T_s = 0.01$ s, $\psi = 25, n = 3$
$\mathfrak{P}_2^*$	$T_s = 0.1$ s, $\psi = 25, n = 3$
$\mathfrak{H}$ und $\hat{\mathfrak{H}}^{-1}$	$T_s = 0.01$ s, $s = 25, n = 3$ $\boldsymbol{r}_h = (0.1, 0.2, 0.3, 0.4, 0.5)^T, \boldsymbol{w}_h = (0.2, 0.2, 0.2, 0.2, 0.2)^T$

Tabelle 2.1: Übersicht der Parameter der Approximationen in Abb. 2.6

Neben den min- und max-Funktionen bereitet auch der Faktor $y(k-1)$ in Gl. (2.13) Probleme bei der kontinuierlichen Simulation und insbesondere bei der direkten Invertierung in *Modelica*/*Dymola* (siehe Kapitel 3).

Eine Abhilfe hierfür ist die Approximation von $y(k-1)$ in Gl. (2.13) durch eine Padé-Approximation eines Totzeitgliedes. Die Padé-Approximation eines Totzeitgliedes ergibt sich als rationale Übertragungsfunktion mit Zählergrad n und Nennergrad

m durch eine Taylorentwicklung der Übertragungsfunktion des Totzeitgliedes $G_T(s)$ (mit Laplace-Variablen s). Eine schwingungsarme Form der Padé-Approximation ergibt sich für $m = n - 1$ nach Gl. (2.18) (für die Herleitung siehe [31]).

$$G_T(s) = \frac{1}{e^{T_t s}} \approx \frac{1 + \sum_{i=1}^{n-1} b_i \left(T_t s\right)^i}{1 + \sum_{i=1}^{n} a_i \left(T_t s\right)^i} \tag{2.18a}$$

$$a_i = \binom{n}{i} \frac{1}{(2n-1)(2n-2)\dots(2n-i)}, i = 1, \dots, n \tag{2.18b}$$

$$b_i = (-1)^i \binom{n-1}{i} \frac{1}{(2n-1)(2n-2)\dots(2n-i)}, i = 1, \dots, n-1 \tag{2.18c}$$

Für die Totzeit T_t muss dabei ein (für das System) ausreichend kleiner Wert gewählt werden, damit eine gute Näherung für $y(k-1)$ erreicht wird (wobei $T_t > 0$ bleiben muss).

Durch diese beiden Näherungen ergibt sich der approximative Hystereseoperator $\hat{\mathfrak{H}}[x](t)$ unter Verwendung von $\mathfrak{P}^*$. Dieser kann direkt stetig differenziert werden und ist auch (automatisch) durch *Modelica/Dymola* invertierbar (siehe Abschnitt 3.4), ohne die direkte Verwendung von Gl. (2.16). Durch die Approximation entstehen allerdings kleine Fehler. Die Abweichungen können durch das Erhöhen der Parameter n und ψ und Absenken von T_t verringert werden.

Einige Ergebnisse für unterschiedliche Approximationen des Play-Operators und des Prandtl-Ishlinskii-Hystereseoperators zeigt Abb. 2.6 wobei entsprechende Parameterwerte in Tabelle 2.1 aufgelistet werden. Wie man erkennen kann wird $\mathfrak{P}$ gut durch $\mathfrak{P}^*$ approximiert, wenn die Parameter entsprechend gewählt werden. Ein aus $\mathfrak{P}^*$ aufgebauter approximativer inverser Prandtl-Ishlinskii-Hystereseoperator $\hat{\mathfrak{H}}^{-1}$ genügt um ein Prandtl-Ishlinskii-Hystereseoperator $\mathfrak{H}$ mit hoher Genauigkeit invertieren zu können. In einem Getriebemodell kann der approximierte Operator $\hat{\mathfrak{H}}$ als Teil der Getriebesteifigkeit eines Feder-Dämpfer-Ersatzsystems verwendet werden. Gl. (2.7) kann durch einen entsprechenden Term erweitert werden (Gl. (2.19)).

$$\tau(\dot{\varphi}_{rel}, \varphi_{rel}) = \hat{\mathfrak{H}}(\varphi_{rel}) \left(c_0 \left(1 + \gamma \varphi_{rel}^2\right)\right) + d\dot{\varphi}_{rel} \tag{2.19}$$

Nachteilig an dieser Form der Modellierung ist, dass sie nicht physikalisch ist. Das Modell von $\hat{\mathfrak{H}}$ bzw. das Gesamtsystemverhalten kann durch die Verwendung von $\hat{\mathfrak{H}}$ physikalisch unplausibel beeinflusst werden. Es ist daher nötig, das Verhalten anhand des Vergleichs von Messdaten und Simulationsergebnissen ausgiebig zu verifizieren.

Eine exakte physikalische Modellierung ist im Rahmen eines Modells, welches zur Vorsteuerung eingesetzt werden soll, allerdings kaum möglich. Physikalisch entsteht Hysterese, unter anderem, aus dem komplizierten dynamischen Kontaktverhalten der Zähne im Getriebe. Hier spielt insbesondere die Kombination von (Haft-) Reibung und Elastizität eine entscheidende Rolle. Um die komplette Bandbreite der verursachenden Effekte abzudecken wären sehr aufwändige Getriebemodelle nötig, welche die verfügbare Rechenzeit und Anforderungen an die Invertierbarkeit nicht erfüllen könnten.

Daher stellt Gl. (2.19) einen guten Kompromiss zwischen Genauigkeit und benötigter Rechenzeit dar.

2.4 Elastische Struktur des Roboters

Zusätzlich zu den Antriebssträngen des Roboters, soll auch die Struktur des Roboters elastisch modelliert werden. In der klassischen Industrieroboter-Modellierung (z. B. [14]) wird diese meist als starr angenommen. Der Trend in der Industrierobotik geht allerdings mehr und mehr zu leichteren Bauweisen, was in der Folge meist auch zu niedrigeren Steifigkeiten der Bauteile führt. Hiervon sind insbesondere lange Bauteile eines Roboters betroffen. Zudem sind die Lagerungen und Getriebe der einzelnen Achsen nicht perfekt steif, so dass auch Verformungen orthogonal zu Rotationsachsen auftreten können. Im Fall der hier untersuchten Roboter KR16 und KR210 ist ein kri-

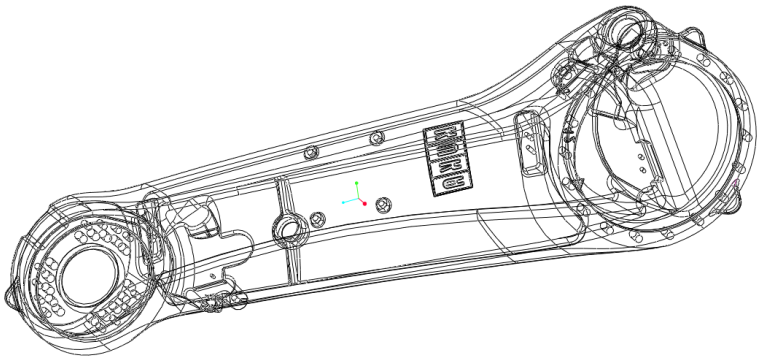


Abbildung 2.7: CAD-Modell einer Schwinge, das Glied zwischen der zweiten und dritten Achse.

tisches Bauteile die sogenannte Schwinge (Abb. 2.7), das verbindende Glied zwischen der zweiten und dritten Achse und der daran anschließende Arm. Unter hohen Belastungen, die bei hochdynamischen Bewegungen unter hoher Last vorkommen, können sich Verformungen zeigen, welche das dynamische Verhalten des Roboters beeinflussen.

Steifigkeiten orthogonal zur Rotationsrichtung einer Achse werden im Folgenden mit Kippsteifigkeiten der Achse bezeichnet (siehe Abb. 2.8). Diese resultieren hauptsächlich aus der Lagerung und dem Aufbau der Getriebe.

Im Folgenden sollen einige Modellierungsansätze vorgestellt werden, um die elastische Struktur des Roboters (getrennt von den Elastizitäten der Antriebsstränge) zu modellieren.

2.4.1 Kippsteifigkeit einer Achse

Die Kippsteifigkeit einer Achse kann über zusätzliche Gelenke mit Feder-Dämpfer-Elemente modelliert werden. Unter Kippsteifigkeit wird hierbei die Steifigkeit der Gelenke orthogonal zur Rotationsachse verstanden, welche durch den Aufbau der Lagerung und der mechanischen Bauteile bedingt ist.

Diese zusätzlichen Gelenke werden im Folgenden als Kippgelenke bezeichnet. Nach Konvention in dieser Arbeit ist die Drehrichtung einer Achse die z-Richtung ($\boldsymbol{e}_z = \{0, 0, 1\}$) im lokalen Koordinatensystem, die Kippsteifigkeit wird daher durch zwei zusätzliche Gelenke in x- ($\boldsymbol{e}_x = \{1, 0, 0\}$) und y-Richtung ($\boldsymbol{e}_y = \{0, 1, 0\}$) modelliert (siehe Abb. 2.8).

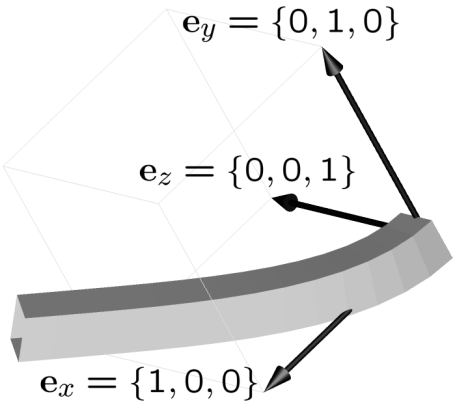


Abbildung 2.8: Anordnung der lokalen Koordinatensysteme für die Roboter-Bauteile. Die Roboter-Gelenke erlauben (nach Konvention) eine Rotation um die $\boldsymbol{e}_z$-Achse.

Die Feder-Dämpfer-Elemente können linear oder nichtlinear (analog zu Gl. (2.7)) definiert werden.

2.4.2 Balken

Lange Bauteile des Roboters, wie die Schwinge, können als elastischer Balken modelliert werden. In der Literatur existieren unterschiedliche Ansätze, um einen Balken mathematisch zu beschreiben (für eine Übersicht sei z. B. auf [32] und [33] verwiesen). Da die Verformungen relativ klein sind können approximative Theorien verwendet werden. Zur Simulation des Balkens wird die *Modelica FlexibleBodies Library* (MFBL) verwendet (siehe auch [34]). Die mechanische Beschreibung basiert hierbei auf dem Ansatz

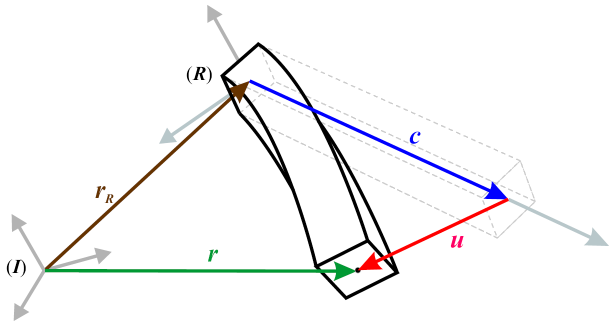


Abbildung 2.9: Referenz System eines elastischen Balkens (aus [34]).

des mitbewegten Referenzkoordinatensystems (englisch: „floating frame of reference approach“). Hierbei werden die Positionsvektoren $\boldsymbol{r}_t$ zu den einzelnen Masseelementen dm in drei Teile aufgespalten (siehe Gleichungen[3] (2.20) sowie Abbildung 2.9). $\boldsymbol{v}$ ist hierbei der Vektor der Geschwindigkeit und $\boldsymbol{a}$ der Vektor der Beschleunigung eines Masseelements dm. Orientierungen werden wahlweise auf Basis von Quaternionen

[3] Der $\tilde{(\cdot)}$-Operator steht hierbei für eine schiefsymmetrische Matrix zur Realisierung des Kreuzprodukts als Matrixmultiplikation (Details im Anhang A.4)

(siehe A.8.4) oder Rotationsmatrizen (siehe A.8.1) berechnet.

$$\begin{aligned} \boldsymbol{r}_t &= \boldsymbol{r}_R + \boldsymbol{c} + \boldsymbol{u} && \text{(2.20a)} \\ \boldsymbol{v} &= \boldsymbol{v}_R + \tilde{\omega}_{\mathbf{R}}(\mathbf{c} + \boldsymbol{u}) + \dot{\mathbf{u}} && \text{(2.20b)} \\ \boldsymbol{a} &= \boldsymbol{a}_R + (\dot{\tilde{\omega}}_{\mathbf{R}} + \tilde{\omega}_{\mathbf{R}}\tilde{\omega}_{\mathbf{R}})(\boldsymbol{c} + \boldsymbol{u}) + 2\tilde{\omega}_{\mathbf{R}}\dot{\mathbf{u}} + \ddot{\mathbf{u}} && \text{(2.20c)} \end{aligned}$$

Durch die Aufteilung der Koordinaten ist es möglich große Starrkörperbewegungen mit überlagerten kleinen elastischen Verformungen effizient zu simulieren. Die Verschiebungen im Raum werden durch Taylorentwicklungen zweiter Ordnung mit raumabhängigen Modalformen $\boldsymbol{\Phi}(\boldsymbol{c})$ und zeitabhängigen, modalen Amplituden $\boldsymbol{q}(t)$ approximiert.

$$\boldsymbol{u} = \boldsymbol{\Phi}\boldsymbol{q} + \frac{1}{2}\begin{pmatrix} \boldsymbol{q}^T\boldsymbol{\Phi}_x \\ \boldsymbol{q}^T\boldsymbol{\Phi}_y \\ \boldsymbol{q}^T\boldsymbol{\Phi}_z \end{pmatrix}\boldsymbol{q} \tag{2.21}$$

Die Bewegungsgleichungen für den Balken können, wie in [34] gezeigt wurde, dann in Form von Gleichung (2.22) aufgestellt werden.

$$\begin{pmatrix} m\mathbf{I_3} & & sym. \\ m\tilde{\mathbf{d}}_{\mathbf{CM}} & \mathbf{J} & \\ \mathbf{C_t} & \mathbf{C_r} & \mathbf{M_e} \end{pmatrix}\begin{pmatrix} \mathbf{a_R} \\ \dot{\omega}_{\mathbf{R}} \\ \ddot{\mathbf{q}} \end{pmatrix} = \mathbf{h}_\omega - \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{K_e q} + \mathbf{D_e} \end{pmatrix} + \boldsymbol{h}_{\mathbf{ext}} \tag{2.22}$$

Die Abkürzungen in Gleichung (2.22) haben die folgende Bedeutung:

- m : Masse des Körpers
- $\mathbf{I_3}$: 3x3-Einheitsmatrix
- $\mathbf{d_{CM}}$: Position des Massemittelpunktes
- $\mathbf{J}$: Trägheitstensor
- $\mathbf{C_t}$: Inertiale Kopplungsmatrix (translatorisch)
- $\mathbf{C_r}$: Inertiale Kopplungsmatrix (rotatorisch)
- $\mathbf{h}_\omega$: Gyroskopische und Zentripetalkräfte
- $\boldsymbol{h}_{ext}$: Äußere Krafteinwirkung
- $\mathbf{M_e}$: Massenmatrix der elastischen Verformung
- $\mathbf{K_e}$: Strukturelle Steifigkeitsmatrix
- $\mathbf{D_e}$: Strukturelle Dämpfungsmatrix

Mit Hilfe der Theorie von Rayleigh und Bernoulli können die räumlichen Verschiebungen $\mathbf{u}$ analytisch bis zu Termen 2. Ordnung beschrieben werden. Über einen Ritz-Ansatz können dann die Verschiebungen in Richtung u_1, u_2 und u_3 sowie für die Torsion in Θ

über die analytischen Lösungen der Eigenformen eines Rayleigh-Balkens gefunden werden. Hierzu werden dann noch geeignete Randbedingungen, die durch die Lagerung des Balkens vorgegeben sind, sowie dynamische Anfangsbedingungen benötigt.

Für die Schwinge und den Arm des Roboters können die Randbedingungen eingespannt-frei ($\mathbf{u}(\mathbf{c}=\mathbf{0})=\mathbf{0}$ sowie $\mathbf{u}'(\mathbf{c}=\mathbf{0})=\mathbf{0}$), sowie unterstützt-frei ($\mathbf{u}(\mathbf{c}=\mathbf{0})=\mathbf{0}$) gewählt werden.

Weitere Details zu Gl. (2.22) finden sich in Abschnitt 3.7, wo auch auf die Invertierung von Systemen, die mit Balkenelementen modelliert wurden, eingegangen wird.

Zusätzliche Informationen zur *Modelica FlexibleBodies Library* finden sich in [34].

2.4.3 Balken-Ersatzmodelle

Für Echtzeit-Modelle reicht die Rechenzeit - je nach Komplexität des übrigen Modells und der eingesetzten CPU - nicht immer aus um einen Balken (insbesondere als Teil eines inversen Robotermodells), nach den Gleichungen die im vorherigen Abschnitt 2.4.2 beschrieben wurden, zu berechnen.

Es wurden daher Ersatzmodelle für Balken modelliert, mit welchen die wesentliche Dynamik des Balkens gut approximiert werden kann und welche sich zudem für eine Invertierung eignen (mehr dazu in Kapitel 3).

Eine Variante zur Approximation des Balkens ist, die Struktur in einzelne Starrkörper aufzuteilen, die mit Gelenken verbunden sind, welche durch Feder-Dämpfer-Elemente gestützt werden. Die Modellierung basiert dabei auf [35].

Die Massen und Trägheiten werden so aufgeteilt, dass das Gesamtsystem im unverformten Zustand identisch mit dem zu approximierenden Bauteil ist (Schwerpunkt sowie Trägheit des approximativen Modells bezüglich der Einspannung stimmen überein). Abbildung 2.10 zeigt Schematisch den Aufbau der Mehrkörper-Approximation eines Balkens mit fünf Freiheitsgraden. Gleichungen (2.23) beschreiben die Aufteilung der Masse und Längen auf die einzelnen Teilkörper (siehe auch Abb. 2.10).

$$
\begin{aligned}
k &= \frac{1-\frac{1}{\sqrt{3}}}{2} && \text{(2.23a)}\\
l_1 &= l_4 = kl && \text{(2.23b)}\\
l_2 &= l_3 = \left(\frac{1}{2}-k\right) l && \text{(2.23c)}\\
m_1 &= mk + \Delta_m && \text{(2.23d)}\\
m_2 &= m_3 = \left(\frac{1}{2}-k\right) m && \text{(2.23e)}\\
m_4 &= mk - \Delta_m && \text{(2.23f)}\\
\Delta_m &= \frac{m}{2}\frac{2l_s - l}{l(k-1)} && \text{(2.23g)}
\end{aligned}
$$

Die Trägheit des Balkens wird ebenfalls auf die vier Einzelmassen m_i aufgeteilt. Hierzu wird angenommen, dass es sich bei dem Balken um einen homogenen Hohlkörper handelt und nur die Diagonalelemente des Trägheitstensors $\mathbf{\Theta}_B^{(S)}$, im lokalen Koordina-

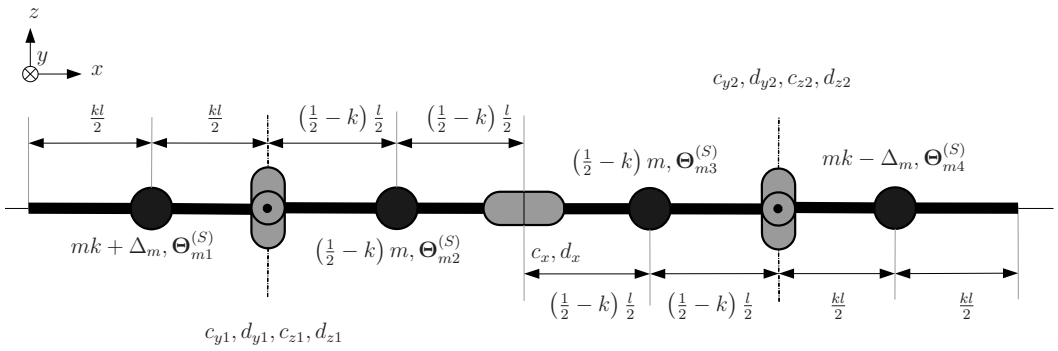


Abbildung 2.10: Schematische Darstellung der Mehrkörper-Approximation eines Balkens mit fünf Freiheitsgraden. Die Gelenke werden von Federn und Dampfern (Parameter $c_{y1}, d_{y1}, c_{z1}, d_{z1}, c_{y2}, d_{y2}, c_{z2}, d_{z2}, c_x, d_x$) gestützt. Das Modell verfügt über vier Starrkörper sowie zwei Doppelgelenke (Rotation um y und z) sowie ein Torsionsgelenk (Rotation um x) in der Mitte des Elements.

tensystem bezüglich des Schwerpunktes des Balkens, besetzt sind. Die Aufteilung der Trägheit des Balkens $\mathbf{\Theta}_B^{(S)}$ auf die Trägheit der Einzelmassen $\mathbf{\Theta}_{m,i}^{(S)}$ erfolgt dann proportional zu den Längen der Segmente, wobei zunächst ein Ersatzradius r_{er} berechnet wird (Gleichung (2.24)). Θ_{st} ist der Steineranteil der Trägheit durch die Verschiebung l_s des Schwerpunktes, durch die Annahme eines homogenen Körpers.

$$\mathbf{\Theta}_B^{(S)} = \begin{pmatrix} \Theta_{xx} & 0 & 0 \\ 0 & \Theta_{yy} & 0 \\ 0 & 0 & \Theta_{zz} \end{pmatrix} \tag{2.24a}$$

$$\Theta_{st} = \left(\frac{l}{2} - l_s\right)^2 m \tag{2.24b}$$

$$r_{er} = \sqrt{\frac{\Theta_{yy} + \Theta_{zz} - 2\Theta_{st}}{m} - \frac{l^2}{6}} \tag{2.24c}$$

$$\mathbf{\Theta}_{m1}^{(S)} = \mathbf{\Theta}_{m4}^{(S)} = k \begin{pmatrix} \Theta_{xx} & 0 & 0 \\ 0 & m\frac{(kl)^2+6r_{er}^2}{12} & 0 \\ 0 & 0 & m\frac{(kl)^2+6r_{er}^2}{12} \end{pmatrix} \tag{2.24d}$$

$$\mathbf{\Theta}_{m2}^{(S)} = \left(\frac{1}{2} - k\right) \begin{pmatrix} \Theta_{xx} & 0 & 0 \\ 0 & \frac{m(((\frac{1}{2}-k)l)^2+6r_{er}^2)}{12} & 0 \\ 0 & 0 & \frac{m(((\frac{1}{2}-k)l)^2+6r_{er}^2)}{12} \end{pmatrix} \tag{2.24e}$$

$$\mathbf{\Theta}_{m3}^{(S)} = \mathbf{\Theta}_{m2}^{(S)} \tag{2.24f}$$

Eine Variante zur Approximation eines Balkens kann dadurch erreicht werden, dass eine quasistatische Abschätzung der Krümmung am Balkenende berechnet wird. Hierdurch können Freiheitsgrade eingespart werden. Dazu wird die Masse m und der Trägheitstensor bezüglich des Schwerpunktes $\mathbf{\Theta}^{(S)}$ des Balkens durch eine Ersatzmasse und Trägheitstensor modelliert. Zusätzlich werden drei Drehgelenke am Ort der Einspannung des Balkens für die Modellierung der Steifigkeit des Balkens verwendet, welche durch Feder-Dämpferelemente (Parameter $c_x, d_x, c_y, d_y, c_z, d_z$) gestützt werden (Abb. 2.11). Aus den Verformungen dieser Gelenke können die Krümmungswinkel α_i

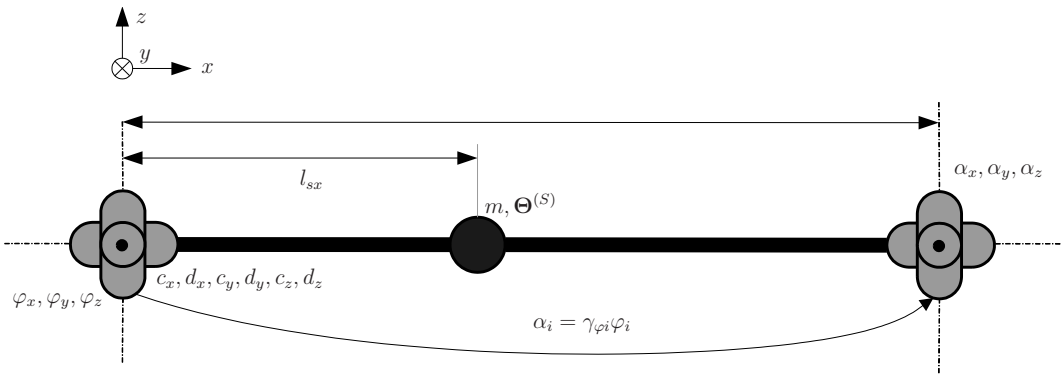


Abbildung 2.11: Schematische Darstellung der Mehrkörper Approximation eines Balkens mit drei Freiheitsgrade, sowie drei weiteren gekoppelten, abhängigen Freiheitsgraden zur Berücksichtigung der Krümmung am Balkenende.

am Balkenende geschätzt werden. Motiviert ist dieser Ansatz durch die statische Betrachtungsweise eines Balkens nach Abb. 2.12. Wird die Steifigkeit des Balkens durch

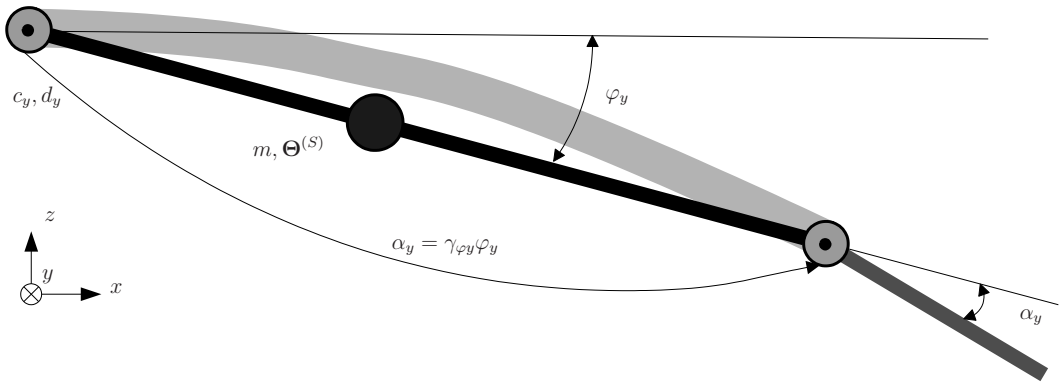


Abbildung 2.12: Statische Betrachtungsweise für die Durchbiegung in z-Richtung: Die Approximation der Durchbiegung eines Balkens (Biegelinie als dicke graue Line in der Abbildung) mit Hilfe eines durch Feder und Dämpfer gestützten Gelenks (Parameter c_y, d_y) und einem Starrkörper führt dazu, dass am Balkenende eine Winkelabweichung für den folgenden Körper in der Kinematischen Kette des Roboter auftritt. Dies kann approx. kompensiert werden, indem ein zusätzliches Gelenk am Balkenende verwendet wird, dessen Winkel in Abhängigkeit des ersten Gelenks zu $\alpha_y = \gamma_{\varphi y}\varphi_y$ gesetzt wird.

Federn am Ort der Einspannung modelliert, so kann die Krümmung am Balkenende dadurch approximiert werden, dass zusätzliche Gelenke am Balkenende verwendet werden. Die Größe der Winkel der zusätzlichen Gelenke am Ende des Ersatzbalkenmodells α_i wird durch die Multiplikation des jeweiligen Winkels des Ersatzbalkens am Ort der Einspannung $\varphi_i(x_l = 0)$ mit einem konstanten Faktor $\gamma_{\varphi i}$ approximiert. Die Schätzung des Winkels am Balkenende ist vor allem dann interessant, wenn lange Werkzeuge am Roboter eingesetzt werden oder um die Genauigkeit der Approximation zu erhöhen. Bei hohen Steifigkeiten und kurzen Strukturen ist der Effekt weniger wichtig und kann gegebenenfalls vernachlässigt werden.

2.5 Gewichtsausgleich

Der Roboter KR210 verfügt über einen Gewichtsausgleich (GWA). Hierbei handelt es sich um eine vorgespannte Feder welche den Motor, an Achse 2 des Roboters, entlastet. Hierdurch wird vor allem das benötige Motormoment τ_m, wenn sich der Roboter in Strecklage befindet, reduziert. Abbildung 2.13 zeigt den Gewichtsausgleich am KR210, wobei die Feder in einer schwarzen Hülle eingebettet ist. Es gibt auch Varianten bei denen keine Feder, sondern Gasdruck für die Vorspannung sorgt. Der GWA wurde

Abbildung 2.13: Gewichtsausgleich (GWA) des KR210 (Abbildung: KUKA).

als (optional massebehaftete) Feder mit Dämpfer modelliert. Durch die leicht seitliche Anbringung des Hebelarms des GWAs erzeugt dieser auch ein Moment auf die Roboter-Struktur orthogonal zu Achse 2. Wird dieser Effekt vernachlässigt kann das Moment, welcher der GWA auf die Achse 2 ausübt, auch über eine einfache Gleichung, ohne Dynamik in Abhängigkeit der Stellung von Achse 2, berechnet werden. Diese Näherung ist meist zulässig, da der GWA keinen wesentlichen Effekt auf die Dynamik des gesamten Roboters hat.

2.6 Modellierung des Mehrkörpersystems

Die Modellierung der mechanischen Struktur erfolgt durch elastische Mehrkörpersysteme. Für die Modellierung der Steifigkeiten der Struktur können dabei verschiedene Ansätze, je nach verfügbarer Rechenzeit und gewünschter Genauigkeit, verfolgt werden. Wie in Abschnitt 2.4.3 gezeigt, können Strukturelastizitäten durch zusätzliche Gelenke, welche durch Feder-Dämpfer-Systeme gestützt werden, approximiert werden. Die einzelnen Segmente des Roboters können dann als Starrkörper mit Masse m_i, Schwerpunkt S_i und Trägheitstensor $\mathbf{\Theta}_i^{(S)}$ angenommen werden. Alternativ können die Bauteile auch als elastische Körper wie Balken modelliert werden. Spielt die Rechenzeit eine untergeordnete Rolle können Bauteile, nach einer Modalanalyse der FEM-Daten von CAD-Modellen, über SID-Datensätze in *Modelica* mit Hilfe der *FlexibleBodies Library* modelliert werden. Siehe hierzu auch [34] sowie [36].

SID ist die Abkürzung für „Standard Input Data“. Die Struktur der SID-Datensätze wird in [36] definiert. SID-Datensätze basieren auf einer modalen Beschreibung flexibler Körper für kleine Verformungen in Kombination mit großen Starrkörperbewegungen.

Die Daten können für Balken analytisch abgeleitet werden oder aus FEM-Daten durch Preprozessoren gewonnen werden. Die SID-Struktur ist objektorientiert aufgebaut und die einzelnen Matrizen – welche das System beschreiben – sind dabei als Taylorentwicklungen (bis zur zweiten Ordnung) bezüglich der elastischen Freiheitsgrade gespeichert. Die Matrizen enthalten vorausberechnete Integrale bezüglich der ortsabhängigen Modalformen der Körper.

Bei der Modellierung des Roboters ist es sinnvoll, die Kopplungen der Dynamik der Antriebsstränge von der Dynamik der mechanischen Struktur über Kreiselkräfte, welche über die rotierenden Antriebsstrangträgheiten entstehen, zu vernachlässigen. Dies vereinfacht die Komplexität der Gleichungen deutlich ohne hohen Genauigkeitsverlust.

Hierzu wird die Annahme getroffen, dass die Antriebsstränge symmetrisch bezüglich der jeweiligen Rotationsachse sind und dass gyroskopische Effekte auf die mechanische Struktur, die durch die Rotation der Trägheiten der Antriebsstränge entstehen, vernachlässigt werden können. Zwar ist die Berechnung der gyroskopischen Effekte, wie in [37] gezeigt wurde, auch bei der Verwendung von 1D-Antriebsstrangmodellen möglich, jedoch haben Simulationen gezeigt, dass diese um Größenordnungen kleiner sind als andere dynamische Effekte am Roboter und daher vernachlässigt werden können. Diese Näherung ist zulässig, wenn eine hohe Getriebeübersetzung existiert (siehe [37],[15] sowie [10]).

Zum Aufstellen der Bewegungsgleichungen dieser Struktur gibt es verschiedene Vorgehensweisen. Ein effizienter Ansatz ist die Auswertung von Impuls- und Drallsatz in Kombination mit dem Prinzip von Jourdain (virtuelle Arbeit), welche auch als Projektionsgleichungen in der Literatur bekannt sind. Alternativ kann auch die Energiemethode nach Lagrange verwendet werden. Beide Ansätze werden in [38] und [32] ausführlich beschrieben.

Als erster Schritt muss in beiden Fällen zunächst die Kinematik der mechanischen Struktur aufgestellt werden. Dies erfolgt auf der Basis von homogenen Transformationsmatrizen ${}^{l}_{k}\boldsymbol{T}$. Durch diese ist es möglich Ort und Rotation der einzelnen Starrkörper zu berechnen. Die Indizes k und l stehen dabei für die Transformation von Frame l nach Frame k. Die Transformationsmatrizen $\boldsymbol{T}$ können dabei aufgeteilt werden in die Rotationsmatrix $\boldsymbol{R}$ sowie den Translationsvektor $\boldsymbol{r}_t$ (siehe Gleichung (2.25)). Durch die Eigenschaften einer Rotationsmatrix kann die inverse Transformation $\boldsymbol{T}^{-1}$ vereinfacht berechnet werden.

$$\boldsymbol{T} = \left(\begin{array}{c|c} \boldsymbol{R} & \boldsymbol{r}_t \\ \hline \boldsymbol{0} & 1 \end{array}\right), \; \boldsymbol{T}^{-1} = \left(\begin{array}{c|c} \boldsymbol{R}^T & -\boldsymbol{R}^T\boldsymbol{r}_t \\ \hline \boldsymbol{0} & 1 \end{array}\right) \tag{2.25}$$

Die Transformationsmatrizen der Gelenke $\boldsymbol{T}_R$ sind dabei nur von dem jeweiligen Gelenkwinkel $\boldsymbol{R} = \boldsymbol{R}(\varphi_i)$ abhängig und ihr Translationsvektor $\boldsymbol{r}_t$ ist der Nullvektor.

Zur Aufstellung der Bewegungsgleichungen müssen zunächst alle Transformationsmatrizen für alle Segmente, sowie zu allen Schwerpunkten der einzelnen Starrkörper aufgestellt werden. Durch das Aneinanderreihen der entsprechenden Transformationsmatrizen $\boldsymbol{T}$ ist es nun möglich Ort und Orientierung jedes Elementes i zu berechnen. Die Struktur des Roboters ist die einer offenen Kinematischen Kette. Durch die Hintereinanderschaltung der entsprechenden Transformationsmatrizen $\boldsymbol{T}$ kann so für jeden

Punkt des Roboters eine entsprechende Transformation gefunden werden.

$$ {}_0^k\boldsymbol{T} = {}_0^1\boldsymbol{T} \cdot {}_1^2\boldsymbol{T} \cdot \ldots \cdot {}_{k-1}^k\boldsymbol{T} \tag{2.26} $$

Zusätzlich müssen die Winkelgeschwindigkeiten ω_i der einzelnen Körper bezüglich den Schwerpunkten S_i berechnet werden. Hierbei wird ausgenutzt, dass Winkelgeschwindigkeiten, welche bezüglich dem gleichen Koordinatensystem aufgestellt sind, vektoriell addiert werden dürfen. Hierzu werden die einzelnen Winkelgeschwindigkeiten zuerst in das Basis-Koordinatensystem '0' transformiert, addiert und dann in das Koordinatensystem des jeweiligen Schwerpunktes transformiert (Gl. (2.27)).

$$ \boldsymbol{\omega}_i^{(0)} = {}_0^i\boldsymbol{R} \begin{pmatrix} \dot{\varphi}_{ix} \\ \dot{\varphi}_{iy} \\ \dot{\varphi}_{iz} \end{pmatrix} \tag{2.27a} $$

$$ \boldsymbol{\omega}_{m,i}^{(S)} = {}_0^{iC}\boldsymbol{R}^T \left(\boldsymbol{\omega}_1^{(0)} + \boldsymbol{\omega}_2^{(0)} + \ldots + \boldsymbol{\omega}_i^{(0)} \right) \tag{2.27b} $$

$\varphi_{ix}, \varphi_{iy}$ und φ_{iz} sind hierbei die entsprechenden Komponenten der Winkelgeschwindigkeit im lokalen Koordinatensystem (x, y, z) des entsprechenden Frames i. Eine effizientere Lösung kann erreicht werden, wenn die Geschwindigkeiten jeweils lokal ins nächste System transformiert werden.

Alternativ kann die Winkelgeschwindigkeit auch direkt aus der Ableitung der Rotationsmatrix nach Gleichung (2.28) gewonnen werden, was rechentechnisch jedoch nicht effizient ist, aber mit modernen Computer-Algebra-Systemen problemlos möglich ist.

$$ \tilde{\boldsymbol{\omega}}_i = {}_0^i\boldsymbol{R}^T \cdot {}_0^i\dot{\boldsymbol{R}} \tag{2.28} $$

Geschwindigkeiten und Beschleunigungen können ebenfalls aus den Transformationsmatrizen gewonnen werden. Hierzu werden die Vektoren zunächst in ein Inertialsystem transformiert, dort bezüglich der Zeit differenziert und dann wieder in das gewünschte Bezugsystem transformiert (Gl. (2.29)). Ein absolutes Zeitmaß erhält man nur, wenn man die zeitliche Änderung im Vergleich zu einer inertialen Basis beschreibt (siehe [32]).

$$ \boldsymbol{v}_i = \dot{\boldsymbol{r}}_{t,i} = {}_0^i\boldsymbol{R}^T \cdot \frac{d}{dt}\boldsymbol{r}_{t,0} \tag{2.29a} $$

$$ \boldsymbol{a}_i = \ddot{\boldsymbol{r}}_{t,i} = {}_0^i\boldsymbol{R}^T \cdot \frac{d^2}{dt^2}\boldsymbol{r}_{t,0} \tag{2.29b} $$

Sind die kinematischen Zusammenhänge ermittelt und alle benötigten Geschwindigkeiten und Beschleunigungen in den jeweils benötigten Koordinatensystemen (für Energiemethoden bezüglich des Inertialsystems; für die Newton-Euler-Jourdain Methode bezüglich der Schwerpunkte der einzelnen Körper) berechnet, können die Bewegungsgleichungen aufgestellt werden. Die zentrale Gleichung ist hierbei für die Newton-Euler-Jourdain (NEJ) Methode Gl. (2.30) (Projektionsgleichung nach [32]).

$$ \sum_{i=1}^{n} \int_{K_i} \left\{ \begin{pmatrix} \left(\frac{\partial_E \boldsymbol{v}_s}{\partial \dot{\boldsymbol{s}}}\right)_i \\ \left(\frac{\partial_E \boldsymbol{\omega}_{0E}}{\partial \dot{\boldsymbol{s}}}\right)_i \end{pmatrix}^T \begin{pmatrix} \left({}_E d\dot{\boldsymbol{p}} + {}_E\tilde{\boldsymbol{\omega}}_{0E} \cdot {}_E d\boldsymbol{p} - {}_E d\boldsymbol{f}^e\right)_i \\ \left({}_E d\dot{\boldsymbol{L}} + {}_E\tilde{\boldsymbol{\omega}}_{0E} \cdot {}_E d\boldsymbol{L} - {}_E d\boldsymbol{l}^e\right)_i \end{pmatrix} \right\} = 0 \tag{2.30} $$

Wird ein Ersatzsystem mit Starrkörpern und Feder-Dämpfer-Elementen gewählt, lässt sich Gleichung (2.30) weiter vereinfachen und man gelangt zu Gl. (2.31). Hierin wurde

als Bezugspunkt der jeweilige Schwerpunkt S gewählt, was auf eine einfache Gleichungsstruktur führt.

$$\sum_{i=1}^{n} \left\{ \begin{pmatrix} \left(\frac{\partial \boldsymbol{v}_{S,i}}{\partial \dot{\boldsymbol{q}}}\right)_i \\ \left(\frac{\partial \boldsymbol{\omega}_i}{\partial \dot{\boldsymbol{q}}}\right)_i \end{pmatrix}^T \begin{pmatrix} \left(m_i \dot{\boldsymbol{v}}_{S,i} - \boldsymbol{f}_i^e\right)_i \\ \left(\boldsymbol{\Theta}_i^{(S)} \dot{\boldsymbol{\omega}}_i + \tilde{\boldsymbol{\omega}}_i \boldsymbol{\Theta}_i^{(S)} \boldsymbol{\omega}_i - \boldsymbol{l}_i^e\right)_i \end{pmatrix} \right\} = 0 \tag{2.31}$$

In Gleichung (2.31) wurden die Minimalgeschwindigkeiten $\dot{\boldsymbol{s}}_i$ durch die Gelenkgeschwindigkeiten $\dot{\boldsymbol{q}}_i$ (welche auch die zusätzlichen Feder-Dämpfer-Elemente beinhalten) ersetzt, da für das vereinfachte (holonome) System keine Unterscheidung mehr getroffen werden muss.

Die Jacobi-Matrizen in Gleichung (2.30) und (2.31) müssen jeweils bezüglich der Geschwindigkeit im betrachteten lokalen Koordinatensystem aufgestellt werden.

Für die Berechnung nach Lagrange müssen die potentielle und kinetische Energie des Systems bestimmt werden. Die kinetische Energie für jeden einzelnen Starrkörper ergibt sich dabei nach Gleichung (2.32). Die Geschwindigkeiten müssen hierbei bezüglich eines inertialen Systems aufgestellt werden. In Gl. (2.32) stellt $\boldsymbol{\eta}_{S,i}$ den Vektor von Bezugspunkt Q zu Schwerpunkt S des jeweiligen Körpers da.

$$T_{kin,i} = \frac{m}{2} \left(\dot{\mathbf{r}}_{Q,t,i}^T \dot{\mathbf{r}}_{Q,t,i} + \boldsymbol{\omega}_i^T \boldsymbol{\Theta}_i^{(Q)} \boldsymbol{\omega}_i \right) + m_i \dot{\mathbf{r}}_{Q,t,i}^T \dot{\mathbf{r}}_{Q,t,i}^T (\tilde{\boldsymbol{\omega}}_i \boldsymbol{\eta}_{S,i}) \tag{2.32}$$

Da als Körper-Bezugspunkt Q der Massenmittelpunkt S gewählt wird, vereinfacht sich Gleichung (2.32) zu Gl. (2.33).

$$T_{kin,i} = \frac{1}{2} m_i \dot{\mathbf{r}}_{S,t,i}^T \dot{\mathbf{r}}_{S,t,i} + \frac{1}{2} \boldsymbol{\omega}_i^T \boldsymbol{\Theta}_i^{(S)} \boldsymbol{\omega}_i \tag{2.33}$$

Die potentielle Energie ergibt sich nach Gl. (2.34) für jeden einzelnen Starrkörper, wobei $\mathbf{g}$ den Gravitationsvektor bezeichnet.

$$V_{pot,i} = m_i \mathbf{g}^T \mathbf{r}_{S,t,i} \tag{2.34}$$

Die Energie, die im Feder-Dämpfer-System gespeichert ist, stellt Gl. (2.35) dar.

$$V_{e,j} = \frac{1}{2} c_{e,j} \varphi_{e,j}^2 \tag{2.35}$$

Die Lagrange-Funktion L ergibt sich dann nach Gl. (2.36) für die $i = 1, 2, 3, \ldots, n$ Körper und $j = 1, 2, 3, \ldots, m$ Gelenken mit Feder-Dämpfer Systemen.

$$L = \sum_{i=1}^{n} T_{kin,i}(\boldsymbol{\varphi}, \dot{\boldsymbol{\varphi}}, t) - \sum_{i=1}^{n} V_{pot,i}(\boldsymbol{\varphi}) - \sum_{j=1}^{m} V_{e,j}(\boldsymbol{\varphi}) \tag{2.36}$$

Auf diesen Term können nun die Lagrange-Gleichungen zweiter Art, für die x_F Freiheitsgrade q_ξ, angewendet werden (Gl. (2.37)).

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}_\xi} \right) - \frac{\partial L}{\partial q_\xi} = f_{diss,\xi} + f_{ext,\xi}, \quad \xi = 1, ..., x_F \tag{2.37}$$

In der Gleichung für (2.37) stellt $f_{diss,\xi}$ die dissipative verallgemeinerte Kraft und $f_{ext,xi}$ die externe verallg. Kraft dar. Für angetriebene Gelenke gilt $f_{diss,\xi} = 0$ und $f_{ext,xi} =$

$\tau_{m,xi}$. Für die Freiheitsgrade, die durch Feder-Dämpfer Elemente gestützt werden, gilt $f_{diss,\xi} = -d_\xi \dot{\varphi}_\xi$ und $f_{ext,\xi} = 0$.

Die Bewegungsgleichungen des Roboters – unter Vernachlässigung der gyroskopische Kopplungen der Motorträgheiten auf die Struktur des Roboters und Approximation der Strukturstеifigkeit über zusätzlich Feder-Dämpfer-Elemente – können in der Form von Gl. (2.38) dargestellt werden (mit n_a Gelenken und n_e strukturelastischen Freiheitsgraden, sowie $n_m \equiv n_a$ Antriebsstrang-Freiheitsgraden).

$$\begin{pmatrix} \boldsymbol{M}_{aa} & \boldsymbol{M}_{ae} & \mathbf{0} \\ \boldsymbol{M}_{ea} & \boldsymbol{M}_{ee} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Theta} \end{pmatrix} \begin{pmatrix} \ddot{\boldsymbol{\varphi}}_a \\ \ddot{\boldsymbol{\varphi}}_e \\ \ddot{\boldsymbol{\varphi}}_m \end{pmatrix} + \begin{pmatrix} \boldsymbol{h}_a \\ \boldsymbol{h}_e \\ \boldsymbol{h}_m \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \boldsymbol{\tau}_m \end{pmatrix} \tag{2.38a}$$

$$\boldsymbol{M}_{aa} = \boldsymbol{M}_{aa}(\boldsymbol{\varphi}_a, \boldsymbol{\varphi}_e) \tag{2.38b}$$

$$\boldsymbol{M}_{ae} = \boldsymbol{M}_{ea}^T = \boldsymbol{M}_{ae}(\boldsymbol{\varphi}_a, \boldsymbol{\varphi}_e) \tag{2.38c}$$

$$\boldsymbol{M}_{ee} = \boldsymbol{M}_{ee}(\boldsymbol{\varphi}_a, \boldsymbol{\varphi}_e) \tag{2.38d}$$

$$\boldsymbol{h}_a = \boldsymbol{h}_a(\boldsymbol{\varphi}_a, \dot{\boldsymbol{\varphi}}_a, \boldsymbol{\varphi}_e, \dot{\boldsymbol{\varphi}}_e, \boldsymbol{\varphi}_m, \dot{\boldsymbol{\varphi}}_m) \tag{2.38e}$$

$$\boldsymbol{h}_e = \boldsymbol{h}_e(\boldsymbol{\varphi}_a, \dot{\boldsymbol{\varphi}}_a, \boldsymbol{\varphi}_e, \dot{\boldsymbol{\varphi}}_e) \tag{2.38f}$$

$$\boldsymbol{h}_m = \boldsymbol{h}_m(\boldsymbol{\varphi}_a, \dot{\boldsymbol{\varphi}}_a, \boldsymbol{\varphi}_m, \dot{\boldsymbol{\varphi}}_m) \tag{2.38g}$$

$$\boldsymbol{\Theta} = \mathrm{diag}(\Theta_i) \text{ mit } i = 1, \ldots, n_m \tag{2.38h}$$

$$\boldsymbol{\varphi}_a = \begin{pmatrix} \varphi_{a,1} & \cdots & \varphi_{a,n_a} \end{pmatrix}^T \tag{2.38i}$$

$$\boldsymbol{\varphi}_e = \begin{pmatrix} \varphi_{e,1} & \cdots & \varphi_{e,n_e} \end{pmatrix}^T \tag{2.38j}$$

$$\boldsymbol{\varphi}_m = \begin{pmatrix} \varphi_{m,1} & \cdots & \varphi_{m,n_m} \end{pmatrix}^T \tag{2.38k}$$

$$\boldsymbol{\tau}_m = \begin{pmatrix} \tau_{m,1} & \cdots & \tau_{m,n_m} \end{pmatrix}^T \tag{2.38l}$$

Die Bezeichnungen in Gl. (2.38) sind dabei wie folgt gewählt:

- $\boldsymbol{M}(\boldsymbol{\varphi}_a, \boldsymbol{\varphi}_e)$: Massenmatrix des Roboters zusammen mit den Trägheiten $\boldsymbol{\Theta}$ der Motoren.
- $\boldsymbol{h}(\boldsymbol{\varphi}_a, \dot{\boldsymbol{\varphi}}_a, \boldsymbol{\varphi}_e, \dot{\boldsymbol{\varphi}}_e, \boldsymbol{\varphi}_m, \dot{\boldsymbol{\varphi}}_m)$: Von $\ddot{\boldsymbol{\varphi}} = \begin{pmatrix} \ddot{\boldsymbol{\varphi}}_a & \ddot{\boldsymbol{\varphi}}_e & \ddot{\boldsymbol{\varphi}}_m \end{pmatrix}^T$ unabhängige Terme der Bewegungsgleichungen und Antriebsstrangdynamik.
- $\boldsymbol{\varphi}_a$: Abtriebsseitige Winkel der Antriebsstränge.
- $\boldsymbol{\varphi}_e$: Winkel der strukturelastischen Freiheitsgraden innerhalb der Roboterkinematik.
- $\boldsymbol{\varphi}_m$: Motorwinkel der Antriebsstränge.
- $\boldsymbol{\tau}_m$: Motormomente.

Für Modelle, die in *Modelica/Dymola* erstellt werden, können die Bewegungsgleichungen durch *Dymola* mit Hilfe der *Modelica Multibody Library* (siehe [39]) direkt automatisch erstellt werden. In *Modelica/Dymola* werden für alle Kopplungspunkte der einzelnen Elemente und für alle Massen Bilanzgleichungen nach dem Prinzip von Newton-Euler (in Kombination mit dem Prinzip von d'Alembert) erstellt. Hierfür werden

auch alle benötigten Geschwindigkeiten automatisch aus den kinematischen Zusammenhängen ermittelt. In *Modelica/Dymola* wird zudem zwischen eindimensionalen, rotationssymmetrischen mechanischen Elementen und 3D-Elementen unterschieden, was eine effiziente Berechnung ermöglicht. Durch Kopplungselemente, können die Gleichungen aus 1D und 3D-Elementen kombiniert werden. Hierdurch können die Bewegungsgleichungen sehr effizient aufgestellt werden.

Die Bewegungsgleichungen liegen dann in Form eines DAE-Systems (Gl. (2.39)) vor, welches durch die in *Dymola* integrierten symbolischen Vorverarbeitungsalgorithmen und numerischen Integrationsverfahren gelöst werden kann (siehe hierzu auch Abschnitt 3.4).

$$\boldsymbol{F}\left(\dot{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{z}, \boldsymbol{u}, t\right) = 0 \tag{2.39a}$$

$$\boldsymbol{F}\left(\dot{\boldsymbol{x}}(t_0), \boldsymbol{x}(t_0), \boldsymbol{z}(t_0), \boldsymbol{u}_0, t_0\right) = 0 \tag{2.39b}$$

In Gl. (2.39) ist $\boldsymbol{x}$ ein Vektor der abhängigen, differenzierten Variablen, $\boldsymbol{z}$ der Vektor der abhängigen, algebraischen Variablen, $\boldsymbol{u}$ der Eingangsvektor und die Zeit t die unabhängige Variable und $\boldsymbol{F}$ ein Vektor von Funktionen dieser Variablen. Zusätzlich sind zur Lösung Startbedingungen zum Startzeitpunkt t_0 vorzugeben, welche die Gleichungen erfüllen. Für eine korrekte Auswertung der Gleichung ist es hierbei entscheidend, die richtigen Zustände (und entsprechenden Initialisierungen) zu wählen, damit man numerisch stabile und sinnvolle Lösungen erhält. Für mechanische Systeme sind dies i. Allg. die Geschwindigkeiten und Winkel der Gelenke. Hierbei ist es sinnvoll – soweit möglich – immer Relativwinkel (und Geschwindigkeiten) statt Absolutwinkel als Zustände des DAE-Systems zu bevorzugen. Dies bietet numerische Vorteile, da der Zustand sich dann innerhalb eines kleineren Bereichs bewegt und durch die relative Toleranz der numerischen Integration genauer bestimmt werden kann.

2.7 Gesamtmodell Varianten

Aus den vorgestellten Komponenten der Modell-Bibliothek können nun, je nach Anwendungsfall und Robotertyp (KR16, KR210), unterschiedliche Gesamtmodelle zusammengestellt werden. Bei der Zusammenstellung ist entscheidend, wie viel Rechenzeit verfügbar ist und welche physikalischen Effekte entscheiden sind. So ist die Schwinge des KR16 deutlich steifer aufgebaut, als die des KR210. Dadurch genügen hier einfachere Modelle, um die Verformungen zu approximieren. Ein anderer wesentlicher Unterschied zwischen dem KR210 und KR16 ist, dass nur der KR210 über einen Gewichtsausgleich verfügt und dieser für diesen Roboter immer mit modelliert werden muss. Modelle, die in Echtzeit invertiert werden sollen, dürfen eine gewisse Komplexität nicht überschreiten, da sonst die Rechenzeit und der Speicherbedarf der Modelle die verfügbare Rechenleistung des Steuerrechners überschreitet. Für Referenzmodelle, die nur Offline simuliert werden, können allerdings deutlich aufwändigere Modelle verwendet werden, wie z. B. Balken- und FEM-Modelle mit mehreren Moden pro Freiheitsgrad. Diese Referenzmodelle können zum Verifizieren der approximativen Modelle verwendet werden oder als Ausgangspunkt zur Analyse der Dynamik des Systems benutzt werden.

Welche Zusammenstellung im Einzelnen verwendet wird hängt also stark von der jeweiligen Aufgabenstellung ab. Durch den objektorientierten Aufbau der Modelle in *Mo-*

delica/Dymola ist ein Austauschen unterschiedlicher Modellkomponenten problemlos möglich. Dadurch ist ein sehr flexibler Umgang mit den Modellkomponenten möglich.

Die einzelnen *Modelica*-Komponenten sind teil einer umfangreichen Objektbibliothek. In dieser sind die Komponenten thematisch sortiert. Häufig wiederkehrende Systemkomponenten wie Antriebsstränge, sowie Roboterkinematiken – in unterschiedlichen Detaillierungsgraden – sind dort ebenfalls als Teilmodelle zusammengestellt, um schnell zu neuen Gesamtmodellen zu gelangen.

Für die Gesamtmodelle werden die Parameter der Modelle in Daten-Objekten[4] gespeichert, wodurch diese bequem mit wenig Aufwand geändert werden können, um etwa die Parameter eines KR16 mit denen eines KR210 zu vertauschen oder z. B. Steifigkeits-Parameter des Systems zu ändern.

[4]In *Modelica* werden diese Daten-Objekte mit *record* bezeichnet. Sie eigenen sich zum Speichern von Parametern und erlauben einen flexiblen Umgang mit unterschiedlichen Parametersätzen, ohne tief in die Modellstruktur eingreifen zu müssen.

Kapitel 3

Invertierung von strukturelastischen Robotern

Das direkte Invertieren, der vorgestellten elastischen Modelle, stellt die eleganteste Möglichkeit dar, um eine Vorsteuerung für den Roboter zu entwerfen.

Die aufgestellten Modelle sind differential algebraische Gleichungssysteme (DAE-Systeme). Die Invertierung dieser nichtlinearen Systeme ist nicht immer möglich: Von den Systemen müssen bestimmte Eigenschaften erfüllt sein, so dass sie invertiert werden können.

Im Folgenden sollen einige theoretische Vorüberlegungen zur Invertierung von Systemen im Allgemeinen durchgeführt werden.

Anschließend wird gezeigt wie die vorgestellten Modelle invertiert werden können, um zu einer Vorsteuerung zu gelangen.

Mit den erstellten Vorsteuerungen können – je nach Approximationsverfahren sogar in Echtzeit – Solltrajektorien für die antriebsseitigen Größen des Roboters berechnet werden, welche die Schwingungen des Roboters minimieren.

3.1 Invertierung linearer Systeme

Um die erstellten Modelle des strukturelastischen Roboters im Rahmen einer modellbasierten Vorsteuerung verwenden zu können müssen sie invertiert werden. Die Berechnung der Eingangsgrößen aus den Ausgangsgrößen ist nicht für jedes System möglich (siehe [40]).

Damit ein lineares System stabil invertierbar ist dürfen Nullstellen N_i des Systems $G(s)$ keinen positiven Realteil besitzen. Zusätzlich muss das System einen Differenzengrad $d_G \geq 0$ (siehe Gl. (3.2)) besitzen damit das resultierende inverse System stabil und realisierbar ist.

$$G(s) = k \cdot \frac{b_0 + b_1 \cdot s + ... + b_{m_G-1} \cdot s^{m_G-1} + s^{m_G}}{a_0 + a_1 \cdot s + ... + a_{n_G-1} \cdot s^{n_G-1} + s^{n_G}} = k \cdot \frac{\prod_{i=1}^{m_G}(s - N_i)}{\prod_{j=1}^{n_G}(s - P_j)} \tag{3.1}$$

Bei einer Invertierung werden im linearen Fall, Zähler und Nenner vertauscht. Nullstel-

len N_i in der rechten komplexen Halbebene werden dadurch zu Polen P_j mit positivem Realteil, wodurch das inverse System instabil wird. Zwischen Zähler- und Nennergrad besteht der Differenzengrad d_G.

$$d_G = n_G - m_G \tag{3.2}$$

Erfüllt das System nicht die Bedingungen an den Differenzengrad, so können Filter verwendet werden, um das System approximativ zu invertieren.

Approximativ können lineare Systeme auch über eine (hohe) Rückführungsverstärkung invertiert werden (siehe [41]). Hierzu wird der Ausgang des Systems über eine proportionale Verstärkung k_{inv} wieder auf den Eingang des Systems zurückgeführt (Gl. (3.3)).

$$G(s)^{-1} \approx \frac{k_{inv}}{1 + k_{inv}G(s)} = \frac{k_{inv}P(s)}{P(s) + k_{inv}N(s)} \tag{3.3a}$$

$$\lim_{k_{inv}\to\infty} = \frac{P(s)}{N(s)} \tag{3.3b}$$

Dieses Vorgehen führt jedoch auf numerisch schlecht konditionierte inverse Systeme.

Ähnliche Zusammenhänge können auch für nichtlineare Systeme aufgestellt werden. Allerdings ist hier der theoretische Aufwand deutlich größer. In Abschnitt 3.2 soll daher die benötigte Theorie zur Bestimmung der Nulldynamik kurz vorgestellt werden und diese Theorie dann auch auf das aufgestellte Robotermodell angewendet werden.

3.2 Nulldynamik nichtlinearer Systeme

Es finden sich in der Literatur unterschiedliche Ansätze (z. B.[42, 43]) um die Nulldynamik von nichtlinearen Systemen berechnen zu können. Die folgenden Ableitungen orientieren sich an denen von Isidori [42].

Grundlage für die Untersuchungen ist zunächst das SISO[1]-System in der Form von Gleichung (3.4).

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})u \tag{3.4a}$$

$$y = h(\mathbf{x}) \tag{3.4b}$$

Der relative Grad r dieses Systems ist definiert nach Gleichung (3.5) und entspricht dem Differenzengrad d_G im linearen Fall. Er ist allerdings nur lokal um den Punkt $\mathbf{x}^0$ definiert und für bestimmte Systemen kann der relative Grad auch nicht für beliebige Zustände bestimmt sein. Der Operator L steht hierbei für die Lie-Ableitung (siehe Anhang A.3).

$$L_{\mathbf{g}}L_{\mathbf{f}}^{k}h(\mathbf{x}) = 0, \quad k < r-1 \tag{3.5a}$$

$$L_{\mathbf{g}}L_{\mathbf{f}}^{r-1}h(\mathbf{x}^0) \neq 0 \tag{3.5b}$$

Nur wenn der relative Grad r kleiner als die Ordnung n des Systems ist, existiert eine Nulldynamik.

[1]Englisch: SISO - Single Input, Single Output

Eine alternative Interpretation von Gl. (3.5) ist es, dass r der Anzahl an Differentiationen von $y(t)$ zum Zeitpunkt $t = t_0$ entspricht, welche benötigt werden, um $u(t_0)$ explizit erscheinen zu lassen. Die ersten k Ableitungen, welche dies zeigen, können in der Form von Gl. (3.6) geschrieben werden:

$$
\begin{aligned}
y(t_0) &= h(\boldsymbol{x}(t_0)) = h(\boldsymbol{x}_0) && \text{(3.6a)}\\
\dot{y} &= \frac{\partial h}{\partial \boldsymbol{x}}\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \frac{\partial h}{\partial \boldsymbol{x}}\left(\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})u\right) && \text{(3.6b)}\\
&= L_f h(\boldsymbol{x}) + L_g h(\boldsymbol{x})u && \text{(3.6c)}\\
\text{Falls gilt: } r &> 1 \Rightarrow L_g h(\boldsymbol{x}) \equiv 0 \Rightarrow \dot{y} = L_f h(\boldsymbol{x}) && \text{(3.6d)}\\
\ddot{y} &= \frac{\partial L_f h}{\partial \boldsymbol{x}}\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} && \text{(3.6e)}\\
&= L_f^2 h(\boldsymbol{x}) + L_g L_f h(\boldsymbol{x})u && \text{(3.6f)}\\
\text{Falls gilt: } r &> 2 \Rightarrow L_g L_f h(\boldsymbol{x}) \equiv 0 \Rightarrow \ddot{y} = L_f^2 h(\boldsymbol{x}) && \text{(3.6g)}\\
&\vdots &&\\
y^{(k)} &= L_f^k h(\boldsymbol{x}), \quad k < r \text{ und } t \text{ um } t_0 && \text{(3.6h)}\\
y^{(r)} &= L_f^r h(\boldsymbol{x}_0) + L_g L_f^{r-1} h(\boldsymbol{x}_0)u(t_0) && \text{(3.6i)}
\end{aligned}
$$

Ein Vorgehen nach Isidori zum Bestimmen der Nulldynamik ist, das System auf Isidori-Normalform zu bringen, in welcher die Nulldynamik einfach abgelesen werden kann.

Die Normalform stellt eine lokale Koordinatentransformation dar. Der Zustandsvektor des transformierten Systems wird mit $\mathbf{z}$ bezeichnet. Für die Koordinatentransformation wird eine Transformationsmatrix $\boldsymbol{\Phi}(\mathbf{x})$ nach Gleichung (3.7) benötigt, wobei vorausgesetzt wird, dass eine Nulldynamik existiert und r kleiner als n ist. Alle Bedingungen gelten daher wieder nur lokal um $\mathbf{x}^0$.

$$
\boldsymbol{\Phi}(\mathbf{x}) = \begin{pmatrix} \phi_1(\mathbf{x}) \\ \vdots \\ \phi_n(\mathbf{x}) \end{pmatrix} \tag{3.7}
$$

Die ersten r Elemente der Transformationsmatrix ergeben sich nach Gleichung (3.8) durch Lie-Ableitungen der Ausgangsfunktion $h(\mathbf{x})$ bezüglich $\mathbf{f}(\mathbf{x})$.

$$
\begin{aligned}
\phi_1(\mathbf{x}) &= h(\mathbf{x}) && \text{(3.8a)}\\
\phi_2(\mathbf{x}) &= L_{\mathbf{f}} h(\mathbf{x}) && \text{(3.8b)}\\
&\vdots && \text{(3.8c)}\\
\phi_r(\mathbf{x}) &= L_{\mathbf{f}}^{r-1} h(\mathbf{x}) && \text{(3.8d)}
\end{aligned}
$$

Um zur Isidori-Normalform zu gelangen müssen ϕ_{r+1} bis ϕ_n nach Gleichung (3.9) gewählt werden, wobei die Jacobimatrix $\boldsymbol{J}_{\boldsymbol{\Phi}}$ nicht singulär werden darf (Gl. (3.10)). Dies hat zur Folge, dass partielle Differentialgleichungen (PDE[2]) gelöst werden müssen.

$$
L_{\mathbf{g}}\phi_i(\mathbf{x}) = 0 \quad \forall\, \mathbf{x} \text{ um } \mathbf{x}^0 \quad \text{mit} \quad r+1 < i < n \tag{3.9}
$$

$$
det\left(\boldsymbol{J}_{\boldsymbol{\Phi}}\right) = \left|\frac{\partial \boldsymbol{\Phi}(\mathbf{x})}{\partial \mathbf{x}}\right| \neq 0 \tag{3.10}
$$

[2]Englische Abkürzung für: Partial differential equation.

In den neuen Koordinaten $\boldsymbol{z}$ ergibt sich nach [42] Gl. (3.11).

$$\dot{z}_1 = \frac{\delta \phi_1}{\delta \boldsymbol{x}} \frac{dx}{dt} = \phi_2(x) = z_2 \tag{3.11a}$$

$$\vdots \tag{3.11b}$$

$$\dot{z}_{r-1} = \frac{\delta \phi_{r-1}}{\delta \boldsymbol{x}} \frac{dx}{dt} = \phi_r(x) = z_r \tag{3.11c}$$

$$\dot{z}_r = L_{\boldsymbol{f}}^r h(\boldsymbol{x}) + L_{\boldsymbol{g}} L_{\boldsymbol{f}}^{r-1} h(\boldsymbol{x}) u \tag{3.11d}$$

Als nächster Schritt muss in Gl. (3.11) $\boldsymbol{x}$ durch eine Funktion von $\boldsymbol{z}$ ersetzt werden. Die Lösung dieses Gleichungssystems wird mit $\boldsymbol{\Phi}^{-1}(\mathbf{z})$ bezeichnet.

Das transformierte System ergibt sich dann, wie in Gleichung (3.12) dargestellt, in den neuen Zustandskoordinaten $\mathbf{z}$, wobei der Ausgang des neuen Systems nun $y = z_1$ ist.

$$\dot{z}_1 = z_2 \tag{3.12a}$$

$$\dot{z}_2 = z_3 \tag{3.12b}$$

$$\vdots \tag{3.12c}$$

$$\dot{z}_{r-1} = z_r \tag{3.12d}$$

$$\dot{z}_r = b(\mathbf{z}) + a(\mathbf{z}) u \tag{3.12e}$$

$$\dot{z}_{r+1} = q_{r+1}(\mathbf{z}) \tag{3.12f}$$

$$\vdots \tag{3.12g}$$

$$\dot{z}_n = q_n(\mathbf{z}) \tag{3.12h}$$

$$a(\mathbf{z}) = L_{\mathbf{g}} L_{\mathbf{f}}^{r-1} h(\boldsymbol{\Phi}^{-1}(\mathbf{z})) \tag{3.12i}$$

$$b(\mathbf{z}) = L_{\mathbf{f}}^r h(\boldsymbol{\Phi}^{-1}(\mathbf{z})) \tag{3.12j}$$

$$q_i(\mathbf{z}) = L_{\mathbf{f}} \phi_i(\boldsymbol{\Phi}^{-1}(\mathbf{z})) + L_{\mathbf{g}} \phi_i(\boldsymbol{\Phi}^{-1}(\mathbf{z})) u(t) \quad \text{mit } r+1 \leq i \leq n \tag{3.12k}$$

Wurden ϕ_{r+1} bis ϕ_n nach Gleichung (3.9) bestimmt, dann fällt der Term mit $u(t)$ für alle $q_i(\mathbf{z})$ mit $r+1 \leq i \leq n$ weg. Nur dann spricht man von der (Isidori-) Normalform.

Mit Hilfe der, auf Normalform transformierten, Zustandsgleichungen kann nun die Nulldynamik bestimmt werden. Dies bedeutet, es wird eine Eingangsfunktion $u^0(\cdot)$ zu einem Anfangszustand $\mathbf{x}^0$ gesucht, so dass der daraus resultierende Ausgang des Systems $y(t)$ identisch Null für alle t in der Nähe von $t = 0$ wird. Dabei werden alle Lösungen gesucht und nicht nur die triviale Lösung $\mathbf{x}^0 = \mathbf{0}, u^0 = 0$.

In der Normalform ist der Ausgang des Systems $y(t) = z_1(t)$. Anhand der Bedingung, dass $y(t) \equiv 0$ gelten soll, sieht man dass Gl. (3.13) erfüllt werden muss.

$$z_1(t) = z_2(t) = \cdots = z_r(t) = 0 \tag{3.13}$$

Dies bedeutet, dass sich die Nulldynamik durch Nullsetzen von $z_1(t)$ bis $z_r(t)$, des auf Normalform transformierten Systems, ergibt. Ist dieses gewonnene System stabil, so ist die Nulldynamik stabil und das System stabil invertierbar. Ist sie nicht stabil, so ist keine exakte Inversion möglich.

Die Gleichungen, die hier für SISO-Systeme gezeigt wurden, können auch auf den MIMO[3]-Fall erweitert werden, mit $m = n$ Ein- und Ausgängen. Das System lässt sich dann weiterhin in Form von Gleichung (3.4) darstellen, wobei jetzt allerdings Vektoren durch Matrizen ersetzt werden müssen (Gleichung (3.14)).

$$\begin{aligned} \mathbf{u} &= (u_1, \ldots, u_m)^T && \text{(3.14a)} \\ \mathbf{y} &= (y_1, \ldots, y_m)^T && \text{(3.14b)} \\ \mathbf{g}(\mathbf{x}) &= (\mathbf{g}_1(\mathbf{x}) \ \ldots \ \mathbf{g}_m(\mathbf{x})) && \text{(3.14c)} \\ \mathbf{h}(\mathbf{x}) &= (h_1(\mathbf{x}), \ldots, h_m(\mathbf{x}))^T && \text{(3.14d)} \end{aligned}$$

Der relative Grad ist dann als Vektor $\boldsymbol{r} = (r_1, \ldots, r_m)$ an einem Punkt $\mathbf{x}^0$ definiert, wenn Gleichung (3.15) für alle $\mathbf{x}$ in der Umgebung von $\mathbf{x}^0$ gilt und Gleichung (3.16) bei $\mathbf{x}^0$ erfüllt ist, daher die Matrix $\mathbf{A}(\mathbf{x}^0)$ nicht singulär ist.

$$L_{\mathbf{g}j} L_{\mathbf{f}}^k h_i(\mathbf{x}) = 0, \quad k < r_i - 1, 1 \leq j \leq m, 1 \leq i \leq m \tag{3.15}$$

$$det(\mathbf{A}(\mathbf{x}^0)) = \left| \begin{pmatrix} L_{\mathbf{g}1} L_{\mathbf{f}}^{r_1-1} h_1(\mathbf{x}^0) & \ldots & L_{\mathbf{g}m} L_{\mathbf{f}}^{r_1-1} h_1(\mathbf{x}^0) \\ L_{\mathbf{g}1} L_{\mathbf{f}}^{r_2-1} h_2(\mathbf{x}^0) & \ldots & L_{\mathbf{g}m} L_{\mathbf{f}}^{r_2-1} h_2(\mathbf{x}^0) \\ L_{\mathbf{g}1} L_{\mathbf{f}}^{r_m-1} h_m(\mathbf{x}^0) & \ldots & L_{\mathbf{g}m} L_{\mathbf{f}}^{r_m-1} h_m(\mathbf{x}^0) \end{pmatrix} \right| \neq 0 \tag{3.16}$$

Analog zum SISO-Fall können dann Gleichungen für $\boldsymbol{\Phi}(\mathbf{x})$ aufgestellt werden (Gl. (3.17)).

$$\begin{aligned} \phi_1^i(\mathbf{x}) &= h_i(\mathbf{x}), \quad r_1 + \cdots + r_m \leq n, 1 \leq i \leq m && \text{(3.17a)} \\ \phi_2^i(\mathbf{x}) &= L_{\mathbf{f}} h_i(\mathbf{x}), \quad r_1 + \cdots + r_m \leq n, 1 \leq i \leq m && \text{(3.17b)} \\ &\ldots && \text{(3.17c)} \\ \phi_{ri}^i(\mathbf{x}) &= L_{\mathbf{f}}^{r_i-1} h_i(\mathbf{x}), \quad r_1 + \cdots + r_m \leq n, 1 \leq i \leq m && \text{(3.17d)} \end{aligned}$$

Die Gleichungen für $\phi_{r+1}(\mathbf{x}), \ldots, \phi_n$ werden aus den partiellen Differentialgleichungen nach Gl. (3.18) bestimmt, wobei die Jacobimatrix von $\boldsymbol{\Phi}(\mathbf{x})$ an der Stelle $\mathbf{x}^0$ nicht singulär sein darf (Gl. (3.10)).

$$L_{gj} \phi_i(\mathbf{x}) = 0 \quad \forall \ \mathbf{x} \text{ um } \mathbf{x}^0 \ , \text{ mit } \quad r + 1 < i < n, 1 \leq j \leq m \tag{3.18}$$

Mit diesen Gleichungen lässt sich das MIMO System dann wie im SISO-Fall auf eine Normalform mit dem neuen Zustandsvektor $\mathbf{z}$ transformieren, wobei die Ausgänge dann durch $y_i = \xi_1^i = z_1^i$ gegeben sind. Die Nulldynamik ergibt sich durch Nullsetzen der Ausgänge y_i was, wie im SISO-Fall, bedeutet, dass die neuen Zustände bis zum jeweiligen relativen Grad zu Null gesetzt werden (Gl. (3.19)).

$$z_1^i(t) = z_2^i(t) = \cdots = z_{ri}^i(t) = 0, \quad 1 \leq i \leq m \tag{3.19}$$

Für weitere Details und Beweise sei auf [42] und [44] verwiesen.

[3]Englisch: MIMO - Multiple Input, Multiple Output.

3.3 Anwendung der Theorie auf ein strukturelastisches Robotermodell

Die in Abschnitt 3.2 vorgestellte Theorie soll nun auf das erstellte strukturelastische Robotermodell angewendet werden.

Problematisch dabei ist, dass die Gleichungen für ein komplexeres System sehr aufwändig werden und symbolisch, selbst mit Computeralgebrasystemen, kaum zu lösen sind. Kritisch sind hierbei vor allem Gleichung (3.9) bzw. (3.18) im MIMO-Fall. Diese Gleichungen führen zu komplizierten partiellen Differentialgleichungen. Auch die Lösung von $\mathbf{\Phi}^{-1}(\mathbf{z})$ ist für stark nichtlineare Gleichungen mit vielen verschachtelten Sinus- und Kosinus-Termen (in höheren Potenzen) oft nicht exakt möglich.

Um dennoch den Einfluss der Strukturelastizitäten auf die Nulldynamik des Robotermodells – mit Hilfe der Theorie von Isidori – zu untersuchen, muss ein stark vereinfachtes Modell gewählt werden, für das noch algebraische Lösungen mittels heutiger Computeralgebrasysteme[4] gefunden werden können. Das untersuchte Modell ist ein

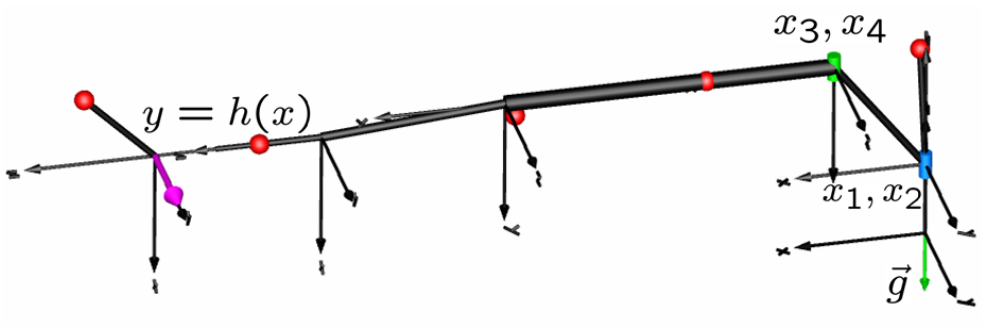


Abbildung 3.1: Schematische Darstellung des stark vereinfachten Modells des KR210 mit vier Zuständen. Ausgang y ist die kartesische Verschiebung in y-Richtung (lila Pfeil) am Handflansch, Eingang u ist das Motormoment $\tau_{m,1}$ bei Achse 1 (blaues Gelenk). Zusätzlich gibt es eine Kippsteifigkeit $(c_{e,1}, d_{e,1})$ bei Achse 2 (grünes Gelenk).

stark vereinfachtes Modell des KR210 in Strecklage (siehe Abb. 3.1). Als Eingang u für das System wird das Motormoment $\tau_{m,1}$ an Achse 1 gewählt, wobei die Dynamik des Antriebsstrangs vernachlässigt wird. Ausgangsgröße $y = h(\mathbf{x})$ ist die kartesische Abweichung in y-Richtung des Handflansches des Roboters. Als Strukturelastizität wird lediglich eine Kippsteifigkeit der Schwinge, modelliert als Feder-Dämpfersystem $(c_{e,1}, d_{e,1})$ bei Achse 2 gewählt. Zusätzlich trägt der Roboter eine Last von 210 kg, deren Schwerpunkt oberhalb des Handflansches liegt. Die $i = 1 \ldots 5$ einzelnen Segmente sind als Starrkörper modelliert, bestehend aus Masse m_i, körperfesten Trägheitstensor bezüglich des Schwerpunktes $\mathbf{\Theta}_i^{(S)}$ und Schwerpunktsvektor $\mathbf{r}_{S,i}$.

Zum Aufstellen der Gleichungen nach Lagrange, werden zunächst die kinematischen Zusammenhänge, in der Form von homogenen 4x4-Matrizen $\mathbf{T}$, aufgestellt (siehe Anhang A.8.3). Da das System als SISO-System betrachtet werden soll, sind alle Transformationsmatrizen $\mathbf{T}$ konstant bis auf ${}^1_0\mathbf{T}_R = {}^1_0\mathbf{T}_R(\varphi_{a,1})$ und $\mathbf{T}_K = \mathbf{T}_K(\varphi_{e,1})$.

Sind diese Matrizen aufgestellt, können die potentielle und kinetische Energie des Sy-

[4] Verwendet wurde *MAPLE* 11.0 der Firma Maplesoft.

stems bestimmt werden. Die kinetische Energie, für jeden einzelnen Starrkörper, ergibt sich dabei nach Gleichung (2.33), da als Körper-Bezugspunkt Q der Schwerpunkt S gewählt wird. Die potentielle Energie ergibt sich über Gl. (2.34) für jeden einzelnen Starrkörper. Die Energie, die im Feder-Dämpfer-System gespeichert ist, stellt Gl. (3.20) dar.

$$V_{e,1} = \frac{1}{2} c_{e,1} \varphi_{e,1}^2 \tag{3.20}$$

Um Gleichung (2.33) verwenden zu können, müssen die Winkelgeschwindigkeiten $\boldsymbol{\omega}_i$ der einzelnen Körper bezüglich der Schwerpunkte S_i berechnet werden. Hierbei wird ausgenutzt, dass Winkelgeschwindigkeiten, welche bezüglich dem gleichen Koordinatensystem aufgestellt sind, vektoriell addiert werden dürfen. Hierzu werden die einzelnen Winkelgeschwindigkeiten zuerst in das Inertial-Koordinatensystem '0' transformiert, addiert und dann bezüglich des jeweiligen Schwerpunktes transformiert (Gl. (3.21)). Das Vorzeichen für die Geschwindigkeit $\dot{\varphi}_{a,1}$, richtet sich dabei nach der Konvention in dieser Arbeit (Details im Anhang unter Abschnitt A.6).

$$\begin{aligned}
\boldsymbol{\omega}_{a,1}^{(0)} &= {}_0^1\mathbf{R} \begin{pmatrix} 0 \\ 0 \\ -\dot{\varphi}_{a,1} \end{pmatrix} && \text{(3.21a)} \\
\boldsymbol{\omega}_{m1}^{(S)} &= {}_0^{1C}\mathbf{R}^T \boldsymbol{\omega}_{a,1}^{(0)} && \text{(3.21b)} \\
\boldsymbol{\omega}_{e,1}^{(0)} &= {}_0^K\mathbf{R} \begin{pmatrix} 0 \\ 0 \\ \dot{\varphi}_{e,1} \end{pmatrix} && \text{(3.21c)} \\
\boldsymbol{\omega}_{m2}^{(S)} &= {}_0^{2C}\mathbf{R}^T (\boldsymbol{\omega}_{a,1}^{(0)} + \boldsymbol{\omega}_{e,1}^{(0)}) && \text{(3.21d)} \\
\boldsymbol{\omega}_{m3}^{(S)} &= {}_0^{3C}\mathbf{R}^T (\boldsymbol{\omega}_{a,1}^{(0)} + \boldsymbol{\omega}_{e,1}^{(0)}) && \text{(3.21e)} \\
\boldsymbol{\omega}_{mh}^{(S)} &= {}_0^{HC}\mathbf{R}^T (\boldsymbol{\omega}_{a,1}^{(0)} + \boldsymbol{\omega}_{e,1}^{(0)}) && \text{(3.21f)} \\
\boldsymbol{\omega}_{ml}^{(S)} &= {}_0^{LC}\mathbf{R}^T (\boldsymbol{\omega}_{a,1}^{(0)} + \boldsymbol{\omega}_{e,1}^{(0)}) && \text{(3.21g)}
\end{aligned}$$

Die Lagrange-Funktion L ergibt sich dann nach Gl. (2.36). Auf diesen Term können nun die Lagrange-Gleichungen zweiter Art, für die zwei Freiheitsgrade $(\varphi_{e,1}, \varphi_{a,1})$, angewendet werden (Gl. (3.22)).

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\varphi}_j}\right) - \frac{\partial L}{\partial \varphi_j} = f_{diss,j} + f_{ext,j}, \quad j = \{(a,1),(e,1)\} \tag{3.22}$$

In der Gleichung für $\varphi_{a,1}$ ist die dissipative verallgemeinerte Kraft $f_{diss,1} = 0$ und die externe verallg. Kraft $f_{ext,1} = \tau_{m,1}$. Für $\varphi_{e,1}$ ist $f_{diss,e} = -d_{e,1}\dot{\varphi}_{e,1}$ und $f_{ext,e} = 0$.

Um die Theorie von Isidori anwenden zu können, muss das System nun auf Zustandsform nach Gleichung (3.4) gebracht werden. Hierzu müssen vier Zustände gewählt werden. In diesem Fall ist es vorteilhaft die Zustände nach Gleichung (3.23) zu wählen.

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} \varphi_{a,1} \\ \dot{\varphi}_{a,1} \\ \varphi_{e,1} \\ \dot{\varphi}_{e,1} \end{pmatrix} \tag{3.23}$$

Für die Zustandsableitung $\dot{\mathbf{x}}$ müssen nun die gewonnen Gleichungen nach den jeweiligen Ableitungen der Variablen gelöst werden und ineinander eingesetzt werden, um auf die

Form von Gleichung (3.24) zu kommen.

$$\dot{\mathbf{x}} = \begin{pmatrix} x_2 \\ f_1(x_1, x_2, x_3, x_4, u) \\ x_4 \\ f_2(x_1, x_2, x_3, x_4, u) \end{pmatrix} \tag{3.24}$$

Aus Gleichung (3.24) muss nun noch der Eingang $u = \tau_{m,1}$ abgespalten werden, um wieder auf die Form von Gl. (3.25) zu gelangen.

$$\dot{\mathbf{x}} \;=\; \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})u \tag{3.25}$$

Der Ausgang $y = h(x_1, x_3)$ wird über die Transformationsmatrix ${}^6_0\mathbf{T}$ bestimmt, in welcher die Verschiebung in y-Richtung enthalten ist (y=${}^6_0\mathbf{T}_{(2,4)}$).

Auf das auf Zustandsform transformierte System können nun die Gleichungen nach Abschnitt 3.2 angewendet werden.

Für das System ergibt sich nach Gl. (3.5) ein relativer Grad von $r = 2$. Es existiert also eine Nulldynamik. Als Ausgangspunkt $\mathbf{x}^0$ für die Untersuchungen wird dabei die Strecklage verwendet, wobei die Anfangsgeschwindigkeit identisch Null ist.

Für die partielle Differentialgleichung (3.9) konnten dabei nur Lösungen in *MAPLE* erzielt werden, nachdem die gewonnen Gleichungen mit Hilfe einer Taylor-Entwicklung[5] zweiter Ordnung um $\mathbf{x}^0$ durchgeführt wurde. Die Gleichungen erstrecken sich über mehrere Seiten und besitzen komplizierte verschachtelte Ausdrücke.

Auch das Gleichungssystem $\mathbf{\Phi}^{-1}(\mathbf{z})$ lies sich erst nach der Taylor-Entwicklung auflösen. Um die Nulldynamik zu untersuchen müssen nun die ersten $r = 2$ Zustände $\mathbf{z}$ auf 0 gesetzt werden (Gl. (3.13)).

Das so gewonnene (Teil-) System beschreibt die Nulldynamik des Systems und ergibt sich (linearisiert) in der Form von Gl. (3.26), mit den Konstanten C_1, C_2, C_3, welche die geometrischen Terme und die Massen und Trägheiten enthalten.

$$\dot{z}_3 \;=\; C_1 \cdot z_4 \tag{3.26a}$$
$$\dot{z}_4 \;=\; C_2 \cdot d_{e,1} \cdot z_4 + C_3 \cdot c_{e,1} \cdot z_3 \tag{3.26b}$$

Werden die Parameter für den KR210 in Gleichung (3.26) eingesetzt, so ergibt sich Gleichung (3.27).

$$\dot{z}_3 \;=\; 55.588 \cdot z_4 \tag{3.27a}$$
$$\dot{z}_4 \;=\; 0.0244 \cdot d_{e,1} \cdot z_4 + 4.393e^{-4} \cdot c_{e,1} \cdot z_3 \tag{3.27b}$$

Wie man sieht ist dieses System, für physikalisch sinnvolle Werte für $c_{e,1}$ und $d_{e,1}$, instabil. Die Nulldynamik dieses System ist also instabil und es kann kein exaktes inverses Modell aufgestellt werden.

[5]Eine Linearisierung ist laut einer Anmerkung (Remark 4.3.2) von Isidori in [42] gültig, um die Nulldynamik zu untersuchen

Wie bereits bei diesem extrem vereinfachten System gesehen werden kann, wird durch die Einführung von zusätzlichen elastischen Freiheitsgraden in das Robotermodell (im Beispiel lediglich eine Kippsteifigkeit), eine instabile Nulldynamik erzeugt, wenn man als Eingang des Modells das Motormoment und als Ausgang die TCP-Position wählt. Genau diese Größen sind jedoch für eine Robotervorsteuerung sinnvoll, da hierbei Trajektorien für die motorseitigen Größen (also $\boldsymbol{\tau}_m, \boldsymbol{\varphi}_m$ und $\dot{\boldsymbol{\varphi}}_m$) gesucht werden, mit welchen der TCP, die Werkzeugspitze des Roboters, möglichst schwingungsfrei, auf einer gewünschte Trajektorie gesteuert werden kann.

Zudem zeigt sich, dass man bei der Anwendung der Theorie von Isidori, bei realistischen Systemen schnell an Grenzen stößt: Die partiellen Differential-Gleichungen (PDE) werden zu kompliziert, um sie selbst mit Computeralgebrasystemen lösen zu können. Für die Untersuchung ist man daher oft auf das Hilfsmittel der Linearisierung angewiesen oder muss andere Theorien oder strukturelle Untersuchungen der Gleichungssysteme zur Hilfe nehmen.

3.4 Lösung und Invertierung von DAE-Systemen mit Modelica/Dymola

In *Dymola*, einem Werkzeug zum Erstellen von Modellen mit der Modellierungssprache *Modelica*, sind Algorithmen implementiert die zum Erstellen von inversen Modellen verwendet werden können.

Das Vorgehen in *Dymola* wird im Folgenden kurz beschrieben. Zusätzliche Informationen finden sich in [45], [46] und [47].

Modelica-Modelle liegen in *Dymola* zunächst in DAE-Form (3.28) vor.

$$\boldsymbol{F}\left(\dot{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{z}, \boldsymbol{u}, t\right) = 0 \tag{3.28}$$

Hierbei sind $\mathbf{x}$ Variablen, welche differenziert im Modell vorkommen, $\mathbf{z}$ sind algebraische Größen und $\mathbf{u}$ sind bekannte Eingangsfunktionen der Zeit t. Den Parametern $\boldsymbol{p}$ des Systems können konkrete Werte vorgegeben werden, weshalb sie nicht explizit aufgeführt sind.

Die kleinste Anzahl an Differentiationen ν welche benötigt werden um Gleichung (3.28) in die Form (3.29) zu transformieren wird nach [48] als (differentieller) Index der DAE bezeichnet.

$$\boldsymbol{F}_\nu\left(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{u}, t\right) = \begin{pmatrix} \dot{\boldsymbol{x}} \\ \dot{\boldsymbol{y}} \end{pmatrix} \tag{3.29}$$

Die Jacobimatrix von $\boldsymbol{F}(.)$ nach Gl. (3.30) kann im allgemeinen Fall (stellenweise) singulär sein.

$$\left| \frac{\delta \boldsymbol{F}}{\delta \dot{\boldsymbol{x}}} \quad \vdots \quad \frac{\delta F}{\delta \boldsymbol{z}} \right| \tag{3.30}$$

Um Gl. (3.28) auf Zustandsform (mit algebraischen Ausgangsgleichungen nach [47]) zu transformieren, welche mit Standardverfahren gelöst werden können, sind verschiedene Zwischenschritte nötig.

Hierzu werden in *Dymola* zum einen neue Gleichungen und Zwangsbedingungen durch Differentiation von $\boldsymbol{F}(.)$ gewonnen. Welche Gleichungen aus $\boldsymbol{F}(.)$ hierzu differenziert werden müssen, kann mit dem graphentheoretischen Algorithmus von Pantelides [49] bestimmt werden, welcher ursprünglich zur konsistenten Initialisierungen von DAE Systemen konzipiert wurde. Hierin wird ein Graph aufgebaut und verarbeitet welcher Gleichungen des Systems und in den Gleichungen auftretende Variablen verknüpft.

Zudem werden über den Algorithmus von Mattsson und Söderlind (siehe [50]), welcher auf den Algorithmus von Pantelides aufbaut, die Zustände des Systems neu gewählt („dummy derivatives"). Über symbolische Vorverarbeitung und Umsortierung der Gleichungen über verschiedene Algorithmen (Transformation auf „Block-Lower-Triangular"-Form [45] und „Tearing"[51], Graph-Algorithmen [52]), lassen sich einzelne Gleichungsblöcke in $\boldsymbol{F}(.)$ vereinfachen, wodurch möglichst viele Zustandsableitungen und algebraische Variablen explizit berechenbar sind.

Mit Hilfe dieser Algorithmen kann das DAE-System mit höherem Index auf ein System mit Index 1 transformiert werden, so dass es in Zustandsform nach Gl. (3.31) mit algebraischen Ausgangsgleichungen für $\boldsymbol{y}$ vorliegt (mit durch die Algorithmen modifiziertem Zustand $\boldsymbol{x}_*$).

$$\dot{\boldsymbol{x}}_* = \boldsymbol{f}(\boldsymbol{x}_*, \boldsymbol{u}, t) \tag{3.31a}$$

$$\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x}_*, \boldsymbol{u}, t) \tag{3.31b}$$

Dieser Vorgang wird als Indexreduktion bezeichnet. Zusätzlich müssen auch die Anfangsbedingungen für das System transformiert werden, wobei diese nicht nur die Ausgangs-DAE (3.28) erfüllen müssen, sondern auch diejenigen differenzierten Gleichungen von (3.28), die benötigt werden um die DAE in eine äquivalente Zustandsform (3.31) zu transformieren (siehe [45]). Sind im Modelle diskrete Ereignisse (Englisch: Event) vorhanden (Schaltfunktionen) oder finden neue Initialisierungen von Gleichungen zur Simulationszeit im Modell statt, so muss die Transformation auf (3.31) erneut durchgeführt werden (Details hierzu in [47]). Hierdurch können diskrete und kontinuierliche Gleichungen in (3.28) verarbeitet werden (hybride Systeme).

Ein inverses Modell kann mit Hilfe dieser Algorithmen dadurch bestimmt werden, dass einige (oder alle) Elemente des Eingangsvektors $\mathbf{u}$ nun nicht mehr als bekannt, sondern als unbekannt angenommen werden und dafür Variablen aus $\mathbf{x}$ oder $\mathbf{z}$ als bekannt gesetzt werden. Das Resultat ist wieder ein DAE-System nach (3.28), welches analog zum Vorwärtsmodell gelöst werden kann.

Voraussetzung ist allerdings, dass alle Funktionen, die im Modell verwendet werden, eindeutig invertierbar sind (im eindimensionalem Fall, für jeden Abszissenwert nur ein Ordinatenwert) und die Nulldynamik des Systems stabil ist. Diese bedeutet, dass das durch die Vertauschung der Ein- und Ausgänge erstandene neue DAE-System stabil ist.

Zudem müssen Ableitungen der neu gewählten Eingänge bereitgestellt werden (dem Index ν entsprechend) oder es muss dem Modell über Filter ein Referenzmodell vorgeschaltet werden, über welches die benötigten Ableitungen (approximativ) berechnet werden können (siehe Abschnitt 3.5). Alternativ kann auch eine Integratorkette vor die neuen Eingänge des inversen Systems gehängt werden, wodurch ebenfalls benötigte Ableitungen zur Verfügung stehen um die Differentiation von $\boldsymbol{F}(.)$ zu ermöglichen, welche für die Indexreduktion notwendig ist. Der neue Eingang muss dann allerdings auf

die benötigte Ableitung (Länge der Integratorkette) umgerechnet werden, was nicht für jedes Eingangssignal möglich ist.

3.5 Approximationsansätze für inverse Modelle

Auch wenn mit Hilfe von *Modelica/Dymola* relativ einfach inverse Modelle berechnet werden können, müssen zunächst entsprechende Vorkehrungen getroffen werden, damit die, für eine Invertierung benötigten, Bedingung erfüllt sind. Zunächst muss sichergestellt werden, dass alle verwendeten Funktionen eindeutig invertierbar sind.

Dies wurde in dieser Arbeit bereits bei der Modellbildung in Kapitel 2 berücksichtigt. Einige Effekte wie die Haftreibung oder Wirkungsgrade können dadurch nicht exakt berücksichtigt werden, jedoch können approx. Funktionsverläufe, wie sie etwa durch steile Exponentialfunktionen beim Nulldurchgang erzeugt werden können, als Näherung genutzt werden. Diese Form der Näherung kann einige physikalische Effekte, wie das Steckenbleiben eines Gelenks in der Haftphase, nicht beschreiben, jedoch ist die Näherung für eine Vorsteuerung meist ausreichend.

Auch das Bereitstellen der benötigten Ableitungen, der neu gewählten Eingänge des inversen Systems ist normalerweise unproblematisch, wenn auch numerisch kritisch. Am einfachsten kann dies durch Filter entsprechender Ordnung erreicht werden.

Besonders geeignet sind hierbei Tiefpass-Filter, welche kein oder nur ein extrem geringes Überschwingen aufweisen, da sonst die Signale stark verfälscht werden. Sinnvoll ist hierfür etwa ein kritisch gedämpfter Filter. Wie man mit Hilfe eines Filters Ableitungen approximativ berechnen kann sieht man besonders deutlich wenn man sich die Zustandsraumbeschreibung eines Filters ansieht.

Für den kritisch gedämpften Filter der Ordnung n und der Eckfrequenz f_c mit Eingang $u(t)$ und Ausgang $y(t)$ beschreibt dies Gl. (3.32).

$$\omega_c = 2\pi f_c \tag{3.32a}$$
$$\dot{x}_1 = (u - x_1)\omega_c \tag{3.32b}$$
$$\dot{x}_i = (x_{i-1} - x_i)\omega_c \text{ für } i = 2, 3, \ldots, n \tag{3.32c}$$
$$y = x_n \tag{3.32d}$$

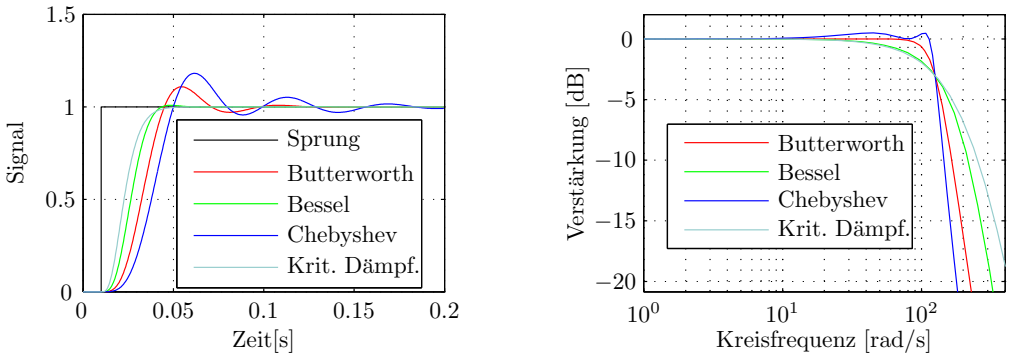
Durch rekursives Einsetzen der Zustandsableitungen $\dot{x}_i$ können approximierte Ableitungen bis zur Ordnung n des Filters erzeugt werden. Dies verdeutlicht Gl. (3.33) für die approximierten Ableitungen über den Filter ($n = 3$) bis zur dritten Ordnung.

$$\dot{y} \approx (x_2 - x_3)\omega_c \tag{3.33a}$$
$$\ddot{y} \approx ((x_1 - x_2)\omega_c - (x_2 - x_3)\omega_c)\omega_c \tag{3.33b}$$
$$y^{(3)} \approx (((u - x_1)\omega_c - (x_1 - x_2)\omega_c)\omega_c \tag{3.33c}$$
$$-((x_1 - x_2)\omega_c - (x_2 - x_3)\omega_c)\omega_c)\omega_c$$

Die durch Filter erzeugten Ableitungen sind nicht exakt und deutlich glätter als durch numerische Differenzen erzeugte Ableitungen. Es muss eine zeitliche Verschiebung (Phasenverschiebung) von $u(t)$ bezüglich $y(t)$, welcher mit sinkenden Frequenzen f_c und steigendem n zunimmt, in Kauf genommen werden.

Theoretisch wäre für die Eckfrequenz f_c daher ein möglichst hoher Wert am besten ($f_c \to \infty$). Numerisch ist dies jedoch nicht realisierbar und auch nicht sinnvoll. Zum einen sind die erstellten Modelle nicht bis zu beliebig hohen Frequenzen genau, zum anderen entstehen erhebliche Probleme für den Integrator bei der Lösung des DAE-Systems, wenn zu hohe Frequenzen zugelassen werden. Die Schrittweite für die Integration muss dadurch stark gesenkt werden, wodurch die Rechenzeit extrem ansteigt. Werden Integratoren verwendet, die mit einer festen Schrittweite rechnen, konvergieren hierdurch die Ergebnisse nicht mehr, was sich durch ein numerisches Zittern (bzw. Grenzzyklus) um die reale Lösung bemerkbar machen kann oder die Integration wird sogar instabil.

Aus diesem Grund sind (numerisch) Filter sinnvoll welche ab der eingestellten Eckfrequenz f_c sehr stark abfallen. Der kritisch gedämpfte Filter erfüllt dies nur bedingt. Er kann aber alternativ in einer höheren Ordnung, als für die Invertierung unbedingt nötig, verwendet werden, wodurch die Steilheit des Abfalls der Verstärkung $|G_f(j\omega)|$ des Filters vergrößert werden kann.



(a) Vergleich der Sprungantwort von vier Filtertypen der Ordnung $n_f = 4$ mit $f_c = 20Hz$.

(b) Vergleich der Verstärkung von vier Filtertypen der Ordnung $n_f = 4$ mit $f_c = 20Hz$.

Abbildung 3.2: Vergleich verschiedener Tiefpassfilter. Die Filter sind jeweils so normalisiert, dass bei der Frequenz f_c eine Absenkung der Amplitude von $3db$ resultiert. Die Abbildung 3.2(a) zeigt den Sprung σ bei $t = 0.01s$ als Eingangssignal sowie die Sprungantworten des kritisch gedämpften Filters, des Besselfilters, des Butterworthfilters und des Chebyshevfilters. Abbildung 3.2(b) zeigt den Verlauf der Verstärkungen der Filter über der Kreisfrequenz des Eingangssignals.

Einen guten Kompromiss zwischen der Steilheit des Filters und geringem Überschwingen des Filters sowie einem glatten Frequenzverlauf im Durchlassbereich, stellt der Besselfilter dar. Das Überschwingen bei der Sprungantwort verringert sich mit der Ordnung des Filters und er besitzt eine konstante Gruppenlaufzeit im Durchlassbereich.

In Abbildung 3.2(a) sind die Sprungantworten von vier gängigen Tiefpassfiltern über der Zeit aufgetragen. Die Filter besitzen alle die Filterordnung $n_f = 4$ und die Eckfrequenz $f_c = 20Hz$. Butterworth- und Chebyshevfilter besitzen zwar einen steilen Abfall in der Verstärkung (siehe Abb. 3.2(b)), sind aber aufgrund ihres starken Überschwingens in der Sprungantwort nicht zur Filterung der Eingänge eines inversen Systems geeignet.

Die Stabilität der Nulldynamik stellt die größte Herausforderung beim Invertieren der Modelle dar. Wie bereits bei dem vereinfachten Robotermodell in Abschnitt 3.3 gezeigt wurde, kann bereits ein zusätzlicher elastischer Freiheitsgrad des Roboters zu einer

instabilen Nulldynamik führen. Sobald diese existiert, kann das System nicht mehr invertiert werden, da das resultierende (inverse) System instabil wäre.

Um dieses Problem zu beseitigen müssen Approximationen oder zusätzliche Algorithmen verwendet werden, um die Modelle so umzugestalten, dass sie invertierbar werden, ohne wesentliche Aspekte des Modells zu vernachlässigen. Im linearen Fall bedeutet dies, es muss eine Approximation für die Inverse $\hat{G}^{-1}$ der Strecke G gefunden werden, so dass gilt $G\hat{G}^{-1} \approx 1$.

Es gibt verschiedene Ansätze, um dies zu erreichen. Im Folgenden sollen verschiedene Möglichkeiten vorgestellt werden.

3.5.1 Variation der Sensorposition

Die Nulldynamik eines Systems hängt immer von den gewählten Ein- und Ausgängen des Systems ab, kann also durch eine entsprechende Wahl beeinflusst werden. In vielen regelungstechnischen Anwendungen ist es sinnvoll, den Sensor direkt am Aktuator zu platzieren, da dies für die Stabilität der Dynamik des geregelten Systems Vorteile bringt. In der Literatur wird die Theorie, die sich mit diesem Phänomen beschäftigt, „collocated[6] control“ genannt. Ist der Sensor weiter vom Aktuator entfernt kann es - im linearen Fall - zur Vertauschung von Polen und Nullstellen des Systems („pole-zero flipping“) kommen, was einer Phasenverschiebung entspricht. Ausführliche Untersuchungen hierzu, bezogen auf unterschiedliche mechanische Systeme, finden sich in [53].

Bei einer Vorsteuerung eines Roboters hat man nun den großen Vorteil, dass prinzipiell an beliebigen Stellen des Systems „gemessen“ werden kann (daher die entsprechenden Größen können in der Simulation berechnet werden). Es existiert allerdings die Vorgabe, dass die Werkzeugspitze des Roboters möglichst exakt auf einer vorgegebenen Trajektorie geführt werden soll. Wie in Abschnitt 3.3 gezeigt kann diese Wahl des Ausgangs des Systems zu einer instabilen Nulldynamik führen.

Im Folgenden wird daher zunächst für ein einfaches Balkensystem untersucht, in wie weit sich die Nulldynamik durch die Verschiebung des Ausgangs des Systems, also des „virtuellen“ Orts des Sensors, beeinflussen lässt. Erkenntnisse die hieraus gewonnen werden, können dann zum Erstellen eines approximativen inversen Robotermodells verwendet werden.

Das untersuchte System besteht aus einem Gelenk an welchem ein Balken nach Abschnitt 2.4.2 befestigt ist (Einspannung: unterstützt-frei). Am Ende des Balkens ist eine Punktmasse als Last befestigt. Eingang u des Systems ist das Antriebsmoment des Gelenks, Ausgang y die Verschiebung der Spitze des Balkens aus der Startstellung in der Bewegungsrichtung welche durch die Rotation um die Antriebsachse erzeugt wird. Abbildung 3.3 stellt den Aufbau dar. Der Balken im Gravitationsfeld $\mathbf{g}$ hat den Durchmesser A, den Elastizitätsmodul E, die Länge l und die homogene Dichte ρ. Er wird mit Hilfe der *FlexibleBodies Library* in *Dymola* simuliert. Zur Untersuchung wird das System linearisiert, anschließend können die Nullstellen N_i und Polstellen P_j des Systems analysiert werden (zur Definition siehe Gleichung (3.1)).

Durch die Verschiebung des Ausgangs y von der Balkenspitze zu einer Position auf der

[6] Collocate ist der englische Begriff für nebeneinanderstehen.

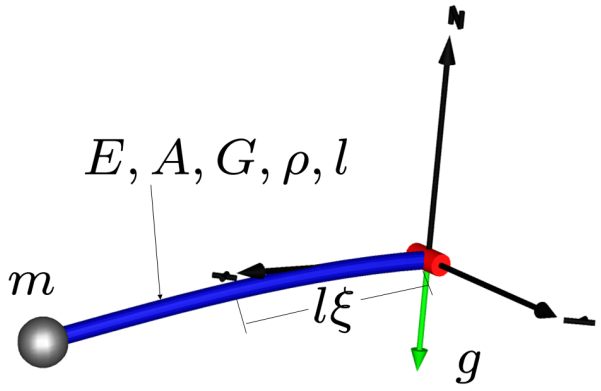


Abbildung 3.3: Aufbau des Testmodells zur Untersuchung des Einflusses der Sensorpositionierung.

Höhe $l\xi$ auf der Balkenlängsachse kann ein verändertes Systemverhalten beobachtet werden. Zunächst wird der Balken mit drei Moden für die Biegung in der XZ-Ebene simuliert. Das linearisierte System besitzt dann $n_G = 8$ Pole und $m_G = 6$ Nullstellen. Variiert man nun den Parameter ξ im Bereich von $0 < \xi \leq 1$, so ergeben sich die in Abbildung 3.4 dargestellten Verläufe der Nullstellen des Systems, wobei die Pole konstant bleiben. Für die Simulation wurden 200 äquidistante Linearisierungen für $0 < \xi \leq 1$ durchgeführt. Die gleiche Simulation wurde mit identischem Balken, allerdings nur mit einem Mode für die Biegung durchgeführt ($n_G = 4$ Pole und $m_G = 2$ Nullstellen). Die Ergebnisse dieser Simulation zeigt Abbildung 3.5. Die Polstellen dieses Systems wurden nicht aufgetragen, da sie sich wie im Fall mit drei Moden nicht verändern.

Wie man den Abbildungen 3.4 und 3.5 entnehmen kann, gibt es Bereiche für den Parameter ξ in denen keine Nullstellen mit positivem Realteil auftreten. Ist dies der Fall ist das System stabil invertierbar.

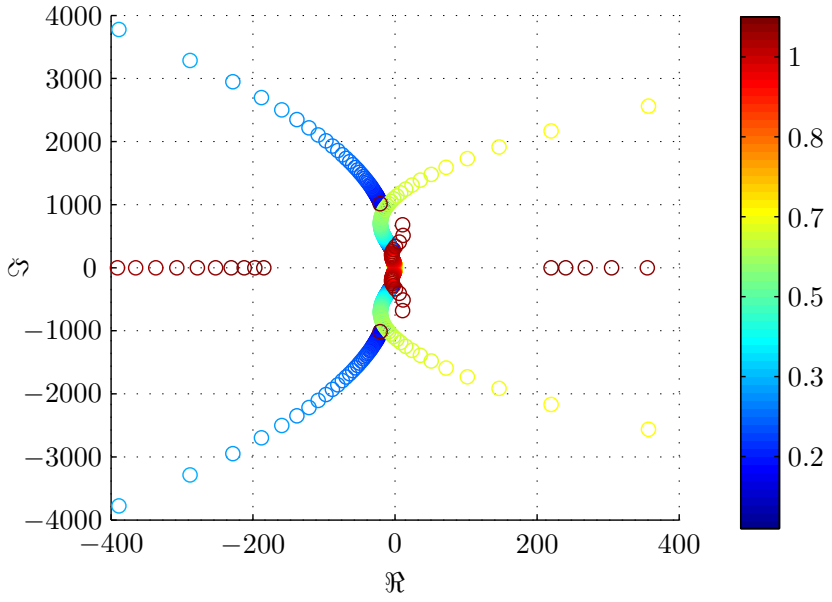
Für das vereinfachte System mit nur einem Mode für die Biegung tritt dieser Effekt bereits für $\xi < 0.9$ auf (ab hellrotem Bereich), für den Fall mit drei Moden erst ab $\xi < 0.3$ (blauer Bereich).

Die Nulldynamik diese Systems hängt also von der Position des Ausganges und dem Detaillierungsgrad des Modells ab.

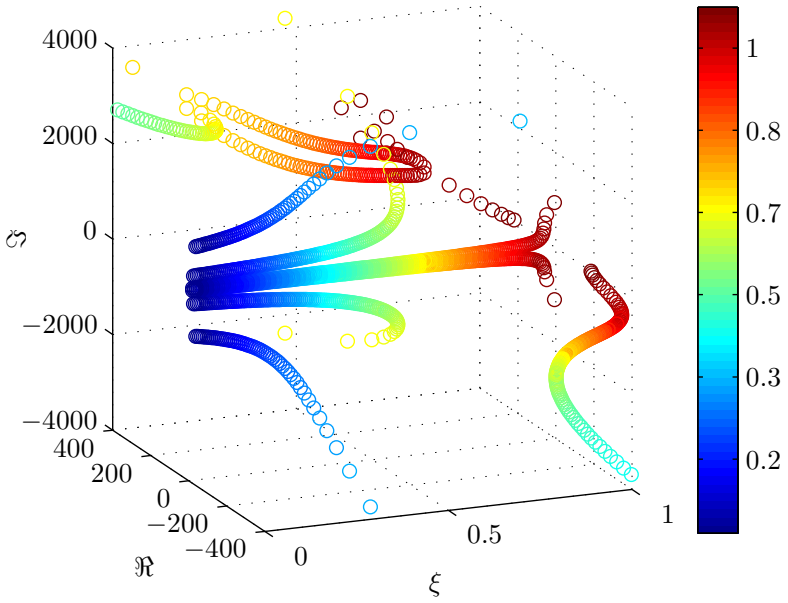
Für ein approximatives inverses Modell hat man daher die Möglichkeit, den Detaillierungsgrad und den Ausgang des Systems so zu modifizieren, dass die Nulldynamik stabil ist (bzw. keine Nullstellen N_i mit positivem Realteil vorhanden sind).

Um zu überprüfen ob diese Überlegungen, welche durch eine Betrachtung des linearisierten Systems gewonnen wurden, auch im nichtlinearen Fall korrekt sind wurde ein weiteres Simulationsexperiment durchgeführt. Den Aufbau dieser Simulation zeigt Abbildung 3.6.

Dieses Modell besteht aus zwei Balken, die analog zum vorherigen Beispiel aufgebaut sind (Parameter: E, A, G, ρ, l). Wieder befindet sich jeweils eine Last m am Ende der Balken. Der zweite Balken ist am Balkenende des ersten über ein zusätzliches Gelenk befestigt. Ziel des Experiments ist es über ein vereinfachtes invertierbares Modell, Vor-

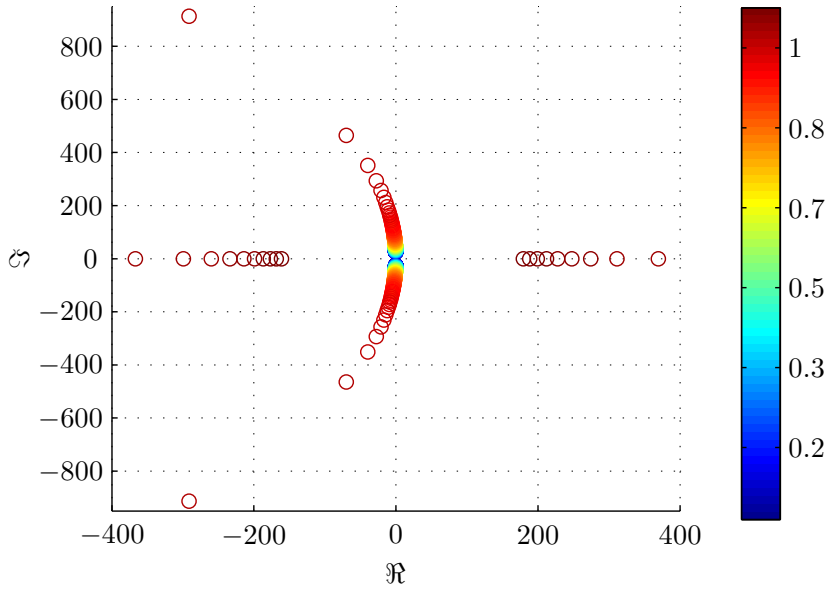


(a) Nullstellen N_i des linearisierten Systems in Abhängigkeit von ξ

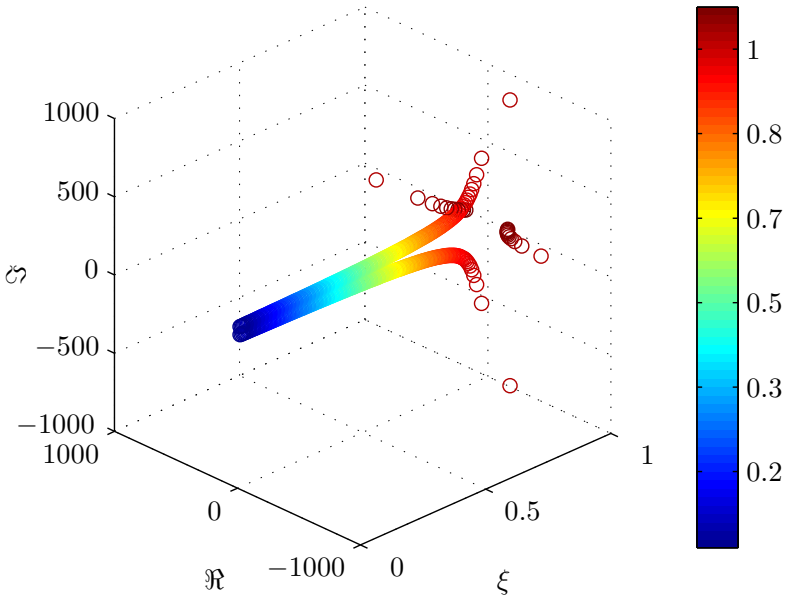


(b) Nullstellen N_i des linearisierten Systems in Abhängigkeit von ξ (in 3D-Darstellung, wobei der Parameter ξ auf der x-Achse dargestellt ist).

Abbildung 3.4: Nullstellen N_i in Abhängigkeit von Parameter ξ für das Balken Modell mit 3 Biegemoden. Farblicher Verlauf: Dunkelblau $\xi = 0$ bis dunkelrot $\xi = 1$.



(a) Nullstellen N_i des linearisierten Systems in Abhängigkeit von ξ



(b) Nullstellen N_i des linearisierten Systems in Abhängigkeit von ξ (in 3D-Darstellung, wobei der Parameter ξ auf der x-Achse dargestellt ist).

Abbildung 3.5: Nullstellen N_i in Abhängigkeit von Parameter ξ für das Balken Modell mit nur einem Biegemode. Farblicher Verlauf: Dunkelblau $\xi = 0$ bis dunkelrot $\xi = 1$.

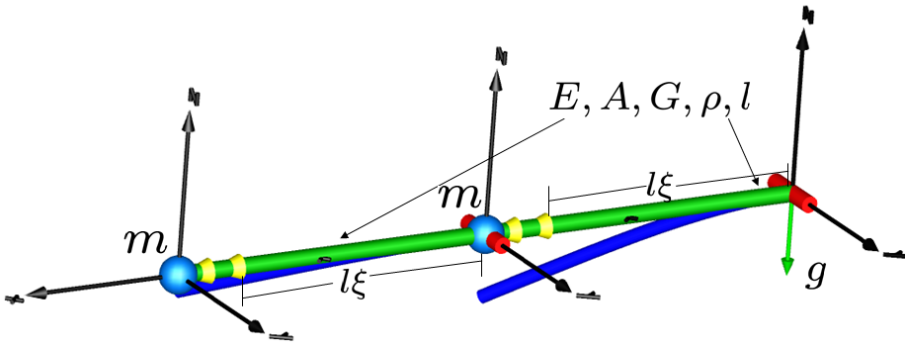


Abbildung 3.6: Simulationsaufbau zur approx. Invertierung über die Sensorplatzierung. In grün der unverformte Balken, in blau die (überhöhte) Darstellung der verformten Balken, in hellblau die Massen am Balkenende und in rot die Gelenke der Balken, wobei das zweite Gelenk am Balkenende des ersten Balkens befestigt ist.

steuergrößen für den Momentenverlauf $\tau_i(t)$ der zwei Gelenke zu berechnen, so dass der zweite Massepunkt am Ende des zweiten Balkens möglichst exakt einer vorgegebenen Trajektorie folgt.

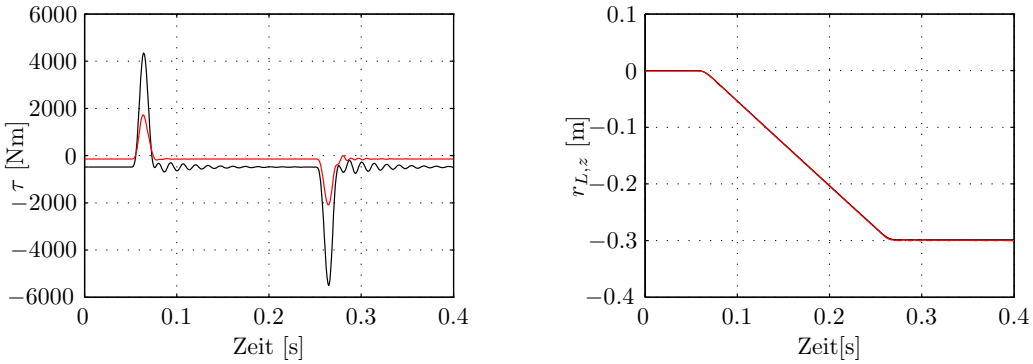
Als approx. inverses Modell werden zwei Balken mit nur einem Mode gewählt. Als Ausgang (bzw. Eingang des inversen Modells) werden die Verschiebungen (tangential bezüglich des Kreises, der durch eine Rotation des Balkens um das Antriebsgelenk beschrieben wird) an der Stelle $\xi = 0.9$ der zwei Balken gewählt. Als Eingang (bzw. Ausgang des inv. Modells) die Antriebsmomente der Gelenke.

Für das Streckenmodell werden identische Balken, allerdings mit drei Moden für die Biegung, gewählt. Eingang sind hier die Momente τ_i der zwei Gelenke und Ausgänge die Verschiebungen an den Stellen $\xi = 1$, also der Massenpunkte.

Als Trajektorie wird ein rampenförmiger Verlauf für den Endmassepunkt vorgegeben, welcher über ein Besselfilter gefiltert wird, um nötige Differentiation der Eingangsgrößen zu ermöglichen. Dieser Verlauf wird über ein starres kinematisches Modell in zwei Solltrajektorien der Balken an den (jeweiligen) Stellen $\xi = 0.9$ umgerechnet und dient als Eingangsgröße für das approx. inverse Modell.

Die so berechneten Ausgänge τ_i dienen als Eingangsgrößen für die zwei Gelenke des Streckenmodells (mit drei Biegemoden). Damit die statischen Verformungen der Balken durch die Schwerkraft $\mathbf{g}$ ausgeglichen wird, werden für das Streckenmodell zur Initialisierung zusätzlich die aus dem inversen Modell berechneten Startwinkel der Gelenke vorgegeben.

Abbildung 3.7 zeigt einige Ergebnisse dieser Simulation. Das Streckenmodell folgt der geforderten Solltrajektorie sehr genau, jedoch bleibt am Ende der Bewegung ein leichtes Zittern übrig, was durch die Ungenauigkeit der Approximation entsteht. Da nur Momente τ_i für die zwei Gelenke vorgegeben werden driftet der Balken am Ende zudem leicht von der Solltrajektorie weg. Dies liegt daran, dass kein Regler und keine Rückführung im Modell verwendet wurden, sondern lediglich die berechneten Vorsteuergrößen für die Momentenverläufe (sowie der Startwinkel zur Initialisierung) verwendet wurden. Durch die Schwerkraft ist die Endposition keine Ruhelage des Systems.



(a) Durch das approx. inverse Modell berechnete Antriebsmomente $\tau_i(t)$ (τ in Nm über der Zeit in s). Schwarz: $\tau_{a,1}(t)$, Rot: $\tau_2(t)$.

(b) Vergleich der Solltrajektorie (schwarz) und dem Ausgang (rot) der Strecke (Verschiebung bei $\xi = 1$ des zweiten Balkens in m über der Zeit in s).

Abbildung 3.7: Ergebnisse der Simulation mit dem approximativen inversen Modell. Gezeigt sind die berechneten Vorsteuergrößen τ_i sowie der Ausgang der Strecke im Vergleich zur Solltrajektorie.

Am Verlauf der berechneten Momentenverläufe (Abbildung 3.10(b)) sieht man, dass auch nach dem Ende der Bewegung, immer noch ein nicht konstanter Momentenverlauf vorgegeben werden muss, um die Schwingung des schwach gedämpften Balkens zu kompensieren. Auch in der nichtlinearen Simulation kann für $\xi > 0.9$ keine Lösung mehr für das inverse Modell gefunden werden (Modell wird instabil).

Analog zu dem Modell in diesem Beispiel, kann auch ein approx. inverses Modell für den gesamten Roboter aufgebaut werden, wobei die Schwinge und der Arm als Balken modelliert werden. Hierzu müssen jedoch auch noch einige andere Methoden verwendet werden, die noch in diesem Kapitel vorgestellt werden.

3.5.2 Approximation der elastischen Verformungen über ein Parallelmodell

Ein weiterer Ansatz zur Approximation der inversen Modelle baut auf dem Modellierungsansatz auf, elastische Strukturen über Starrkörper mit zusätzlichen Gelenken, die mit Feder-Dämpferelementen gestützt werden, zu approximieren. Wie in Abschnitt 2.4.3 beschrieben wurde können hierdurch Systeme mit geringen elastischen Verformungen modelliert werden. Die Bewegungsgleichungen für einen so modellierten Roboter ergeben sich dann nach Gl. (2.38).

Ein Ansatz, um diese Art der Modelle zu invertieren ist es, eine Approximation der Verformungen über ein Parallelmodell zu berechnen.

Da es das Ziel ist, die Last des Roboters am TCP möglichst exakt einer gewünschten Trajektorie folgen zu lassen, kann diese Trajektorie $\mathfrak{T}_L$ als bekannt vorausgesetzt werden.

Aus der Trajektorie der Last, welche sich beim Roboter am Ende der Kinematischen Kette befindet, können über die inverse Starrkörperkinematik abtriebsseitige Gelenk-

winkel[7] $\boldsymbol{\varphi}_{a,r}$ des (ideal) starren Roboters berechnet werden.

Ein System von Differentialgleichungen erster Ordnung zur Approximation der Verformungen ergibt sich nun aus den Bewegungsgleichungen (2.38) durch Vernachlässigung von $\boldsymbol{M}_{ee}$ und Einsetzen der idealen abtriebsseitigen Winkel $\boldsymbol{\varphi}_{a,r}$ in die Bewegungsgleichungen. Zur einfacheren Darstellungen wird $\boldsymbol{h}_e$ aufgeteilt in $\boldsymbol{h}_e = \boldsymbol{h}_e^* + \boldsymbol{K}_e\boldsymbol{\varphi}_e + \boldsymbol{D}_e\dot{\boldsymbol{\varphi}}_e$. Hierin sind $\boldsymbol{K}_e$ die Struktursteifigkeitsmatrix und $\boldsymbol{D}_e$ die Strukturdämpfungsmatrix, welche sich aus den Feder-Dämpferelementen zur Modellierung der Strukturelastizitäten ergeben.

Eine Approximation der Verformungen $\hat{\boldsymbol{\varphi}}_{\boldsymbol{e}}$ kann dann über Gl. (3.34) berechnet werden.

$$\underbrace{\boldsymbol{M}_{ea}|_{(\boldsymbol{\varphi}_a \equiv \boldsymbol{\varphi}_{a,r} \wedge \boldsymbol{\varphi}_e \equiv \boldsymbol{0})}}_{\boldsymbol{M}_{ea,r}} \ddot{\boldsymbol{\varphi}}_{a,r} + \underbrace{\boldsymbol{h}_e^*|_{(\boldsymbol{\varphi}_a \equiv \boldsymbol{\varphi}_{a,r} \wedge \boldsymbol{\varphi}_e \equiv \boldsymbol{0})}}_{\boldsymbol{h}_{e,r}} = -\boldsymbol{K}_e\hat{\boldsymbol{\varphi}}_{\boldsymbol{e}} - \boldsymbol{D}_e\dot{\hat{\boldsymbol{\varphi}}}_e \tag{3.34}$$

Dieses Differentialgleichungssystem ist stabil lösbar (siehe hierzu auch Abschnitt 3.8.2) für die in Kapitel 2 vorgestellten Federkennlinien.

In *Modelica* kann Gl. (3.34) darüber realisiert werden, dass ein zum elastischen Roboter identisches Starrkörpermodell $\mathfrak{S}_r$ (Parallelmodell) des zu invertierenden elastischen Systems $\mathfrak{S}$ aufgestellt wird.

Nun werden an den Stellen im Starrkörpersystem $\mathfrak{S}_r$, an welchen sich in $\mathfrak{S}$ Gelenke mit Federdämpfersystemen - welche die Elastizität der Struktur approximieren - befinden, die Schnittmomente $\boldsymbol{\tau}_{\mathfrak{S}_r j}$ über Sensormodule berechnet. Über die lineare[8] Differentialgleichung (3.35) können nun über die Vektoren der Schnittkräfte $\boldsymbol{\tau}_{\mathfrak{S}j}$ die entsprechende Verformungen $\hat{\boldsymbol{\varphi}}_e = (\varphi_{e,1}, \ldots, \varphi_{e,n_e})^T$ dieser Gelenke approximiert werden.

$$\hat{\tau}_{e,j} = -\boldsymbol{\tau}_{\mathfrak{S}j}^T \mathbf{n}_j = -\left(\tau_{\mathfrak{S}jx}, \tau_{\mathfrak{S}jy}, \tau_{\mathfrak{S}jz} \right) \begin{pmatrix} n_{xj} \\ n_{yj} \\ n_{zj} \end{pmatrix} \tag{3.35a}$$

$$\hat{\tau}_{e,j} = c_j\hat{\varphi}_{e,j} + d_j\dot{\hat{\varphi}}_{e,j} \tag{3.35b}$$

Hierbei ist $\mathbf{n}_j$ der auf die Länge 1 normierte Rotationsvektors des Gelenks j mit der Steifigkeit c_j und Dämpfung d_j.

Sind die Winkel der elastischen Gelenke $\hat{\boldsymbol{\varphi}}_e$ berechnet, so können diese in die Gleichungen des elastischen Systems $\mathfrak{S}$ eingesetzt werden. Für das nun entsprechend verformte System muss dann erneut die inverse Kinematik bezüglich $\mathfrak{T}_L$ berechnet werden, um die Last des verformten Systems wieder auf der Trajektorie $\mathfrak{T}_L$ zu führen. Ist diese nicht geschlossen lösbar müssen approx. Korrekturterme berechnet werden. Hierzu werden in Abschnitt 3.6 verschiedene Verfahren vorgestellt. Das Ergebnis dieser Berechnung sind dann die entsprechenden korrigierten Gelenkwinkel $\boldsymbol{\varphi}_{a,e}$ des verformten Systems.

Über das Einsetzen von $\boldsymbol{\varphi}_{a,e}$ und $\hat{\boldsymbol{\varphi}}_{\boldsymbol{e}}$ und deren Ableitungen in die Bewegungsgleichungen von $\mathfrak{S}$ können die Gelenkmomente des so approximativ invertierten Systems $\hat{\mathfrak{S}}^{-1}$ berechnet werden.

[7] Alle Größen, welche sich auf das Starrkörpermodell beziehen werden mit einem tiefgestellten r gekennzeichnet. Das r leitet sich von dem englischen Wort „rigid“ ab.

[8] Hier sind prinzipiell auch nichtlineare Differentialgleichungen möglich um etwa nichtlineare Feder- bzw. Dämpfer-Kennlinien modellieren zu können.

Die Annahme, dass die Winkel der elastischen Verformungen approximativ aus den Starrkörper-Schnittkräften berechnet werden können, müssen für die Modelle dabei anhand von Simulationen überprüft werden. Diese Approximation sollte dabei nur gemacht werden, wenn die Verformungen klein sind, bei großen elastischen Deformationen ist diese Methode zu ungenau.

Im Folgenden soll dieser Ansatz an einem einfachen Beispiel beschrieben werden. Dabei wird gezeigt, dass durch dieses Vorgehen immer eine Approximation für die Inverse gefunden werden kann, auch wenn das System - aufgrund von instabiler Nulldynamik - nicht exakt invertiert werden kann.

Das Beispielsystem besteht aus zwei Gelenken (jeweils um die z-Achse), welche durch Stäbe verbunden sind (Abb. 3.8). Die Länge des Stabes von Gelenk 1 zu Gelenk 2 beträgt $l\xi$, von Gelenk 2 zu dem Starrkörperschwerpunkt $l(1-\xi)$. Hinter dem Starrkörper ist noch einmal ein Stab der Länge x_l angebracht (die Werkzeugspitze des Manipulators). Das zweite Gelenk wird über ein Feder-Dämpfersystem (c, d) gestützt, das

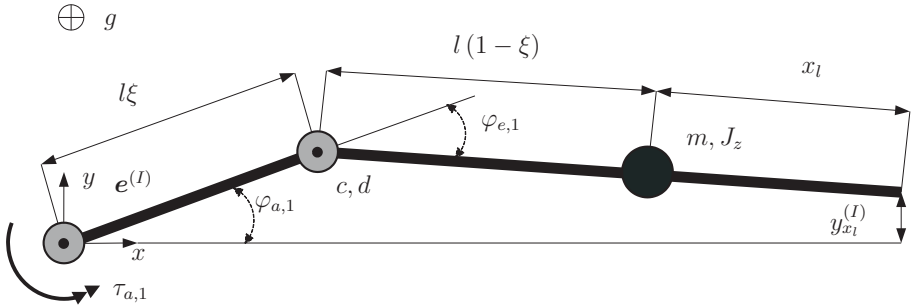


Abbildung 3.8: Beispielsystem zur Invertierung über ein Parallelmodell. Das System besteht aus zwei Gelenken, wovon das 2. über ein Feder-Dämpfersystem (c, d) gestützt wird, sowie einem Starrkörper (m, J_z).

erste Gelenke durch das Moment $\tau_{a,1}$ angetrieben. Nach dem zweiten Gelenk ist ein Starrkörper befestigt mit der Masse m und der Trägheit J_z um die z-Achse. Berechnet man dieses System, mit Hilfe der Gleichungen von Lagrange zweiter Art, erhält man die BewegungsGleichung (3.36).

$$\boldsymbol{M}(\boldsymbol{\varphi})\ddot{\boldsymbol{\varphi}} + \boldsymbol{h}(\boldsymbol{\varphi}, \dot{\boldsymbol{\varphi}}) = \boldsymbol{\tau}_{ext} \tag{3.36}$$

$$\boldsymbol{\tau}_{ext} = \begin{pmatrix} \tau_{a,1} \\ 0 \end{pmatrix}; \boldsymbol{M} = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}; \boldsymbol{\varphi} = \begin{pmatrix} \varphi_{a,1} \\ \varphi_{e,1} \end{pmatrix}$$

$$\boldsymbol{h} = \begin{pmatrix} (m\xi^2 l^2 \dot{\varphi}_{e,1}^2 - ml^2 \dot{\varphi}_{e,1}^2 \xi - 2ml^2 \dot{\varphi}_{a,1} \xi \dot{\varphi}_{e,1} + 2m\xi^2 l^2 \dot{\varphi}_{a,1} \dot{\varphi}_{e,1}) \sin(\varphi_{e,1}) \\ c\varphi_{e,1} + d\dot{\varphi}_{e,1} - m\xi^2 l^2 \dot{\varphi}_{a,1}^2 \sin(\varphi_{e,1}) + ml^2 \dot{\varphi}_{a,1}^2 \xi \sin(\varphi_{e,1}) \end{pmatrix}$$

$$\begin{aligned} M_{11} &= ml^2 - 2m\xi^2 l^2 \cos(\varphi_{e,1}) + 2ml^2 \xi \cos(\varphi_{e,1}) - 2m\xi l^2 + 2m\xi^2 l^2 + J_z \\ M_{21} &= M_{12} = J_z - 2m\xi l^2 + ml^2 \xi \cos(\varphi_{e,1}) - m\xi^2 l^2 \cos(\varphi_{e,1}) + m\xi^2 l^2 + ml^2 \\ M_{22} &= J_z + m\xi^2 l^2 - 2m\xi l^2 + ml^2 \end{aligned}$$

Ziel der Invertierung soll es sein das Moment $\tau_{a,1}$ und Winkel $\varphi_{a,1}$ zu berechnen, für eine gegebene Trajektorie $\mathfrak{T}_{x_l}$, der Spitze des Stabes am Ende des Systems. Da nur ein Freiheitsgrad steuerbar ist, soll hierbei nur die y-Koordinate im Inertialsystem $y_{x_l}^{(I)}$ berücksichtigt werden (siehe Abb. 3.8). Diese ergibt sich nach Gl. (3.37).

$$y_{x_l}^{(I)} = (x_l + l(1-\xi))\sin(\varphi_{a,1} + \varphi_{e,1}) + \sin(\varphi_{a,1})\xi l \tag{3.37}$$

Um die Stabilität der Inversen dieses System zu analysieren können wir es linearisieren und die Nullstellen des linearisierten Systems betrachten. Hierzu linearisieren wir das System mit einer Taylorentwicklung für kleine Winkel $\varphi_{a,1}, \varphi_{e,1}$ und bringen es auf Zustandsform (Gl. (3.38)).

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}u \tag{3.38a}$$
$$y = \boldsymbol{C}^T\boldsymbol{x} \tag{3.38b}$$

$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-m\xi l^2 c + J_z c + ml^2 c}{\xi^2 l^2 m J_z} & \frac{ml^2 d - m\xi l^2 d + J_z d}{\xi^2 l^2 m J_z} \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{-(J_z c + ml^2 c)}{\xi^2 l^2 m J_z} & \frac{-J_z d - ml^2 d}{\xi^2 l^2 m J_z} \end{pmatrix}$$

$$\boldsymbol{B} = \begin{pmatrix} 0 \\ \frac{J_z + m\xi^2 l^2 - 2m\xi l^2 + ml^2}{\xi^2 l^2 m J_z} \\ 0 \\ \frac{-(J_z + ml^2 - m\xi l^2)}{\xi^2 l^2 m J_z} \end{pmatrix}; \boldsymbol{C} = \begin{pmatrix} x_l + l \\ 0 \\ l - \xi l + x_l \\ 0 \end{pmatrix}; \boldsymbol{x} = \begin{pmatrix} \varphi_{a,1} \\ \dot{\varphi}_{a,1} \\ \varphi_{e,1} \\ \dot{\varphi}_{e,1} \end{pmatrix}; u = \tau_{a,1};$$

Von der Zustandsform ausgehend können wir die Übertragungsfunktion $G(s)$ nach Gl. (3.41) aufstellen.

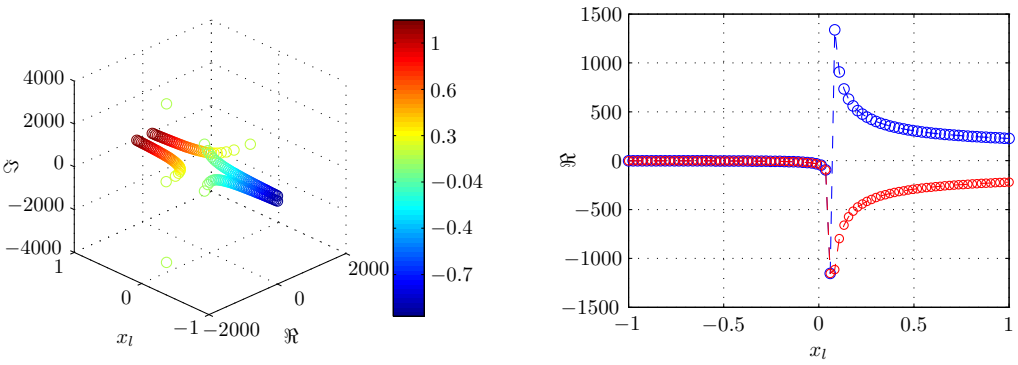
$$G(s) = \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B} \tag{3.39}$$

Hieraus können die Nullstellen des linearisierten Systems errechnet werden. Das System besitzt zwei komplexe Nullstellen N_1 und N_2 (Gl. (3.40)).

$$N_1 = \gamma(ld + dx_l + ((x_l + l)(4l^2 x_l m\xi^2 c - 4cx_l ml^2\xi - ld^2 + 4l\xi J_z c - d^2 x_l)^{\frac{1}{2}})) \tag{3.40a}$$
$$N_2 = \gamma(ld + dx_l - ((x_l + l)(4l^2 x_l m\xi^2 c - 4cx_l ml^2\xi - ld^2 + 4l\xi J_z c - d^2 x_l)^{\frac{1}{2}})) \tag{3.40b}$$

$$\text{mit } \gamma = \frac{1}{-2\xi l(x_l m\xi l + J_z - lx_l m)}$$

Wie man sieht ist die Lage der Nullstellen von den Parametern abhängig. Berechnet man die Lage der Nullstelle in Abhängigkeit der Parameter, so erkennt man, dass man durch die Änderung der Länge von x_l Nullstellen mit positivem Realteil erzeugen kann (Abb. 3.9). Sind Nullstellen mit positivem Realteil vorhanden, ist das inverse System instabil.



(a) Lage der komplexen Nullstellen N_1 und N_2 in Abhängigkeit von Parameter x_l.

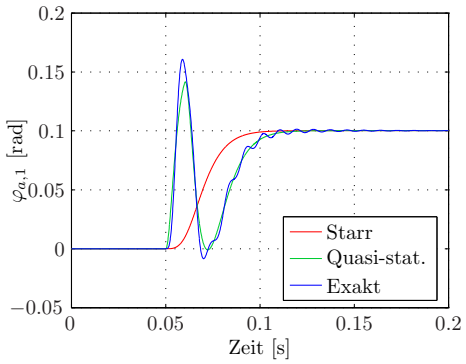
(b) Realteil der Nullstellen $\Re(N_1)$ und $\Re(N_2)$ in Abhängigkeit von Parameter x_l.

Abbildung 3.9: Lage der Nullstellen des Beispielsystems in Abhängigkeit von x_l. Für die übrigen Parameter wurden folgende Werte verwendet: $l = 2m, \xi = 0.2, m = 100kg, c = 1e6\frac{N \cdot m}{rad}, d = 150\frac{N \cdot m \cdot s}{rad}, J_z = 10kg \cdot m^2$.

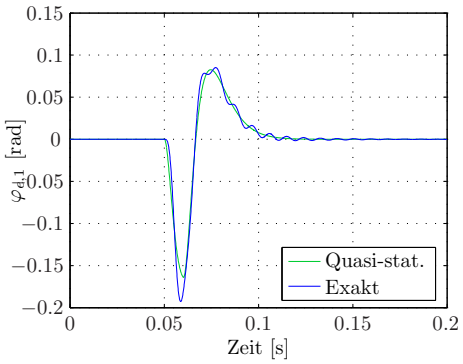
Die Approximation der Verformung $\varphi_{e,1}$ wird über ein starres Parallelmodell $\mathfrak{S}_r$ berechnet. Das Schnittmoment $\hat{\tau}_{e,1}$ in Achsrichtung des zweiten Gelenks im starren System $\mathfrak{S}_r$ ergibt sich nach Gl. (3.41), wobei $\varphi_{a,1,r}$ der Gelenkwinkel des ersten Gelenks des starren Systems darstellt, welcher direkt aus der inversen Kinematik des starren Systems $\mathfrak{S}_r$ berechnet werden kann. Mit Hilfe des Moments $\hat{\tau}_{e,1}$ kann die Verformung $\hat{\varphi}_{e,1}$ approximativ berechnet werden.

$$\hat{\tau}_{e,1} = \ddot{\varphi}_{a,1,r}(-ml^2\xi + ml^2 + J_z) \tag{3.41a}$$
$$y_{x_l}^{(I)} = (x_l + l)\sin(\varphi_{a,1,r}) \tag{3.41b}$$
$$0 = c\hat{\varphi}_{e,1} + \dot{\hat{\varphi}}_{e,1}d + \hat{\tau}_{e,1} \tag{3.41c}$$

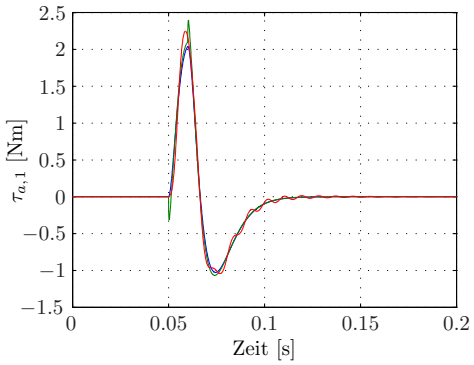
Über Gl. (3.37) kann dann $\varphi_{a,1}$ für ein vorgegebenes $y_{x_l}^{(I)}$ berechnet werden. Für dieses einfache Beispiel kann die inverse Kinematik noch geschlossen gelöst werden, für komplexere Modelle ist dies im allg. Fall aber nicht mehr möglich, dann müssen Näherungsverfahren wie sie in Abschnitt 3.6 vorgestellt werden verwendet werden. Anschließend kann $\hat{\tau}_{a,e,1}$ dann mit Hilfe der ersten Zeile der Bewegungsgleichung (3.36) des Systems berechnet werden, wobei $\varphi_{e,1}$ gleich $\hat{\varphi}_{e,1}$ und $\varphi_{a,1}$ zu $\varphi_{a,e,1}$ (dem über die inverse Kinematik korrigierten Winkel) gesetzt wird (inklusive der Ableitungen der Terme). Um dieses Gleichungssystem zu lösen ist es notwendig, dass $y_{x_l}^{(I)}$ mindestens zweimal differenzierbar ist. Durch die Approximation kann das System nun für beliebige Parameter invertiert werden, da $\hat{\varphi}_{e,1}$ stets berechnet werden kann. Für die inverse Kinematik des verformten Systems muss jedoch weiterhin eine Lösung berechnet werden. Durch Einsetzen der durch die inverse Kinematik bestimmten korrigierten Winkel $\varphi_{a,e,1}$ und $\hat{\varphi}_{e,1}$, sowie deren Ableitungen, in die Bewegungsgleichungen des Systems kann immer eine Approximation für $\tau_{a,1}$ berechnet werden, wodurch das System approximativ invertierbar ist. Abbildung 3.10 zeigt das Ergebnis der Approximation im Vergleich zur exakten Invertierung und die Invertierung des starren Systems ($\varphi_{e,1} = 0$). Für die Simulation wurden die Parameter wie folgt angenommen: $l = 2m, \xi = 0.2, m = 100kg, c = 1e6\frac{N \cdot m}{rad}, d = 150\frac{N \cdot m \cdot s}{rad}, J_z = 10kg \cdot m^2, x_l = 0m$. Für $x_l = 0m$ ist das System auch exakt über Indexreduktion invertierbar (siehe Abb. 3.9, es gibt dann keine Nullstellen mit positivem Realteil).



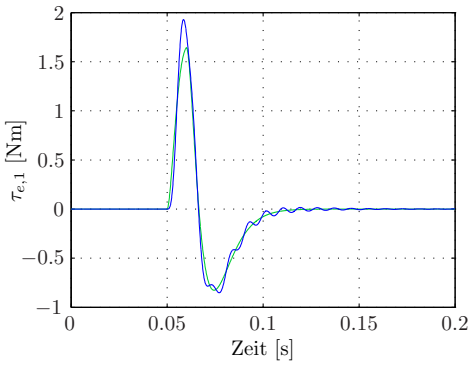
(a) Vergleich der berechneten Winkel $\varphi_{a,1}$ bei Gelenk 1.



(b) Vergleich der berechneten Winkel $\varphi_{e,1}$ bei Gelenk 2.



(c) Vergleich der berechneten Momente $\tau_{a,1}$ bei Gelenk 1.



(d) Vergleich der berechneten Momente $\tau_{e,1}$ bei Gelenk 2.

Abbildung 3.10: Verlauf der Winkel und Momente am ersten und zweiten Gelenk für das Beispielsystem. Rot: Lösung für starres System. Grün: Quasi-statische Lösung. Blau: Exakte Lösung.

Durch die Approximation kann eine gute Schätzung für $\varphi_{a,1}$ und $\varphi_{e,1}$ sowie $\tau_{e,1}$ bestimmt werden. Die Approximation führt jedoch zu Abweichungen beim Momentenverlauf $\tau_{a,1}$, vor allem bei großen Änderungen von $\varphi_{e,1}$.

Ob ein System mit dieser Methode - in ausreichender Genauigkeit - invertiert werden kann, können letztlich nur Simulationen zeigen. Diese Methode hat jedoch den großen Vorteil, dass instabile Nulldynamik keine Probleme bereitet, da alle Kräfte und Verformungen der elastischen Elemente immer berechnet werden können.

Es ist allerdings immer noch das inverse kinematische Problem für das verformte System $\mathfrak{S}$ – zusätzlich zu der inversen Kinematik des starren Systems $\mathfrak{S}_r$ – zu lösen.

Auf die inverse Kinematik für die verformten Systeme, bei denn auch neue Singularitäten auftauchen können, wird in Abschnitt 3.6 genauer eingegangen.

3.5.3 Invertierung bezüglich virtueller Ausgänge

Ein weiterer Ansatz zum invertieren von Systemen mit instabiler Nulldynamik (bzw. Nullstellen mit positivem Realteil im Linearen) ist es, die Ausgänge des Systems so zu ändern, dass bezüglich dieser neuen, virtuellen Ausgänge das System (stabil) invertiert werden kann. Hierzu existieren für spezielle Systemklassen unterschiedliche Algorithmen.

In [54] und [55] wird (mathematisch) ein Ausgang konstruiert, über welchen ein Einachs-System mit Euler-Bernoulli-Balken für spezielle Bewegungsformen (PTP[9]-Bewegungen und Beschleunigungsübergänge) invertiert werden kann, um die entsprechenden Stellgrößen zu berechnen.

Ein Ansatz von Niemiec und Kravaris (siehe z. B. [56, 57, 58]) um virtuelle Ausgänge[10] zu definieren ist es, Ausgänge zu finden die statisch identisch mit den ursprünglichen Ausgängen des Systems sind, im dynamischen Fall jedoch so modifiziert werden, dass keine instabile Nulldynamik erzeugt wird. Dieser Ansatz soll hier kurz vorgestellt werden und dabei auch auf die Anwendung der Methode auf ein strukturelastisches Robotermodell eingegangen werden. Die Methode wird hier so erweitert, dass auch im dynamischen Fall, für beliebige Bewegungen eine möglichst genaue Trajektorienverfolgung erreicht werden kann.

Ausgangspunkt für die Theorie nach Niemiec und Kravaris ist ein nichtlineares System in der Form von Gl. (3.42) mit $\boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{u} \in \mathbb{R}^m, \boldsymbol{y} \in \mathbb{R}^m$.

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{G}(\boldsymbol{x})\boldsymbol{u}, \boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{3.42a}$$

$$\boldsymbol{y} = \boldsymbol{h}_f(\boldsymbol{x}) \tag{3.42b}$$

Auch die Bewegungsgleichungen des Roboters können auf diese Form gebracht werden, wenn als Eingang die Momente der Achsen und als Ausgang des Systems die Position und Orientierung des TCPs gewählt werden. Für die Theorie muss vorausgesetzt werden, dass $\boldsymbol{f}, \boldsymbol{G}$ und $\boldsymbol{h}_f$ hinreichend glatte und zusammenhängende Funktionen sind, was auf die in Kapitel 2 beschriebenen Roboter-Modelle (mit differenzierbaren Näherungen) zutrifft.

Mit $\boldsymbol{h}_A$ werden die neuen virtuellen Ausgänge für dieses System bezeichnet, welche so konstruiert werden, dass sie im statischen Fall mit den ursprünglichen Ausgängen des Systems identisch sind, aber keine instabile Nulldynamik mehr aufweisen. Die Berechnungsvorschrift um diese zu erhalten wird im Folgenden kurz skizziert, für Details zu den Voraussetzungen und Beweise hierzu sei auf [56] und [57] verwiesen.

$$\boldsymbol{h}_A(\boldsymbol{x}) = \begin{pmatrix} h_{A_1}(\boldsymbol{x}) \\ \vdots \\ h_{A_{m-1}}(\boldsymbol{x}) \\ h_{A_m}(\boldsymbol{x}) \end{pmatrix} = \begin{pmatrix} h_1(\boldsymbol{x}) \\ \vdots \\ h_{m-1}(\boldsymbol{x}) \\ h_m(\boldsymbol{x}) + \sum_{j=1}^{n-m} \psi_j q_j(\boldsymbol{x}) \end{pmatrix} \tag{3.43}$$

Gleichung (3.43) zeigt, dass alle Ausgänge bis auf Ausgang m identisch mit den ursprünglichen Ausgängen gewählt werden können. Ausgang m wird durch eine gewichtete Summe über, vom Zustand $\boldsymbol{x}$ des Systems abhängige, Terme modifiziert.

[9] PTP: Point to Point. Verfahren zum Zielpunkte ohne exakte Definition des Bahnverlaufs im (kartesischen) Raum. Üblicherweise eine Gerade im Achswinkelraum.

[10] Im Englischen mit „auxiliary outputs“ bezeichnet.

Die Grundidee zu den virtuellen Ausgängen ist Folgende: Die Konstanten ψ_j werden so bestimmt, dass das System mit den virtuellen Ausgängen nach Gl. (3.43) an einem gewünschten (End-) Zustand minimalphasig wird. Das bedeutet, dass die Positionen der Nullstellen – der Linearisierung des nichtlinearen Systems mit virtuellem Ausgang – so geändert werden, dass sie keinen positiven Realteil mehr besitzen. Die Terme $q_j(\boldsymbol{x})$ werden dabei so konstruiert, dass sie beim gewünschten Zustand (um welchen auch linearisiert wird) zu Null werden und damit stationär keine Abweichung zu den ursprünglichen Ausgängen des Systems mehr auftritt.

Um zu den Termen $q_j(\boldsymbol{x})$ zu gelangen muss zunächst die Matrix $\boldsymbol{Q}$ aufgestellt werden (Gl. (3.44)). Die Notation $[\dots]_{(k,l)}$ steht im Folgenden für den Ausdruck in der k-ten Zeile und der l-ten Spalte.

$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{G}_{(n-m+1,1)}(\boldsymbol{x}) & \dots & \boldsymbol{G}_{(n-m+1,m)}(\boldsymbol{x}) \\ \vdots & \vdots & \vdots \\ \boldsymbol{G}_{(n,1)}(\boldsymbol{x}) & \dots & \boldsymbol{G}_{(m,m)}(\boldsymbol{x}) \end{pmatrix} \tag{3.44}$$

Über Gl. (3.45) ergeben sich $n-m$ Gleichungen für $q_j(\boldsymbol{x})$. Hierzu muss die Matrix $\boldsymbol{Q}(\boldsymbol{x})$ invertiert werden.

$$q_j = \boldsymbol{f}_{(j,1)}(\boldsymbol{x}) - \boldsymbol{G}_{(j,1\dots m)}(\boldsymbol{x})\boldsymbol{Q}^{-1}\boldsymbol{f}_{(n-m+1\dots n,1)}(\boldsymbol{x}), j = 1, \dots, n-m \tag{3.45}$$

Die $n-m$ Gewichte ψ_j ergeben sich aus einem Koeffizientenvergleich bezüglich der Laplace-Variablen s aus Gleichung (3.46). Über z_j^d können $n-m$ Wunsch-Nullstellen[11] des modifizierten Systems mit Ausgang $\boldsymbol{h}_A$ gesetzt werden.

$$\det \begin{pmatrix} s\boldsymbol{I} - \boldsymbol{A} & -\boldsymbol{B} \\ \boldsymbol{c}_1 & \boldsymbol{0} \\ \vdots & \vdots \\ \boldsymbol{c}_{m-1} & \boldsymbol{0} \\ \boldsymbol{c}_{A_m} & \boldsymbol{0} \end{pmatrix} = \det \begin{pmatrix} -\boldsymbol{A} & -\boldsymbol{B} \\ \boldsymbol{c}_1 & \boldsymbol{0} \\ \vdots & \vdots \\ \boldsymbol{c}_{m-1} & \boldsymbol{0} \\ \boldsymbol{c}_m & \boldsymbol{0} \end{pmatrix} \prod_{j=1}^{n-m} \left(1 - \frac{s}{z_j^d}\right) \tag{3.46}$$

Um Gleichung (3.46) lösen zu können muss das System um den Arbeitspunkt (Zustand $\boldsymbol{x}_{ss}$, Eingang $\boldsymbol{u}_{ss}$) linearisiert werden, was Gleichung (3.47) beschreibt.

$$\boldsymbol{A} = \partial \boldsymbol{f}/\partial \boldsymbol{x}|_{\boldsymbol{x}_{ss},\boldsymbol{u}_{ss}}, \boldsymbol{B} = \boldsymbol{G}\left(\boldsymbol{x}_{ss}\right) \tag{3.47a}$$

$$\boldsymbol{c}_1 = \partial \boldsymbol{h}_a/\partial \boldsymbol{x}|_{\boldsymbol{x}_{ss},\boldsymbol{u}_{ss}}, \dots, \boldsymbol{c}_m = \partial \boldsymbol{h}_m/\partial \boldsymbol{x}|_{\boldsymbol{x}_{ss},\boldsymbol{u}_{ss}}, \boldsymbol{c}_{A_m} = \partial \boldsymbol{h}_{A_m}/\partial \boldsymbol{x}|_{\boldsymbol{x}_{ss},\boldsymbol{u}_{ss}} \tag{3.47b}$$

Um diese Gleichungen für mechanische Systeme lösen zu können müssen die Gleichungen des Systems unter Umständen umsortiert werden, da etwa in der Darstellung von $\dot{\boldsymbol{x}}$ nach Gl. (3.48) in der $\boldsymbol{Q}$ Matrix Nullzeilen existieren können, wodurch $\boldsymbol{Q}$ nicht invertierbar ist.

$$\dot{\boldsymbol{x}} = \begin{pmatrix} \dot{\varphi}_1 & \ddot{\varphi}_1 & \dots & \dot{\varphi}_{n/2} & \ddot{\varphi}_{n/2} \end{pmatrix}^T \tag{3.48}$$

Über eine Transformationsmatrix $\boldsymbol{T}$ kann die Zustandsreihenfolge umsortiert werden, so dass $\boldsymbol{Q}$ nicht mehr singulär ist. Auch die linearisierten Systeme für Gl. (3.47) müssen in diesem Fall über die Transformationsmatrix $\boldsymbol{T}$ transformiert werden (Gl. (3.49)).

$$\boldsymbol{x}_* = \boldsymbol{T}\boldsymbol{x}, \boldsymbol{A}_* = \boldsymbol{T}\boldsymbol{A}\boldsymbol{T}^{-1}, \boldsymbol{B}_* = \boldsymbol{T}\boldsymbol{B}, \boldsymbol{C}_* = \boldsymbol{C}\boldsymbol{T}^{-1} \tag{3.49}$$

[11] Der Index d aus dem Englischen steht für „desired“ in z_j^d.

Nach der Umsortierung der Zustände kann für die meisten mechanischen Systeme eine nichtsinguläre $\boldsymbol{Q}$ Matrix erzeugt werden.

Ist es schließlich möglich $\boldsymbol{Q}$ zu invertieren und das Gleichungssystem, welches sich aus einem Koeffizientenvergleich bezüglich s aus Gl. (3.47) ergibt, zu lösen kann der neue Ausgangs-Vektor $\boldsymbol{h}_A$ bestimmt werden.

Da $\boldsymbol{h}_A$ allerdings nur im stationären Fall mit dem ursprünglichen $\boldsymbol{h}_f$ übereinstimmt, treten bei Trajektorienfolgeaufgaben, wie sie in der Robotik üblich sind um den TCP auf einer Bahn zu führen, starke Abweichungen auf der Bahn auf, wenn das inverse System zur (Vor-) Steuerung bezüglich dem virtuellen Ausgang $\boldsymbol{h}_A$ erzeugt wird. Um dieses Problem abzumildern wird die Methode hier so erweitert, dass auch auf der Bahn, im nicht-stationären Fall, ein gutes Trajektorienfolgeverhalten erreicht wird.

Hierzu wird die vorgestellte Methode mit einem quasistatischen (QS) Hilfssystem über ein Parallelmodell nach Abschnitt 3.5.2 kombiniert. Das Prinzip der erweiterten Metho-

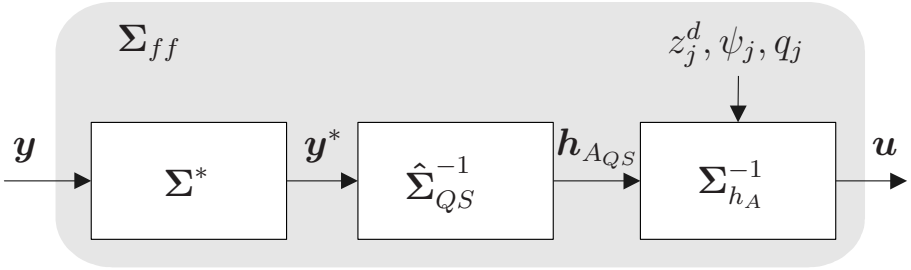


Abbildung 3.11: Prinzipskizze zur Kombination von Invertierung bezügliche einem virtuellem Ausgang $\boldsymbol{h}_A$ mit Kompensation der Verformungen über ein Parallelmodell zu einer Vorsteuerung $\boldsymbol{\Sigma}_{ff}$.

de zeigt Abb. 3.11. Zunächst wird der Soll-Vektor des Ausgangs $\boldsymbol{y}$ gefiltert, um die für die Invertierung benötigten Ableitungen von $\boldsymbol{y}$ zu ermöglichen. Der gefilterte Ausgang wird mit $\boldsymbol{y}^*$ bezeichnet. Mit Hilfe von $\boldsymbol{y}^*$ kann mit der in Abschnitt 3.5.2 vorgestellten Methode über ein Parallelmodell eine quasistatische Näherung der elastischen Verformungen berechnet werden. Über eine inverse Kinematik, welche für einfache Fälle exakt oder approximativ, über die im folgenden Abschnitt 3.6 vorgestellten Methoden, berechnet werden kann, können schließlich Korrekturterme für die direkt steuerbaren Gelenkwinkel berechnet werden, um den TCP oder die Last auf der gewünschten Trajektorie zu halten (siehe hierzu auch Abschnitt 3.8). Diese aus $\boldsymbol{y}^*$ quasistatisch berechneten kinematischen Größen können als approximierter Zustand $\boldsymbol{x}_{QS}$ in den virtuellen Ausgang $\boldsymbol{h}_A$ eingesetzt werden, wodurch sich eine quasistatische Approximation $\boldsymbol{h}_{A,QS}$ ergibt (Gl. (3.50)).

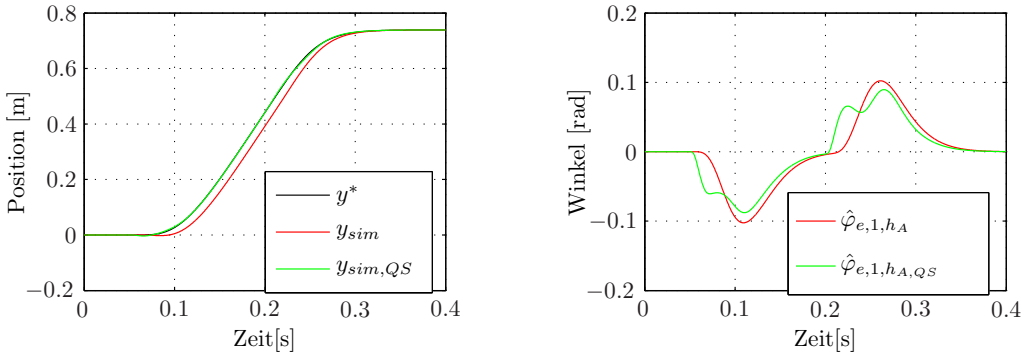
$$\boldsymbol{h}_{A,QS} = \boldsymbol{h}_A(\boldsymbol{x}_{QS}) \tag{3.50}$$

Das System welches $\boldsymbol{h}_{A,QS}$ aus $\boldsymbol{y}^*$ erzeugt wird in Abbildung 3.11 mit $\hat{\boldsymbol{\Sigma}}_{QS}^{-1}$ bezeichnet. $\boldsymbol{h}_{A,QS}$ dient dann als Eingang des inversen Systems bezüglich des virtuellen Ausgangs $\boldsymbol{h}_A$, in Abb. 3.11 mit $\boldsymbol{\Sigma}_{h_A}^{-1}$ bezeichnet[12]. Hieraus werden die Steuergrößen $\boldsymbol{u}$ berechnet,

[12] Auch wenn es sich hierbei um nichtlineare Systeme handelt wird, zwecks der besseren Lesbarkeit, eine Syntax mit $\boldsymbol{\Sigma}$ in Anlehnung an lineare Übertragungsglieder G(s) gewählt. Diese sind in diesem Fall als mehrdimensionale Funktionen bzw. Operatoren der Eingänge zu deuten.

welche als (Vorsteuer-) Eingang der Strecke dienen. Die Invertierung der Systemgleichungen in $\mathbf{\Sigma}_{h_A}^{-1}$ kann ohne Approximation durchgeführt werden, da durch die Wahl Wunsch-Nullstellen z_j^d die instabile Nulldynamik vermieden werden kann.

Um das Verfahren zu demonstrieren wird es auf das Beispielsystem aus Abschnitt 3.5.2 angewendet (siehe Abb. 3.8). Für die Simulation wurden die Parameter $l = 2m, \xi = 0.2, m = 100kg, c = 1e5\frac{N \cdot m}{rad}, d = 15\frac{N \cdot m \cdot s}{rad}, J_z = 10kg \cdot m^2, x_l = 0.5m$ gewählt. Für $x_l = 0.5m$ ergeben sich die die Nullstelle $z_1 = -292,1857, z_2 = 305,5786$ für das linearisierte System. Es existiert daher eine Nullstelle mit positivem Realteil (siehe hierzu auch Abb. 3.9), das System ist also nicht exakt bezüglich dem gewünschten Ausgang invertierbar. Mit den in diesem Abschnitt 3.5.3 vorgestellten Gleichungen lässt sich für



(a) Vergleich der Ausgänge y.

(b) Vergleich der Winkel $\hat{\varphi}_e$.

Abbildung 3.12: Vergleich der Simulationen in welchen die Vorsteuerung $\mathbf{\Sigma}_{ff}$ bezüglich des virtuellen Ausgangs $\boldsymbol{h}_A$ entworfen ist. In rot dargestellt Ergebnisse ohne quasistatische Korrektur, in grün mit Korrektur (Index QS). Dargestellt sind jeweils die Simulationsergebnisse der vorgesteuerten Strecke, unter der Annahme eines perfekten Lagereglers ($\varphi_{a,1}$ wird vorgegeben).

das System ein virtueller Ausgang berechnen. Mit der Wahl der Wunsch-Nullstellen des Systems mit virtuellem Ausgang zu $z_1^d = z_2^d = -292,1857, z_3^d = -305,5786$ ergibt sich ein bezüglich h_A exakt invertierbares System.

Setzt man nun bei der Invertierung $h_A \stackrel{!}{=} y^*$ so kommt es entlang der Trajektorie zu Abweichung, da h_A nur stationär mit h übereinstimmt. Dieses lediglich asymptotische Folgeverhalten des Ausgangs der Strecke y_{sim} zeigt Abb. 3.12(a). In der Simulation wurde ein perfekter Lageregler angenommen, es wird daher direkt $\varphi_{a,e,1}$, welches vom inversen System berechnet wird, auf ein Streckenmodell aufgeschalten. Die Parameter des Streckenmodells und des inversen Modells sind dabei als identisch angenommen und Störungen wurden vernachlässigt. Wird eine Berechnung der Verformungen über ein Parallelmodell (nach Abschnitt 3.5.2) vorgeschaltet (siehe Abb. 3.11) und im inversen Modell $h_A \stackrel{!}{=} h_{A,QS}$ gesetzt so ergibt sich ein deutlich besseres Folgeverhalten der Strecke, was an $y_{sim,QS}$ zu sehen ist. Solltrajektorie y^* und $y_{sim,QS}$ sind nahezu identisch. Durch das Vorschalten der Korrektur ergeben sich für die Winkel $\hat{\varphi}_{e,1}$ des elastischen Elements unterschiedliche Verläufe in der gesteuerten Strecke (Abb. 3.12(b)).

Nachteilig an diesem Verfahren ist allerdings der große Berechnungsaufwand für komplexere Systeme. So ist $\boldsymbol{Q}$ für viele komplexere Systeme nicht mehr symbolisch invertierbar, was Voraussetzung ist um Gl. (3.45) lösen zu können. Auch das Lösen des

Gleichungssystems, welches sich aus Gl. (3.46) ergibt ist problematisch. Für ein stark nichtlineares System, wie ein komplexes Robotermodell, genügt es zudem nicht die Linearisierung bezüglich eines Arbeitspunktes $(\boldsymbol{x}_{ss}, \boldsymbol{u}_{ss})$ durchzuführen, sondern der Arbeitspunkt muss ständig neu mitgezogen werden, wodurch die Parameter ψ_j vom Arbeitspunkt abhängig werden.

3.6 Inverse Kinematik des Roboters

Um die Last oder das Werkzeug auf einer vorgegebene Trajektorie $\mathfrak{T}_L$ führen zu können, müssen die entsprechenden Gelenkwinkel φ_i eines Roboters berechnet werden. Für jeden einzelnen Zeitpunkt bedeutet dies, dass die Funktion $\boldsymbol{f}_k$, welche den kinematischen Zusammenhang zwischen gewünschter Sollstellung $\mathbf{r}_L$ und den dazugehörigen Gelenkwinkeln beschreibt, invertiert werden muss (Gl. (3.51)).

$$\mathbf{r}_L(t) = \boldsymbol{f}_k(\boldsymbol{\varphi}(t)) \tag{3.51a}$$
$$\boldsymbol{\varphi}(t) = \boldsymbol{f}_k^{-1}(\mathbf{r}_L(t)) \tag{3.51b}$$

Problematisch hieran ist, dass im Allgemeinen mehrere Lösungen für $\boldsymbol{f}_k^{-1}$ existieren. Ein weiteres Problem ist, dass in sog. singulären Stellungen des Roboters unendlich schnelle Winkeländerungen $\dot{\boldsymbol{\varphi}}$ nötig sind, um exakt einer Trajektorie $\mathfrak{T}_L$ folgen zu können, welche durch eine Singularität führt. In heutigen Robotersteuerungen, wie auch denen des KR16 und KR210, sind Algorithmen implementiert, welche diese Probleme für den starren Roboter – zumindest Näherungsweise – lösen, bzw. umgehen.

Für einen starren Roboter, der kinematisch wie der KR16 oder KR210 seriell aufgebaut ist, kann eine einfache analytische Lösung für $\boldsymbol{f}_k^{-1}$ gefunden werden. Hierzu wird die inverse Kinematik in zwei Teilprobleme aufgespalten (kinematische Entkopplung). Die drei Grundachsen des Roboters werden dabei dazu verwendet den Handwurzelpunkt[13] (HWP) so zu positionieren, dass Gl. (3.52) erfüllt werden.

$$\mathbf{r}_{HWP,t} = \mathbf{r}_{L,t} - d_6 \mathbf{R} \cdot \mathbf{e}_6 \tag{3.52a}$$
$$\mathbf{R}_L = {}^3_0\mathbf{R} \cdot {}^6_3\mathbf{R} \tag{3.52b}$$
$${}^6_3\mathbf{R} = {}^3_0\mathbf{R}^T \cdot \mathbf{R}_L \tag{3.52c}$$

Hierin ist $\mathbf{r}_{HWP,t}$ die Position des Handwurzelpunktes, $\mathbf{r}_{L,t}$ der gewünschte Ort für die Last des Roboters unter der Orientierung $\mathbf{R}_L$ und ${}^3_0\mathbf{R}$ bzw. ${}^6_3\mathbf{R}$ die Rotationsmatrizen der jeweiligen Achsen des Roboters analog zu Kapitel 2.6. d_6 ist die Länge vom Handwurzelpunkt zum Lastmittelpunkt in Richtung $\mathbf{e}_6$ nach Denavit-Hartenberg Konvention (für weitere Details siehe [14]).

Diese Entkopplung setzt allerdings perfekt zueinander senkrecht, bzw. parallel stehende Achsen voraus, wie sie beim starren Roboter gegeben sind. Sobald die Achsen, wie beim strukturelastischen Roboter der Fall, zueinander verkippt stehen (können), wird das Problem deutlich schwerer zu lösen. Es müssen meist Näherungsalgorithmen verwendet werden. Das Problem der Mehrdeutigkeit kann dadurch gelöst werden, dass eine bevorzugte Achskonfiguration eingestellt wird welche, im Fall von mehreren Lösungen,

[13] Der Handwurzelpunkt des Roboters ist der Punkt bei Achse 4 in dem sich die Rotationsachsen der Gelenke 4, 5 und 6 schneiden.

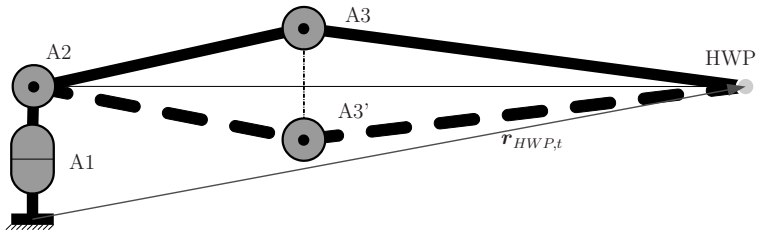


Abbildung 3.13: Beispiel für die Mehrdeutigkeit der inversen Kinematik eines Roboters: Zu einem bestimmten Handwurzelpunkt $\mathbf{r}_{HWP,t}$ - bis auf die Strecklage des Roboters - gibt es immer zwei Lösungen (für A2 und A3), die trapezförmig um die Gerade zwischen Achse 2 und Handwurzelpunkt liegen.

gewählt wird. Eine klassische Mehrdeutigkeit für die Grundachsen ist etwa, dass es zu einem bestimmten Handwurzelpunkt $\mathbf{r}_{HWP,t}$ - bis auf die Strecklage des Roboters - immer zwei Lösungen gibt, die trapezförmig um die Gerade zwischen Achse 2 und Handwurzelpunkt liegen (Abb. 3.13). Beim Programmieren der Bahn wird angegeben welche der beiden Lösungen gewählt werden soll. Ähnliche Verfahren werden auch bei anderen mehrdeutigen Lösungen für $\boldsymbol{f}_k^{-1}$ gewählt.

3.6.1 Direkte Lösung der inversen Kinematik

Die direkte Lösung für die inverse Kinematik des elastisch verformten Roboters ist nur extrem eingeschränkt möglich. Im Fall eines starren Roboters können die Gelenkwinkel der Grundachsen sehr leicht analytisch direkt berechnet werden. Da Achse 1 senkrecht und Achse 2 und Achse 3 parallel im Raum stehen, können die Achswinkel, welche zum Handwurzelpunkt $\mathbf{r}_{HWP,t}$ führen in zwei Teilprobleme aufgeteilt werden. Zunächst wird der benötigte Winkel der Achse 1 berechnet. Anschließend ergibt sich ein ebenes geometrisches Problem für Achse 2 und 3.

$$\mathbf{r}_{HWP,t} = \begin{pmatrix} x_{HWP} \\ y_{HWP} \\ z_{HWP} \end{pmatrix} \tag{3.53a}$$

$$u_{HWP} = \operatorname{sgn}(x_{HWP})\sqrt{x_{HWP}^2 + y_{HWP}^2} \tag{3.53b}$$

$$D = \frac{u_{HWP}^2 + z_{HWP}^2 - l_1^2 - l_2^2}{2 l_1 l_2} \tag{3.53c}$$

$$\varphi_{a,1,r} = -\operatorname{atan2}\left(\frac{y_{HWP}}{x_{HWP}}\right) \tag{3.53d}$$

$$\varphi_{a,2,r} = -\left(\operatorname{atan2}\left(\frac{z_{HWP}}{u_{HWP}}\right) - \operatorname{atan2}\left(\frac{l_2 \sin(-\varphi_{a,3,r})}{l_1 + l_2 \cos(-\varphi_{a,3,r})}\right)\right) \tag{3.53e}$$

$$\varphi_{a,3,r} = \operatorname{atan2}\left(\frac{\pm\sqrt{1 - D^2}}{D}\right) \tag{3.53f}$$

Gleichung (3.53) stellt dies dar, die Lösung ist jedoch nicht eindeutig (Term $\pm\sqrt{1 - D^2}$ in Gl. (3.53)). In Gl. (3.53) bezeichnet l_1 die Länge der Schwinge zwischen Achse 2 und Achse 3 und l_2 die Länge des Armes zwischen Achse 3 und Achse 4 und $\operatorname{sgn}(x_{HWP})$ steht für das Vorzeichen von x_{HWP}. u_{HWP} und D sind Hilfsvariablen, um das Ganze

kompakter schreiben zu können. Vernachlässigt wurde hierbei, dass die Achsen des Roboters in der Realität nicht exakt in einer Ebene liegen und ein Versatz zwischen A1 und A2 existiert. Dies ist zusätzlich zu berücksichtigen. Die Winkel und Koordinaten richten sich nach der in dieser Arbeit verwendeten Konvention (siehe A.6), wobei $\mathbf{r}_{HWP,t}$ bezüglich des Inertial-Koordinatensystems angegeben ist.

Einfach Zusammenhänge wie in Gleichung (3.53) können für einen elastisch verformten Roboter nicht mehr gefunden werden. Sind die Verformungen der Struktur bekannt – für die Approximation der Struktur durch elastische Gelenke bedeutet dies $\boldsymbol{\varphi}_e$ sind bekannt – müssen die entsprechende Transformationsmatrizen $\mathbf{T}$ für die verformte Struktur aufgestellt werden. Durch die Invertierung können dann die Gelenkwinkel $\boldsymbol{\varphi}_a$ der Achs-Gelenke berechnet werden. Für Systeme mit wenigen elastischen Freiheitsgraden ist dies teilweise noch analytisch möglich, allerdings werden hier schnell Grenzen erreicht, da sehr große verschachtelte analytische Ausdrücke entstehen.

Ein Ansatz, um dieses Problem zu vereinfachen ist es, zunächst die Lösung für die Starrkörperkinematik zu berechnen und dann für die inverse Kinematik des elastisch verformten Roboters entweder für Achse 2 oder Achse 3 die Lösung des Starrkörper-Problems zu wählen und dann die inverse Kinematik nur noch bezüglich der zwei verbleibenden Freiheitsgrade zu lösen. Bei relativ starren Systemen können hierdurch gute Näherungslösungen gefunden werden. Allerdings kann für den Fall einer - aufgrund der Verformungen - knapp nicht erreichbare Sollposition keine Lösung gefunden werden (weiter Details hierzu im Folgenden).

Alternativ ist die Lösung des inversen kinematischen Problems über eine Linearisierung und iterativen Lösungsansatz möglich.

Hierzu wird die Jacobi-Matrix $\boldsymbol{J}_f$ der Roboterkinematik verwendet (siehe hierzu auch [59, 60, 61]).

Wie bereits beschrieben, lässt sich die Position und Orientierung der Last über eine homogene Matrix ${}^{L}_{0}\mathbf{T}$ beschreiben. In dieser Matrix beschreiben die ersten drei Zeilen der vierten Spalte die Position bezüglich des inertialen Koordinatensystems und die ersten drei Zeilen der ersten drei Spalten die Orientierung des Last-Koordinatensystems bezüglich des inertialen Koordinatensystems Gl. (2.25).

Soll in der Jacobimatrix $\boldsymbol{J}_{f,t}$ lediglich die translatorische Position berücksichtigt werden, so kann die Jacobimatrix bezüglich der ersten drei Achsen des Roboters aufgestellt werden (Gl. (3.54)).

$$ {}^{L}_{0}\mathbf{T}(\boldsymbol{\varphi}_e, \boldsymbol{\varphi}_{a,r}) = \left(\begin{array}{c|c} {}^{L}_{0}\mathbf{R} & {}^{L}_{0}\mathbf{r}_t \\ \hline \mathbf{0} & 1 \end{array} \right) \tag{3.54a} $$

$$ {}^{L}_{0}\mathbf{r}_t = \begin{pmatrix} {}^{L}_{0}r_{t,x} & {}^{L}_{0}r_{t,y} & {}^{L}_{0}r_{t,z} \end{pmatrix}^T \tag{3.54b} $$

$$ \boldsymbol{J}_{f,t} = \begin{pmatrix} \frac{\partial {}^{L}_{0}r_{t,x}}{\partial \varphi_{a,1,r}} & \frac{\partial {}^{L}_{0}r_{t,x}}{\partial \varphi_{a,2,r}} & \frac{\partial {}^{L}_{0}r_{t,x}}{\partial \varphi_{a,3,r}} \\ \frac{\partial {}^{L}_{0}r_{t,y}}{\partial \varphi_{a,1,r}} & \frac{\partial {}^{L}_{0}r_{t,y}}{\partial \varphi_{a,2,r}} & \frac{\partial {}^{L}_{0}r_{t,y}}{\partial \varphi_{a,3,r}} \\ \frac{\partial {}^{L}_{0}r_{t,z}}{\partial \varphi_{a,1,r}} & \frac{\partial {}^{L}_{0}r_{t,z}}{\partial \varphi_{a,2,r}} & \frac{\partial {}^{L}_{0}r_{t,z}}{\partial \varphi_{a,3,r}} \end{pmatrix} \tag{3.54c} $$

Soll in der Jacobimatrix $\boldsymbol{J}_f$ auch die Orientierung mit berücksichtigt werden, so ist es sinnvoll (siehe [60]), die Jacobimatrix $\boldsymbol{J}_f$ so zu erweitern, dass der schiefsymmetrische

Anteil der Orientierung ${}^{L}_{0}\mathbf{R}_S$ von ${}^{L}_{0}\mathbf{R}$ verwendet wird. Dieser ergibt sich nach Gl. (3.55).

$$ {}^{L}_{0}\mathbf{R}_S = {}^{L}_{0}\mathbf{R} - {}^{L}_{0}\mathbf{R}^T = \begin{pmatrix} 0 & {}^{L}_{0}R_{S,1} & {}^{L}_{0}R_{S,2} \\ -{}^{L}_{0}R_{S,1} & 0 & {}^{L}_{0}R_{S,3} \\ -{}^{L}_{0}R_{S,2} & -{}^{L}_{0}R_{S,3} & 0 \end{pmatrix} \tag{3.55} $$

Mit diesem schiefsymmetrischen Anteil kann die Jacobimatrix dann in der Form von Gl. (3.56) bezüglich der sechs Achswinkel des Roboters aufgestellt werden.

$$ \boldsymbol{J}_f = \begin{pmatrix} \frac{\partial {}^{L}_{0}r_{t,x}}{\partial \varphi_{a,1,r}} & \frac{\partial {}^{L}_{0}r_{t,x}}{\partial \varphi_{a,2,r}} & \frac{\partial {}^{L}_{0}r_{t,x}}{\partial \varphi_{a,3,r}} & \frac{\partial {}^{L}_{0}r_{t,x}}{\partial \varphi_{a,4,r}} & \frac{\partial {}^{L}_{0}r_{t,x}}{\partial \varphi_{a,5,r}} & \frac{\partial {}^{L}_{0}r_{t,x}}{\partial \varphi_{a,6,r}} \\ \frac{\partial {}^{L}_{0}r_{t,y}}{\partial \varphi_{a,1,r}} & \frac{\partial {}^{L}_{0}r_{t,y}}{\partial \varphi_{a,2,r}} & \frac{\partial {}^{L}_{0}r_{t,y}}{\partial \varphi_{a,3,r}} & \frac{\partial {}^{L}_{0}r_{2}y}{\partial \varphi_{a,4,r}} & \frac{\partial {}^{L}_{0}r_{t,y}}{\partial \varphi_{a,5,r}} & \frac{\partial {}^{L}_{0}r_{t,y}}{\partial \varphi_{a,6,r}} \\ \frac{\partial {}^{L}_{0}r_{t,z}}{\partial \varphi_{a,1,r}} & \frac{\partial {}^{L}_{0}r_{t,z}}{\partial \varphi_{a,2,r}} & \frac{\partial {}^{L}_{0}r_{t,z}}{\partial \varphi_{a,3,r}} & \frac{\partial {}^{L}_{0}r_{3}y}{\partial \varphi_{a,4,r}} & \frac{\partial {}^{L}_{0}r_{t,z}}{\partial \varphi_{a,5,r}} & \frac{\partial {}^{L}_{0}r_{3}y}{\partial \varphi_{a,6,r}} \\ \frac{\partial {}^{L}_{0}R_{S,1}}{\partial \varphi_{a,1,r}} & \frac{\partial {}^{L}_{0}R_{S,1}}{\partial \varphi_{a,2,r}} & \frac{\partial {}^{L}_{0}R_{S,1}}{\partial \varphi_{a,3,r}} & \frac{\partial {}^{L}_{0}R_{S,1}}{\partial \varphi_{a,4,r}} & \frac{\partial {}^{L}_{0}R_{S,1}}{\partial \varphi_{a,5,r}} & \frac{\partial {}^{L}_{0}R_{S,1}}{\partial \varphi_{a,6,r}} \\ \frac{\partial {}^{L}_{0}R_{S,2}}{\partial \varphi_{a,1,r}} & \frac{\partial {}^{L}_{0}R_{S,2}}{\partial \varphi_{a,2,r}} & \frac{\partial {}^{L}_{0}R_{S,2}}{\partial \varphi_{a,3,r}} & \frac{\partial {}^{L}_{0}R_{S,2}}{\partial \varphi_{a,4,r}} & \frac{\partial {}^{L}_{0}R_{S,2}}{\partial \varphi_{a,5,r}} & \frac{\partial {}^{L}_{0}R_{S,2}}{\partial \varphi_{a,6,r}} \\ \frac{\partial {}^{L}_{0}R_{S,3}}{\partial \varphi_{a,1,r}} & \frac{\partial {}^{L}_{0}R_{S,3}}{\partial \varphi_{a,2,r}} & \frac{\partial {}^{L}_{0}R_{S,3}}{\partial \varphi_{a,3,r}} & \frac{\partial {}^{L}_{0}R_{S,3}}{\partial \varphi_{a,4,r}} & \frac{\partial {}^{L}_{0}R_{S,3}}{\partial \varphi_{a,5,r}} & \frac{\partial {}^{L}_{0}R_{S,3}}{\partial \varphi_{a,6,r}} \end{pmatrix} \tag{3.56} $$

Die Jacobimatrix kann nun dazu eingesetzt werden um die kartesische Abweichung von der Sollstellung (Position und/oder Orientierung) zu korrigieren.

Hierzu werden Korrekturwinkel $\Delta\boldsymbol{\varphi}_a$ berechnet, die sich aus der kartesischen Abweichung $\Delta\boldsymbol{r}$ von der Sollstellung – welche sich durch ${}^{L}_{0}\mathbf{T}$ beschreiben lässt – ergeben. $\Delta\boldsymbol{r}$ kann dabei, je nach Anwendungsfall und zulässiger Rechenzeit, nur die kartesische Abweichung oder auch die schiefsymmetrische Terme der Abweichung der Orientierung von der Sollstellung beinhalten.

Da die Verformungen beim Roboter klein sind bietet es sich an, als Startlösung die exakte Lösung der starren Kinematik zu wählen, so dass bereits nach wenigen Iterationsschritten (je nach dem gewählten Zeitintervall $\Delta t = \frac{1}{f_s}$ genügt oft auch schon ein Schritt), über die Korrekturwinkel $\Delta\boldsymbol{\varphi}$ eine hohe Genauigkeit erreicht werden kann.

$$ \Delta\mathbf{r} = \begin{pmatrix} {}^{L}_{0}r_{t,x,soll} - {}^{L}_{0}r_{t,x} \\ {}^{L}_{0}r_{t,y,soll} - {}^{L}_{0}r_{t,y} \\ {}^{L}_{0}r_{t,z,soll} - {}^{L}_{0}r_{t,z} \\ {}^{L}_{0}R_{S,1,soll} - {}^{L}_{0}R_{S,1} \\ {}^{L}_{0}R_{S,2,soll} - {}^{L}_{0}R_{S,2} \\ {}^{L}_{0}R_{S,3,soll} - {}^{L}_{0}R_{S,3} \end{pmatrix} \tag{3.57a} $$

$$ \Delta\mathbf{r} = \boldsymbol{J}_f(\boldsymbol{\varphi}_e, \boldsymbol{\varphi}_{a,r})\Delta\boldsymbol{\varphi}_a \tag{3.57b} $$

$$ \Delta\boldsymbol{\varphi}_a = \boldsymbol{J}_f^{-1}(\boldsymbol{\varphi}_e, \boldsymbol{\varphi}_{a,r})\Delta\mathbf{r} \tag{3.57c} $$

Ein großer Nachteil von Ansätzen für eine exakte Lösung ist, dass die vorgegebenen Trajektorien $\mathfrak{T}_L$ meist anhand eines starren kinematischen Modells geplant werden. Hierdurch sind einige Punkte eventuell durch die elastisch verformte Kinematik nicht mehr exakt erreichbar, es existiert daher keine Lösung für ${}^{L}_{0}\mathbf{T}^{-1}$. Dies ist insbesondere in der Strecklage und in der Kerzenstellung entscheidend. In der Strecklage (A1 bis A6 0°) besitzt der Roboter bei einer Verformung der Schwinge oder des Arms, nur noch eine geringere Reichweite als ein starrer unverformter Roboter. In der Kerzenstellung (A1 0°,A2 −90°, sowie A3 bis A6 0°) sowie anderen Stellungen bei denen der Handwurzelpunkt auf der Linie senkrecht über dem Mittelpunkt von Achse 1 liegt (also auf

der Rotationsachse von Achse 1), kann es ebenfalls keine exakte Lösung für die inverse Kinematik geben, wenn die Struktur des Roboters so verformt ist, dass die Rotationsachsen von Achse 2 und Achse 3 nicht mehr exakt orthogonal zu Achse 1 verlaufen. Für diese Fälle fehlt dem Roboter ein weiteres Gelenk, mit welchem diese Verformung ausgeglichen werden könnte (wenn gleichzeitig eine vorgegebene Orientierung des TCPs gefordert wird, für welche die drei Handachsen des Roboters benötigt werden).

Im Falle einer Unerreichbarkeit, bzw. von singulären Stellungen, versagt auch der Lösungsansatz nach Gl. (3.57), da $\boldsymbol{J}_f^{-1}$ in diesem Fall degeneriert und es zu einem Rangabfall kommt. Ein Ansatz um dies zu umgehen ist in diesem Fall die Schrittweite für $\Delta\boldsymbol{\varphi}_a$ zu begrenzen.

$$|\Delta\varphi_{a,k}| \leq \Delta\varphi_{a,max,k} = \left| J_k^{-1}(\boldsymbol{\varphi}_e, \boldsymbol{\varphi}_{a,r}) \frac{\Delta r_k}{c_{lim,k}} \right|, \ \forall\ k = 1 \ldots 6 \tag{3.58}$$

Gleichung (3.58) beschreibt diesen Ansatz, wobei k für die jeweilige k-te Zeile, bzw. Komponente steht. $c_{lim,k}$ muss jeweils so berechnet werden, dass in jedem Zeitschritt der maximale Winkel $\Delta\varphi_{a,max,k}$ für jedes Gelenk eingehalten wird. Durch die Begrenzung der Schritte wird der Roboter aus der Singularität herausgeführt. Für weitere Details zu diesem Ansatz siehe [62].

Weitere Verfahren wie mit Singularitäten und nicht erreichbaren Punkten umgegangen werden können finden sich in Abschnitt 3.6.3.

3.6.2 Geometrische Approximation der inversen Kinematik

Ein weiterer Ansatz zur Approximation der inversen Kinematik des elastischen Roboters baut auf einfachen geometrischen Überlegungen auf.

Da die Verformungen der elastischen Elemente des Roboters klein sind, reichen meist kleine Korrekturen der Lösung für die inverse Kinematik des starren Roboters.

Um dies zu erreichen wird ein dreistufiger Ansatz verwendet, welcher den großen Vorteil besitzt, dass die Jacobimatrix $\boldsymbol{J}_f(\boldsymbol{\varphi}_e, \boldsymbol{\varphi}_{a,r})$ des Roboters nicht invertiert werden muss und nur die Vorwärtskinematik des Roboters berechnet werden muss. Die Handachsen des Roboters werden wieder als starr angenommen, so dass Verformungen des Roboters lediglich aus Verformungen der Schwinge und des Armes des Roboters, sowie durch Kippsteifigkeiten der Gelenke verursacht werden.

Der Algorithmus besteht dabei aus drei Schritten, wobei wieder vorausgesetzt wird, dass die elastischen Verformungen des Systems bereits berechnet oder approximiert wurden. Die lokalen Koordinatensysteme für die einzelnen Strukturelemente sind jeweils im Ursprung der Gelenke gelegen und so orientiert - im Sinne der Denavit-Hartenberg Konvention - dass die Rotationsachse der Gelenke der z-Achse entspricht und die x-Achse in Richtung der Struktur (z. B. Arm, Schwinge) zeigt, die y-Achse ergibt sich dann entsprechend, so dass ein Rechtssystem erhalten wird:

1. Berechnung der Vorwärtskinematik ${}^{3}_{0}\mathbf{T}^{(e)}$ des verformten Roboters bis zur dritten Achse. Mit Hilfe der Transformation erfolgt die Berechnung der Länge des Hebelarms der verformten Schwinge ${}^{A3}_{A2}l_x^{(e2)}$ in x-Richtung im lokalen Koordinatensystem $e2$. Zudem wird die Abweichung bei Achse 3 in lokaler y-Richtung des

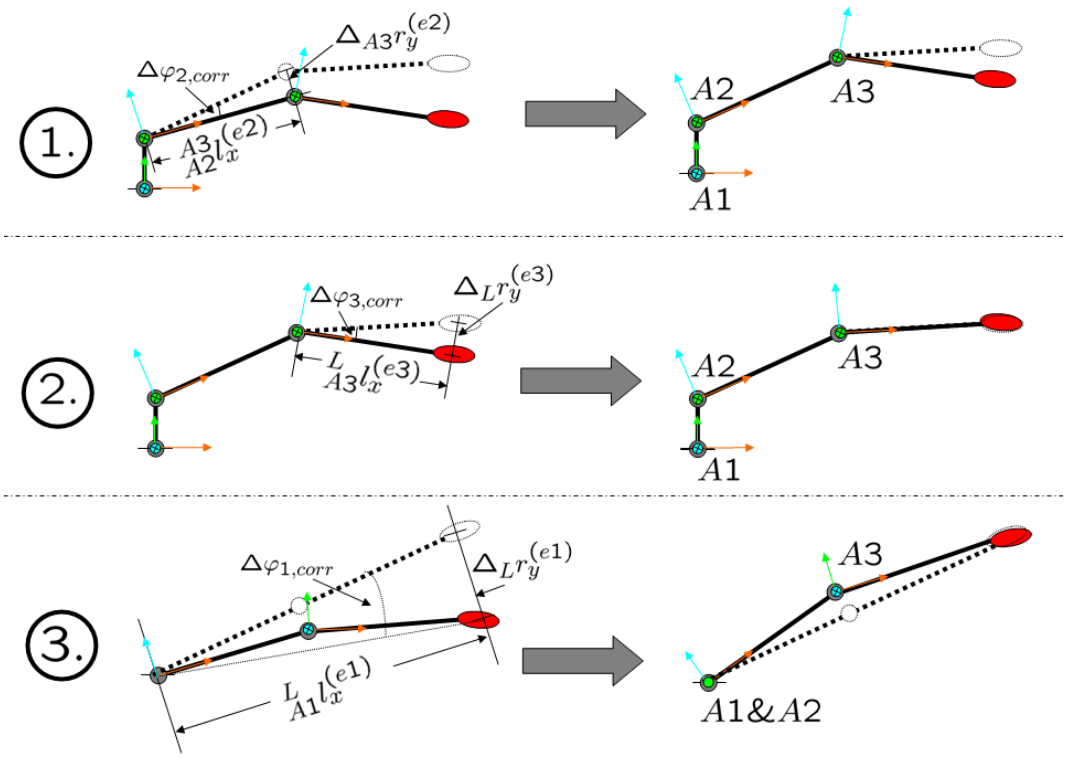


Abbildung 3.14: Ablauf des Algorithmus zur geometrische Approximation der inversen Kinematik. Schritt 1 und 2 zeigen den – stark vereinfacht dargestellten – Roboter von einer Seitenansicht, Schritt 3 in eine Ansicht von Oben. Gestrichelt dargestellt ist die Soll-Stellung des Roboters, durchgezogen der verformte Roboter.

verformten Systems von der Lösung der starren Kinematik $\Delta_{A3}r_y^{(e2)}$ berechnet. Über den geometrischen Zusammenhang $\tan\Delta\varphi_{2,corr} = \frac{\Delta_{A3}r_y^{(e2)}}{{}_{A2}^{A3}l_x^{(e2)}}$ kann dann ein Korrekturwinkel für Achse 2 berechnet werden, welcher zur Lösung der inversen Kinematik des starren Systems addiert wird.

2. Als Nächstes erfolgt die Berechnung der Vorwärtskinematik des verformten Systems bis zum TCP des Roboters ${}_0^L\mathbf{T}^{(e)}$, wobei für den Winkel von Achse 2 hierbei $\Delta\varphi_{2,corr}$ zur Lösung für $\varphi_{a,2,r}$ aus der inversen Kinematik des starren Systems addiert wird. Nun wird der Hebelarm ${}_{A3}^{L}l_x^{(e3)}$ von Achse 3 bis zum TCP des verformten Systems im lokalen Koordinatensystem $e3$ von Achse 3 berechnet sowie die Abweichung $\Delta_L r_y^{(e3)}$. Der Korrekturwinkel für A3 ergibt sich dann aus $\tan\Delta\varphi_{3,corr} = \frac{\Delta_L r_y^{(e3)}}{{}_{A3}^{L}l_x^{(e3)}}$.

3. Analog erfolgt die Berechnung des Korrekturwinkels für Achse 1, wobei hierfür nicht unbedingt erneute die Vorwärtskinematik ${}_0^3\mathbf{T}^{(e)}$ berechnet werden muss, da der Hebelarm sich bezüglich Achse 1 nur minimal ändert. Analog zu Schritt 1 und 2 ergibt sich $\tan\Delta\varphi_{1,corr} = \frac{\Delta_L r_y^{(e1)}}{{}_{A1}^{L}l_x^{(e1)}}$.

Der Algorithmus korrigiert lediglich Abweichungen in den drei Raumrichtungen. Kleine Änderungen der Orientierung werden dabei vernachlässigt. Abbildung 3.14 verdeutlicht

den Ablauf des Algorithmus. Der große Vorteil dieser Methode liegt darin, dass keine Invertierung benötigt wird und sie sehr robust funktioniert.

Sie setzt allerdings eine Lösung für die inverse Starrkörperkinematik voraus, die im Allgemeinen aber leicht zu erreichen ist. Zudem sollten die Verformungen der Elemente nicht zu groß sein, damit dieses Verfahren sinnvoll angewendet werden kann. Ist ein Punkt, aufgrund von Verformungen der Struktur nicht mehr exakt erreichbar (siehe hierzu auch Abschnitt 3.6.1), führt auch diese Methode auf sehr große Stellgrößen, bzw. Korrekturwinkel $\Delta\varphi_{i,corr}$, so dass diese mit einer zusätzlichen Funktion beschränkt werden müssen. Dieser Fall tritt immer ein, wenn die Länge der entsprechenden Hebelarme gegen Null geht. Eine Lösung, um dieses Problem abzumildern ist die Einführung einer inversen Gaußfunktion zur Skalierung des Korrekturwinkels in der Nähe eines unerreichbaren Punktes (Gl. (3.59)).

$$\Delta\varphi^*_{i,corr} = \Delta\varphi_{i,corr}\left(1 - e^{-s_\varphi\left(l_x^{(ei)}\right)^2}\right) \tag{3.59}$$

Hierdurch wird der Korrekturwinkel $\Delta\varphi^*_{i,corr}$ begrenzt, bzw. auf Null gesetzt, wenn der entsprechende Hebelarm $l_x^{(ei)}$ (im jeweiligen lokalen Koordinatensystem) gegen Null geht. Der Parameter $s_\varphi \gg 1$ dient zum Einstellen der Steilheit des Abfalls der Exponentialfunktion.

3.6.3 Lösung der inversen Kinematik über Linearisierung

Eine weitere Methode zur approx. Lösung der inversen Kinematik über eine Linearisierung (Jacobimatrix) ist die „Damped Least Square“ (DLS) Methode. Dieser Algorithmus wurde erstmals in [63] und [64] vorgeschlagen. Er besitzt den großen Vorteil, dass auch für singuläre Stellungen oder (knapp) nicht erreichbare Punkte Näherungslösungen gefunden werden.

Die DLS-Methode basiert auf der Pseudoinversen $\boldsymbol{J}_f^+$(Gl. (3.60), auch mit Moore-Penrose-Inverse bezeichnet) der Jacobimatrix nach Gl. (3.56).

$$\boldsymbol{J}_f^+ = \boldsymbol{J}_f^T\left(\boldsymbol{J}_f\boldsymbol{J}_f^T\right)^{-1} \tag{3.60}$$

Lösungen für die inverse Kinematik, welche auf der Pseudoinverse $\boldsymbol{J}_f^+$ aufbauen, besitzen allerdings ähnliche Nachteile wie Lösungen, die durch die direkte Invertierung von $\boldsymbol{J}_f$ entstehen. Befindet sich der Roboter in einer singulären Stellung oder ist der Zielpunkt unerreichbar kommt es auch bei dieser Methode zu einem Rangabfall und extrem große Stellgrößen werden berechnet (siehe auch Abschnitt 3.6.1). Im Algorithmus für die DLS-Inverse werden daher große Stellgrößen durch einen Dämpfungsterm vermieden.

Hierzu wird der Term nach Gl. (3.61) minimiert, wobei $\lambda \in \mathbb{R}^+$ eine Dämpfungskonstante größer Null darstellt.

$$\left\{\|\boldsymbol{J}_f\Delta\boldsymbol{\varphi}_a - \Delta\mathbf{r}\|^2 + \lambda^2\|\Delta\boldsymbol{\varphi}_a\|^2\right\} \rightarrow min \tag{3.61}$$

Dies ist Äquivalent zur Minimierung von Term (3.62).

$$\left\|\begin{pmatrix}\boldsymbol{J}_f\\ \lambda\mathbf{I}\end{pmatrix}\Delta\boldsymbol{\varphi}_a - \begin{pmatrix}\Delta\mathbf{r}\\ \mathbf{0}\end{pmatrix}\right\| \rightarrow min \tag{3.62}$$

Gl. (3.62) kann in Gl. (3.63) umgeformt werden und als Normalgleichung geschrieben werden.

$$\begin{pmatrix} \boldsymbol{J}_f \\ \lambda\mathbf{I} \end{pmatrix}^T \begin{pmatrix} \boldsymbol{J}_f \\ \lambda\mathbf{I} \end{pmatrix} \Delta\boldsymbol{\varphi}_a = \begin{pmatrix} \boldsymbol{J}_f \\ \lambda\mathbf{I} \end{pmatrix}^T \begin{pmatrix} \Delta\mathbf{r} \\ \mathbf{0} \end{pmatrix} \tag{3.63a}$$

$$\left(\boldsymbol{J}_f^T\boldsymbol{J}_f + \lambda^2\mathbf{I}\right)\Delta\boldsymbol{\varphi}_a = \boldsymbol{J}_f^T\Delta\mathbf{r} \tag{3.63b}$$

Die numerische Güte der Jacobimatrix $\boldsymbol{J}_f$ kann über eine Singulärwertzerlegung (siehe hierzu [65, 66, 67]) bestimmt werden (Gl. (3.64)).

$$\boldsymbol{J}_f = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T \text{ mit } \boldsymbol{U} \in \mathbb{R}^{m\times m}, \boldsymbol{V} \in \mathbb{R}^{n\times n}, \boldsymbol{D} \in \mathbb{R}^{m\times n} \tag{3.64a}$$

$$\boldsymbol{D} = \begin{pmatrix} \sigma_1 & 0 & 0 & & 0 & 0 & & 0 \\ 0 & \sigma_2 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 0 & \sigma_3 & & 0 & 0 & & 0 \\ & \vdots & & \ddots & & \vdots & & \vdots \\ 0 & 0 & 0 & & \sigma_r & 0 & & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ & \vdots & & & & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{pmatrix} \tag{3.64b}$$

$$\boldsymbol{J}_f = \sum_{i=1}^{m} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T \text{ , mit } \sigma_1 \geq \cdots \geq \sigma_r > 0 \tag{3.64c}$$

Mit Hilfe dieser Zerlegung kann der kleinste Wert von σ_i, welcher mit σ_{min} bezeichnet wird, bestimmt werden. Dieser ist ein Indikator dafür wie numerisch robust sich die Matrix $\boldsymbol{J}_f$ invertieren lässt. Die Lösung der DLS-Inversen lässt sich, wie in [68] hergeleitet, auch in der Form von Gl. (3.65) darstellen.

$$\boldsymbol{J}_f^T\left(\boldsymbol{J}_f\boldsymbol{J}_f^T + \lambda^2\boldsymbol{I}\right)^{-1} = \sum_{i=1}^{r} \frac{\sigma_i}{\sigma_i^2 + \lambda^2}\boldsymbol{v}_i\boldsymbol{u}_i^T \tag{3.65}$$

Hier erkennt man die Aufgabe des Dämpfungsterms λ in der Nähe von kleinen Werten von σ_i sehr deutlich. Ein zu hoher Dämpfungswert verschlechtert allerdings das Ergebnis der Invertierung. Numerisch günstiger ist es daher λ als Funktion von σ_{min} zu wählen. In [69] wird ein linearer Ansatz vorgeschlagen. Um eine untere und obere Grenze für λ vorgeben zu können und glatte Übergänge zu erhalten wird hier eine Sigmoid-Funktion nach Gl. (3.66) verwendet.

$$\lambda(\sigma_{min}) = \hat{\lambda} - \frac{\hat{\lambda} - \check{\lambda}}{1 + \mathrm{e}^{-s_\sigma(\sigma_{min} - \sigma_{mid})}} \tag{3.66}$$

Hierin ist der Parameter $\hat{\lambda} \in \mathbb{R}^+$ ein oberer Grenzwert und $\check{\lambda} \in \mathbb{R}^+$ ein unterer Grenzwert für λ (wobei $\hat{\lambda} > \check{\lambda} > 0$). Die Parameter s_σ und σ_{mid} können über das Gleichungssystem (3.67) bestimmt werden. Hierin sind die Parameter $\sigma_{lim1} \in \mathbb{R}^+$ und $\sigma_{lim2} \in \mathbb{R}^+$ untere und obere Schranke für σ_{min} zwischen welchen λ vom oberen Grenzwert $\hat{\lambda}$ zum unteren Grenzwert $\check{\lambda}$ verläuft. Sinnvolle Werte sind $k_1 = 0,9$ und $k_2 = 1,1$. Werte für σ_{lim1} und σ_{lim2} können aus dem Verlauf des minimalen Singulärwertes σ_{min} der Jacobimatrix $\boldsymbol{J}_f$ über den Arbeitsbereich des Roboters gefunden werden.

$$\lambda(\sigma_{lim1}) = k_1\hat{\lambda} \tag{3.67a}$$

$$\lambda(\sigma_{lim2}) = k_2\check{\lambda} \tag{3.67b}$$

$$\text{mit: } 0 < k_1 < 1 \wedge k_2 > 1 \wedge \sigma_{lim1} < \sigma_{lim2} \wedge k_2\check{\lambda} < k_1\hat{\lambda} \tag{3.67c}$$

Da die Berechnung von σ_{min} über eine Singulärwertzerlegung im Normalfall über einen numerischen Algorithmus erfolgt, kann σ_{min} und damit auch Gl. (3.66) nicht nach der Zeit differenziert werden.

Soll Gl. (3.66) in einem inversen Modell eingesetzt werden, so müssen im Modell entweder gefilterte Werte von λ verwendet werden, was durch einen kritisch gedämpften Tiefpassfilter $f_{krit}(.)$ nach Gl. (3.32) (ohne Überschwingen) erreicht werden kann oder es werden explizit alle Ableitungen von λ nach der Zeit zu Null gesetzt (Gl. (3.69)).

$$\sigma^*_{min} = f_{krit}(\sigma_{min}) \tag{3.68a}$$
$$\lambda^*(\sigma^*_{min}) = f_{krit}(\lambda(\sigma^*_{min})) \tag{3.68b}$$

$$\frac{d^j \lambda(\sigma_{min})}{dt^j} \stackrel{!}{=} 0 \text{ mit } j = 1, 2, \ldots \tag{3.69}$$

Wie man anhand der Singulärwertzerlegung sieht (siehe hierzu auch [70]), ist der Term $\boldsymbol{J}_f^T \boldsymbol{J}_f + \lambda^2 \mathbf{I}$ für $\lambda > 0$ nicht singulär (Gl. (3.65)), weshalb die Lösung für die Winkel in jedem Iterationsschritt als Gleichung (3.70) geschrieben werden kann.

$$\Delta\boldsymbol{\varphi}_a = \left(\boldsymbol{J}_f^T \boldsymbol{J}_f + \lambda^2 \mathbf{I}\right)^{-1} \boldsymbol{J}_f^T \Delta\mathbf{r} \tag{3.70}$$

Für eine schnelle Konvergenz empfiehlt es sich hierbei wieder die Lösung aus der inversen Kinematik des starren Roboters als Startlösung zu verwenden. Der Dämpfungsterm λ muss möglichst klein gewählt werden, um zu möglichst genauen Lösungen zu gelangen, aber gleichzeitig groß genug, um den Algorithmus im Falle von singulären Stellungen oder unerreichbaren Punkten zu stabilisieren. Für λ muss daher ein Kompromiss gefunden werden.

Zusätzlich zur Skalierung von λ ist es möglich, eine Begrenzungsfunktion für maximal zulässige Korrekturterme $\Delta\boldsymbol{\varphi}_a$ vorzugeben.

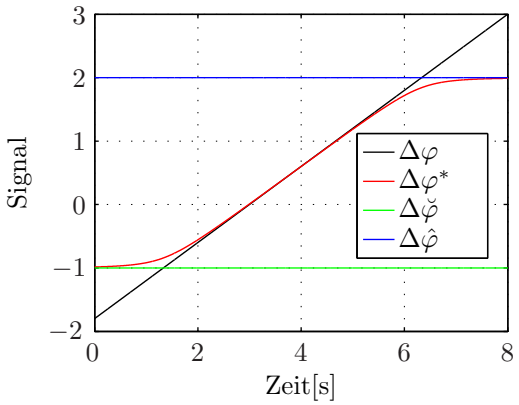


Abbildung 3.15: Verlauf der differenzierbaren Begrenzungsfunktion nach Gl. (3.71). Im Beispiel ist $\Delta\hat{\varphi} = 2$ und $\Delta\breve{\varphi} = -1$. In schwarz ist ein beispielhafter Verlauf von $\Delta\varphi$ über der Zeit dargestellt, in rot der von $\Delta\varphi^*$.

Damit hierbei keine sprungförmigen Änderungen auftreten, eignet sich eine Funktion

nach Gleichung (3.71).

$$\Delta\varphi_i^* = \Delta\varphi_i - \Delta\varphi_i(\zeta_1+\zeta_2) + \Delta\hat{\varphi}_i\zeta_1 + \breve{\varphi}_i\zeta_2 \tag{3.71a}$$

$$\zeta_1 = \frac{\operatorname{atan2}((\Delta\varphi_i - \Delta\hat{\varphi}_i)s)}{\pi} + \frac{1}{2} \tag{3.71b}$$

$$\zeta_2 = 1 - \frac{\operatorname{atan2}((\Delta\varphi_i - \Delta\breve{\varphi}_i)s)}{\pi} + \frac{1}{2} \tag{3.71c}$$

Hierin ist $\Delta\varphi_i^*$ die limitierte Achswinkeländerung der Achse i, wobei $\Delta\hat{\varphi}_i$ die maximal und $\Delta\breve{\varphi}_i$ die minimal zulässige Änderung des Winkels beschreibt. $s \gg 1$ ist ein Parameter zur Einstellung der Steilheit der Begrenzer-Funktion und ζ_1 und ζ_2 sind Hilfsvariablen. Den Verlauf der Funktion zeigt Abbildung 3.15. Durch die Verwendung der atan2-Funktion ist diese Begrenzungsfunktion beliebig oft differenzierbar.

3.7 Approximative Invertierung von Balken und FEM-Modellen

Balken können, kleine Verformungen vorausgesetzt, über einen Ansatz des mitbewegten Referenzkoordinatensystems in Kombination mit einer Taylorentwicklung – bezüglich der elastischen Koordinaten – modelliert werde (siehe Abschnitt 2.4.2). Dies führt auf eine Bewegungsgleichung in der Form von Gl. (3.72) für das Bauteil.

$$\begin{pmatrix} m\boldsymbol{I}_3 & & sym. \\ m\tilde{\boldsymbol{d}}_{CM} & \boldsymbol{J} & \\ \boldsymbol{C}_t & \boldsymbol{C}_r & \boldsymbol{M}e \end{pmatrix} \begin{pmatrix} \boldsymbol{a}_R \\ \dot{\boldsymbol{\omega}}_R \\ \ddot{\boldsymbol{q}} \end{pmatrix} + \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{K}_e\boldsymbol{q} + \boldsymbol{D}_e\dot{\boldsymbol{q}} \end{pmatrix} + \begin{pmatrix} 2\tilde{\boldsymbol{\omega}}_R\boldsymbol{C}_t^T\dot{\boldsymbol{q}} + \tilde{\boldsymbol{\omega}}_R\tilde{\boldsymbol{\omega}}_R\boldsymbol{d}_{CM} \\ \boldsymbol{G}_{rn}\dot{q}_n\tilde{\boldsymbol{\omega}}_R + \tilde{\boldsymbol{\omega}}_R\boldsymbol{J}\boldsymbol{\omega}_R \\ \boldsymbol{G}_{en}\dot{q}_n\tilde{\boldsymbol{\omega}} + \boldsymbol{O}_e\boldsymbol{\Omega} \end{pmatrix} = \begin{pmatrix} \boldsymbol{h}_t \\ \boldsymbol{h}_r \\ \boldsymbol{h}_e \end{pmatrix} \tag{3.72a}$$

$$\boldsymbol{\Omega} = \begin{pmatrix} \omega_x^2 & \omega_y^2 & \omega_z^2 & \omega_x\omega_y & \omega_y\omega_z & \omega_x\omega_z \end{pmatrix}^T \tag{3.72b}$$

Eine genauere Beschreibung und Herleitung findet sich in [34],[36] und [71], der Ansatz zum Linearisieren bezüglich eines mitbewegten Koordinatensystems ausführlich in [71] und [32].

FEM-Modelle von Teilkomponenten eines Roboters können nach einer Modalanalyse (die z. B. mit *SIMPACK* und *FEMBS* [72] durchgeführt werden kann) und Reduktion auf die wichtigsten Moden ebenfalls in die Form von Gl. (3.72) gebracht werden.

Für Balken und FEM-Modelle, welche über das SID-Format (siehe [36]) importiert werden können, ist diese Struktur in *Modelica* in der *FlexibleBodies Library* [34] implementiert.

Eine direkte und exakte Invertierung von mechanischen Systemen (hier in der Form von Robotern bzw. Manipulatoren angenommen), welche elastische Balken bzw. FEM-Modelle in der Form von Gl. (3.72) beinhalten, ist im Normalfall nicht möglich wenn die Motorgrößen (Moment oder Motorposition) als Ausgang und die Position der Manipulator-Spitze bzw. des TCP als Eingang des inversen Systems gewählt werden, da hierbei eine instabile Nulldynamik auftreten kann, oder die inverse Kinematik des

Systems nicht mehr eindeutig und exakt gelöst werden kann. Die Stabilität der Nulldynamik kann dabei näherungsweise durch eine Betrachtung der Nullstellen des linearisierten Systems erfolgen (siehe auch Abschnitt 3.2).

Für eine approximative Invertierung müssen daher die Gleichungen (3.72) und die Struktur der Systeme vereinfacht werden. Wie man in Gl. (3.72) erkennt, sind die modalen elastischen Verformungen $\boldsymbol{q}$ und die Starrkörpergrößen (bezüglich des Referenzkoordinatensystems R) $\boldsymbol{a}_R$ und $\dot{\boldsymbol{\omega}}_R$ über die Kopplungsmatrizen der Translation $\boldsymbol{C}_t$ und Rotation $\boldsymbol{C}_r$ verknüpft. Eine zusätzliche Verkopplung entsteht durch die Matrizen der gyroskopischen Terme bezüglich der rotatorischen Koordinaten $\boldsymbol{G}_{rn}$ und der elastischen Koordinaten $\boldsymbol{G}_{en}$, sowie den Abhängigkeiten der Terme des Tensors der Trägheitsmomente $\boldsymbol{J}$, der Zentrifugalkräfte $\boldsymbol{O}_e$ und dem Massemittelpunktes $\boldsymbol{d}_{CM}$ von sowohl den elastischen als auch den Starrkörperkoordinaten. Sind die Verformungen klein so können die Abhängigkeiten der Terme von den elastischen Koordinaten vernachlässigt werden, wobei ein gewisser Approximationsfehler (der proportional zu Größe der elastischen Verformungen ansteigt) in Kauf genommen werden muss. Da alle Terme in Gl. (3.72) aus einer Taylorentwicklung bezüglich der elastischen Koordinaten generiert werden (dies ist Teil des SID-Konzepts, welches auch in der *FlexibleBodies Library* für Balken übernommen wurde) ist es möglich, für die kritischen Elemente jeweils nur die Terme nullter Ordnung zu verwenden, welche nicht mehr von den elastischen Koordinaten, sondern nur noch von den Starrkörperkoordinaten des Referenzsystems R abhängen. Wir setzen daher $\boldsymbol{J} \approx \boldsymbol{J}_0$, $\boldsymbol{O}_e \approx \boldsymbol{O}_{e,0}$ und die Position des Massemittelpunktes zu $\boldsymbol{d}_{CM} \approx \boldsymbol{d}_{CM,0}$. Wir vernachlässigen zudem die Terme $\boldsymbol{C}_t$ und $\boldsymbol{C}_r$. Die modale Massenmatrix $\boldsymbol{M}_e$ wird als Einheitsmatrix angenommen mit entsprechend skalierten modalen Steifigkeiten $\boldsymbol{K}_e$ als Diagonalmatrix, was der Verwendung von Eigenformen zur Diskretisierung des Deformationsfeldes des elastischen Körpers entspricht. Da wir $\boldsymbol{C}_t$ vernachlässigen vereinfacht sich auch der Term $\boldsymbol{h}_e$, dieser wird daher $\boldsymbol{h}_e^*$ genannt. Durch die Vereinfachung von $\boldsymbol{d}_{CM}$ zu $\boldsymbol{d}_{CM,0}$ ändert sich ebenfalls der Term $\boldsymbol{h}_r$, welcher daher $\boldsymbol{h}_r^*$ genannt wird.

Soll das inverse Modell im Rahmen einer Vorsteuerung verwendet werden, ist es zusätzlich möglich, die Dämpfungsmatrix für das inverse System zu vergrößern $\boldsymbol{D}_e^* = \boldsymbol{D}_e + \boldsymbol{\Delta D}$. Dies führt zu einem numerisch stabileren Verhalten und verhindert zudem ein extrem langes (Nach-) Schwingungsverhalten in der inversen Lösung, ohne große Genauigkeitsverluste. Dadurch, dass im Normalfall ein zusätzlicher Regler in Kombination mit der Vorsteuerung eingesetzt wird, kann das System zusätzlich gedämpft werden und die im inversen Modell höher angenommene Dämpfung ausgeglichen werden.

Mit diesen Vereinfachungen gelangt man zu Gl. (3.73).

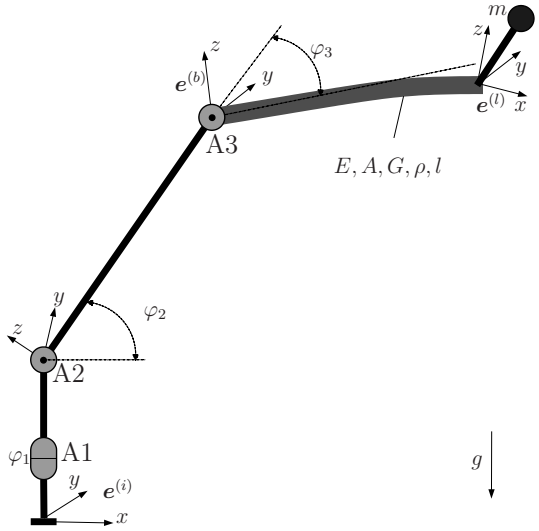
$$\begin{pmatrix} m\boldsymbol{I}_3 & & sym. \\ m\boldsymbol{d}_{CM,0} & \boldsymbol{J}_0 & \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{M}e \end{pmatrix} \begin{pmatrix} \boldsymbol{a}_R \\ \dot{\boldsymbol{\omega}}_R \\ \ddot{\boldsymbol{q}} \end{pmatrix} + \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{K}_e\boldsymbol{q} + \boldsymbol{D}_e^*\dot{\boldsymbol{q}} \end{pmatrix} + \begin{pmatrix} \tilde{\boldsymbol{\omega}}_{\boldsymbol{R}}\tilde{\boldsymbol{\omega}}_{\boldsymbol{R}}\boldsymbol{d}_{CM,0} \\ \tilde{\boldsymbol{\omega}}_{\boldsymbol{R}}\boldsymbol{J}_0\boldsymbol{\omega}_R \\ \boldsymbol{O}_{e,0}\boldsymbol{\Omega} \end{pmatrix} = \begin{pmatrix} \boldsymbol{h}_t \\ \boldsymbol{h}_r^* \\ \boldsymbol{h}_e^* \end{pmatrix} \tag{3.73a}$$

$$\boldsymbol{\Omega} = \begin{pmatrix} \omega_x^2, & \omega_y^2, & \omega_z^2, & \omega_x\omega_y, & \omega_y\omega_z, & \omega_x\omega_z \end{pmatrix}^T \tag{3.73b}$$

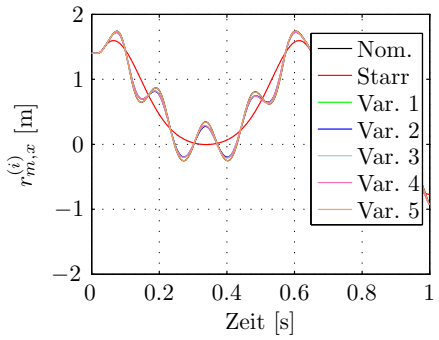
Diese Gleichungen sind entkoppelt bezüglich der elastischen Verformungen und den Starrkörperbewegungen. Sofern die inverse Kinematik des Systems exakt lösbar ist,

kann somit eine Invertierung durchgeführt werden. Damit die inverse Kinematik eindeutig lösbar ist, sollten möglichst wenige Moden für die unterschiedlichen Deformationsrichtungen verwendet werden.

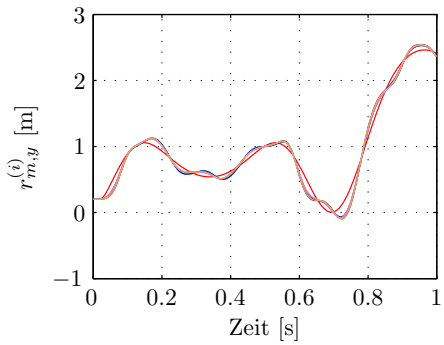
Mit den Vereinfachungen muss allerdings ein Verlust an Genauigkeit in Kauf genommen werden. Eine genaue Angabe über den Genauigkeitsverlust ist für den allgemeinen Fall nur schwer möglich. Grundsätzlich wird diese Approximation aber umso ungenauer, je größer die elastischen Verformungen im System sind und je größer der Beitrag der gyroskopischen Terme der elastischen Verformungen auf die Gesamtlösung sind. Es empfiehlt sich daher stets eine Überprüfung der Genauigkeit mit Hilfe eines Vorwärtsmodells, bei dem die Stabilität der Nulldynamik keine Rolle spielt.



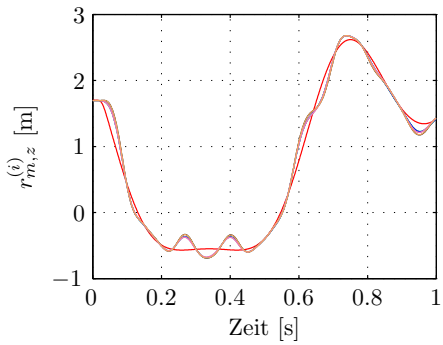
(a) Schematischer Aufbau des Beispielsystems: Dreiachsiger Manipulator mit elastischem Balken nach Achse 3 (A3) und exzentrischer versetzter Last m.



(b) Vergleich der x-Komponente $r^{(i)}_{m,x}$ des berechneten Massemittelpunktes bezüglich des Inertialsystems.



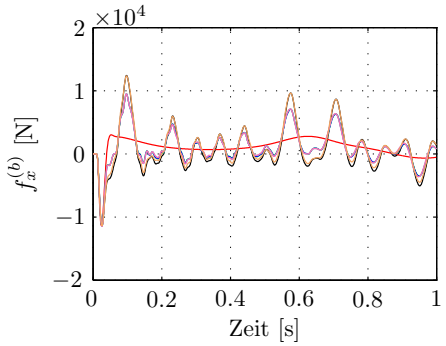
(c) Vergleich der y-Komponente $r^{(i)}_{m,y}$ des berechneten Massemittelpunktes bezüglich des Inertialsystems.



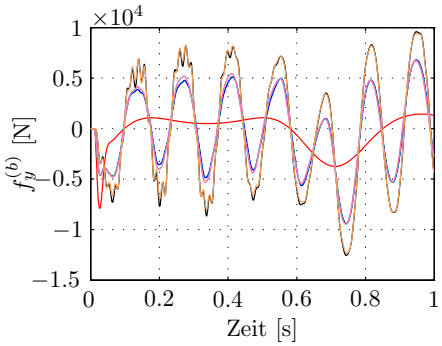
(d) Vergleich der z-Komponente $r^{(i)}_{m,z}$ des berechneten Massemittelpunktes bezüglich des Inertialsystems.

Abbildung 3.16: Aufbau des Beispielsystems und Auswirkung der Vereinfachungen auf die berechnete Position der Last, bei sinusförmiger Bewegung der drei Achsen.

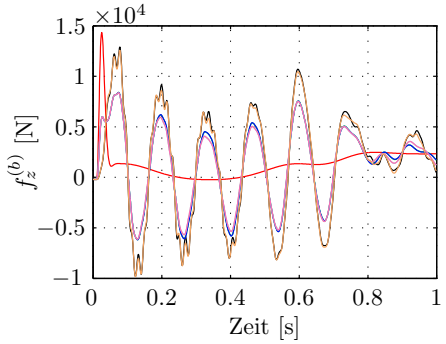
Um den Effekt der Approximation zu überprüfen, wurden die Vereinfachungen an einem Beispielsystem überprüft. Die Ergebnisse zeigen Abb. 3.16 und 3.17. Das Beispielsystem ist ein dreiachsiger Manipulator (Achsen: A1, A2 und A3). Das Verbindungselement von



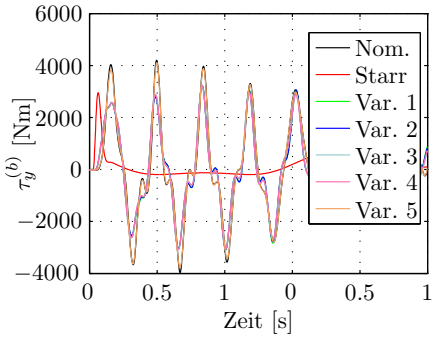
(a) Vergleich der auf die Achse A3, in x-Richtung bezüglich des lokalen Koordinatensystems $\boldsymbol{e}^{(b)}$, wirkenden Kraft.



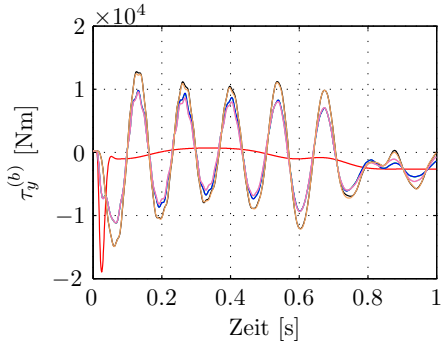
(b) Vergleich der auf die Achse A3, in y-Richtung bezüglich des lokalen Koordinatensystems $\boldsymbol{e}^{(b)}$, wirkenden Kraft.



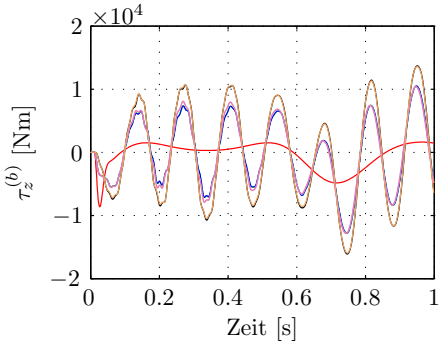
(c) Vergleich der auf die Achse A3, in z-Richtung bezüglich des lokalen Koordinatensystems $\boldsymbol{e}^{(b)}$, wirkenden Kraft.



(d) Vergleich des auf die Achse A3, in x-Richtung bezüglich des lokalen Koordinatensystems $\boldsymbol{e}^{(b)}$, wirkenden Moments.



(e) Vergleich des auf die Achse A3, in y-Richtung bezüglich des lokalen Koordinatensystems $\boldsymbol{e}^{(b)}$, wirkenden Moments.



(f) Vergleich des auf die Achse A3, in z-Richtung bezüglich des lokalen Koordinatensystems $\boldsymbol{e}^{(b)}$, wirkenden Moments.

Abbildung 3.17: Auswirkung der Vereinfachungen auf die berechnete Kräfte und Momente am Ort der Einspannung des Balkens bei A3.

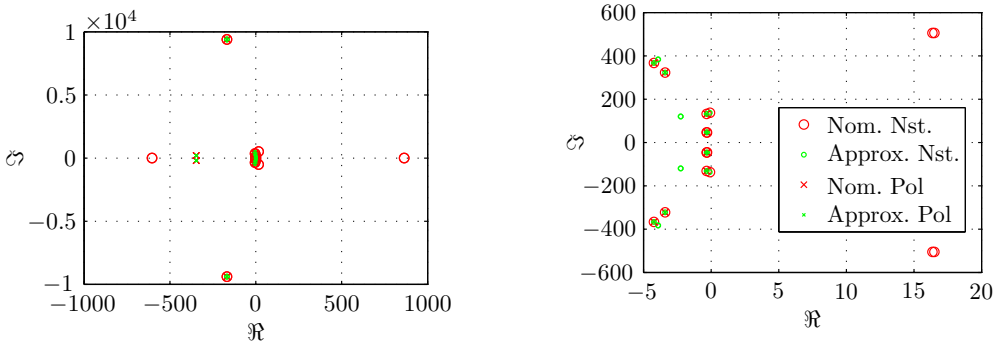
A2 bis A3 ist als starr angenommen, nach A3 befindet sich ein elastischer Balken, der auf Basis der MFBL modelliert wurde. Für den Balken wurden jeweils die ersten zwei Moden für die Biegung in der xy- und xz-Ebene (bezüglich des lokalen mitbewegten Koordinatensystems $\boldsymbol{e}^{(b)}$, siehe Abb. 3.16(a)) und der Torsion berücksichtigt. Am Ende des Balkens befindet sich, exzentrisch versetzt, eine Last m. Für die Simulation wird für Achsen A1 bis A3 jeweils eine identische, gefilterte sinusförmige Bewegung als Anregung angenommen. Es wurden verschiedene Vereinfachungen von Gl. (3.72) untersucht:

- Nominal: Keine Vereinfachung von Gl. (3.72). Taylorentwicklung zweiter Ordnung bezüglich der elastischen Koordinaten.
- Starr: Keine elastischen Verformungen ($\boldsymbol{q} \equiv \boldsymbol{0}$); reine Starrkörperlösung für den Balken.
- Variante 1: Lösung mit Vereinfachungen nach Gl. (3.73).
- Variante 2: Lösung nach Gl. (3.72) mit Vernachlässigung von $\boldsymbol{G}_{rn}, \boldsymbol{G}_{en}, \boldsymbol{C}_t, \boldsymbol{C}_r$ sowie den Vereinfachungen $\boldsymbol{O}_{e,0}$ und $\boldsymbol{h}_e^*$.
- Variante 3: Lösung nach Gl. (3.72) mit Vernachlässigung von $\boldsymbol{C}_t, \boldsymbol{C}_r$ sowie der Vereinfachung $\boldsymbol{h}_e^*$.
- Variante 4: Lösung nach Gl. (3.72) mit Vernachlässigung von $\boldsymbol{C}_t, \boldsymbol{C}_r$.
- Variante 5: Lösung nach Gl. (3.72) mit Vereinfachung von $\boldsymbol{C}_t, \boldsymbol{C}_r$ und $\boldsymbol{h}_e$, durch die ausschließliche Verwendung der Terme nullter Ordnung, daher $\boldsymbol{C}_{t,0}, \boldsymbol{C}_{r,0}$ und $\boldsymbol{h}_{e,0}$.

Wie erwartet wurde hat die Vernachlässigung von $\boldsymbol{C}_t, \boldsymbol{C}_r$ den stärksten Effekt auf die Lösung. Eine Näherung von $\boldsymbol{C}_t, \boldsymbol{C}_r$ durch $\boldsymbol{C}_{t,0}, \boldsymbol{C}_{r,0}$, reicht bereits aus, um eine Lösung sehr nahe der Nominalen zu erhalten (siehe Var. 5 in Abbildung 3.16 und 3.17 im Vergleich zu der Nominallösung). Allerdings reicht eine Vereinfachung durch $\boldsymbol{C}_{t,0}, \boldsymbol{C}_{r,0}$ nicht aus, um das System stabil invertieren zu können. $\boldsymbol{C}_t, \boldsymbol{C}_r$ tragen damit die Hauptverantwortung für die instabile Nulldynamik. Die weiteren Vereinfachungen haben im Vergleich nur eine sehr schwache Auswirkung und unterscheiden sich nur im Detail. Die Starrkörperlösung weicht, aufgrund der fehlenden Dynamik, deutlich von allen Näherungslösungen ab.

Abbildung 3.18 zeigt die Pol- und Übertragungs-Nullstellen für das Nominal-Modell im Vergleich zum approximierten Modell nach Gl. (3.72), wenn als Eingangsgrößen des linearisierten MIMO-Modells die Winkel der Achsen 1 bis 3 gewählt werden und als Ausgänge die (relativen) kartesischen Verschiebungen in den drei Raumrichtungen des globalen Koordinatensystems (x,y,z). Wie man der Abbildung 3.18 entnehmen kann, treten für das approximierte System keine Übertragungs-Nullstellen mit positivem Realteil auf.

Die Berechnung der Übertragungs-Nullstellen (Engl.: Transmission zeros) erfolgt dabei numerisch in *MATLAB* nach dem Verfahren welches in [73] vorgestellt wurde. Weitere Details zu Übertragungs-Nullstellen finden sich in [74] und [75].



(a) Pol- und Übertragungs-Nullstellen im Vergleich

(b) Pol- und Übertragungs-Nullstellen im Vergleich (Detailansicht)

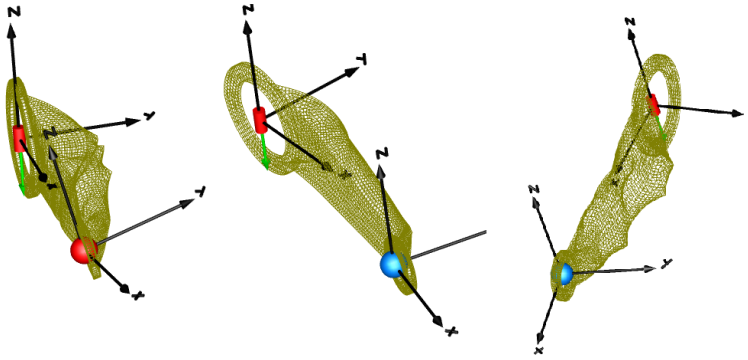
Abbildung 3.18: Pol- und Übertragungs-Nullstellen (abgekürzt mit Nst.) für das Nominal-Modell im Vergleich zum approximierten Modell nach Gl. (3.72). Übertragungs-Nullstellen sind als Kreise, Pole als Kreuze dargestellt. Für das Nominal-System in rot, für die Approximation in grün.

Wie man Abb. 3.16 und 3.17 entnehmen kann eignet sich die Annäherung von Gl. (3.72) (Nominallösung) durch Gl. (3.73) für typische Bewegungen und Steifigkeiten, wie sie in der Robotik üblich sind, um zu einem inversen Modell zu gelangen.

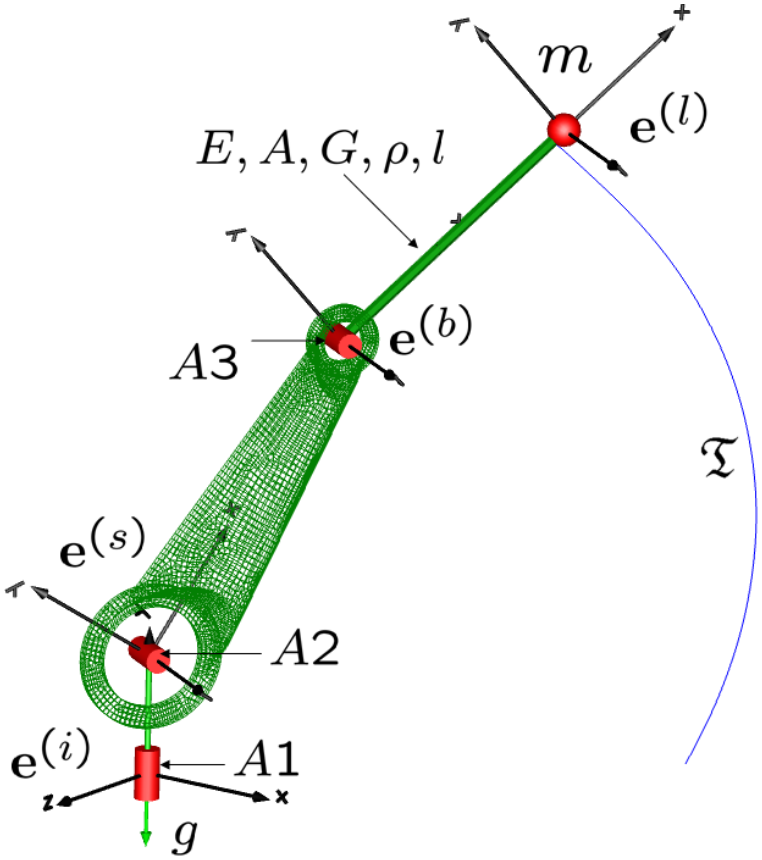
Dieses Verfahren soll anhand einer Simulation es einfachen dreiachsigen Manipulator demonstriert werden, der sowohl einen FEM-Körper als auch einen elastischen Balken beinhaltet (siehe hierzu auch [76]). Der Manipulator besitzt drei Achsen. Alle Achsen bestehen aus einem Antriebsstrangmodell[14], welches durch eine Trägheit (für Motor und Welle) und einem linearen Feder-Dämpfer System (für das Getriebe), sowie linearer Reibung modelliert ist. Jede Achse wird über einen einfachen PID-Regler, in Kombination mit einer Vorsteuerung, im Sinne des Konzepts der zwei Freiheitsgrade (siehe Abschnitt 5.1), geregelt. Eingangssignal der Regler ist der Fehler zwischen motorseitiger Soll- und Istposition, Ausgangssignal ist das Motormoment. Zum Ausgangsmoment wird noch das Vorsteuermoment addiert, über die Vorsteuerung wird zudem die motorseitige Sollposition berechnet. Für den Regler stehen, wie bei Industrieroboter üblich, keine weiteren Messgrößen als die Motorposition (durch Differenzierung damit auch die Motorgeschwindigkeit) zur Verfügung. Das Verbindungselement zwischen der ersten und zweiten Achse wird als starre Verbindung angenommen, das Verbindungsglied zwischen der zweiten und dritten Achse, die Schwinge, wird als modaler Körper modelliert. Diese wird generiert über eine FEM-Analyse in Kombination mit einer Modalreduktion, welche in *SIMPACK* (siehe [72]) durchgeführt wurde. Das Ergebnis dieser Analyse wird als SID-Datei [36], mit Hilfe der MFBL in *Modelica/Dymola* eingebunden. Hierbei werden die 10 dominantesten Moden (mit den niedrigsten Frequenzen) berücksichtigt. Das Schwinge-Modell kann hierdurch wieder auf die Form von Gl. (3.72) überführt werden. Drei der 10 gewählten Moden sind in Abbildung 3.19(a) dargestellt.

Das Verbindungselement nach der dritten Achse ist als elastischer Balken mit drei Biegemoden in der xy-Ebene im lokalen Koordinatensystem $\boldsymbol{e}^{(b)}$ modelliert (Abb. 3.19).

[14]Die Antriebsstränge werden durch eindimensionale Modelle modelliert (rotationssymmetrisch zur jeweiligen Achse). Aufgrund der hohen Getriebeübersetzung können gyroskopische Kopplungseffekte zwischen den Antriebsstrangträgheiten und der restlichen Mechanik vernachlässigt werden.

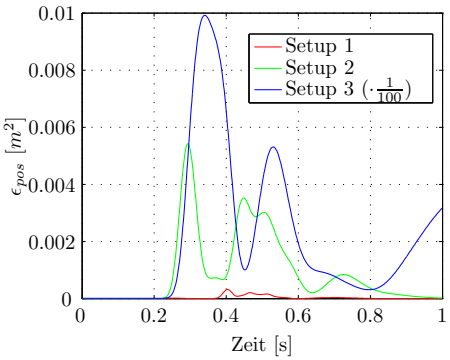


(a) Beispiel Roboter-Schwinge. Das Bild zeigt 3 der 10 gewählten Moden, die im Modell berücksichtigt werden.

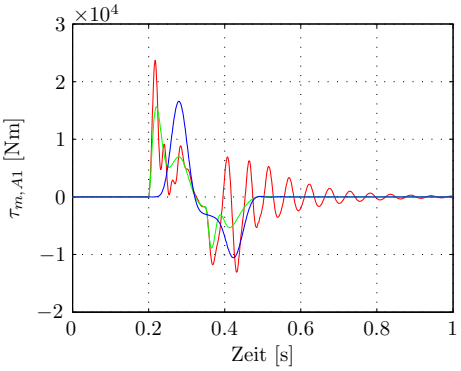


(b) Beispielsystem eines flexiblen Manipulators. Er besteht aus drei flexiblen Antriebssträngen, flexibler Schwinge (FEM-Modell) und flexiblen Balken mit drei Biegemoden in der xy-Ebene. Die Gravitation wirkt in der negativen $\boldsymbol{e}_y^{(I)}$ Richtung. Das Ziel ist es die Last an der Spitze entlang einer Trajektorie $\mathfrak{T}$ (blau im Bild) zu bewegen.

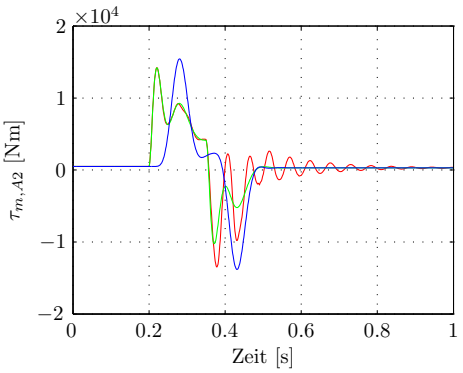
Abbildung 3.19: Beispielsystem zur Invertierung von Balken und FEM-Modellen. Eine genauere Beschreibung findet sich im Text.



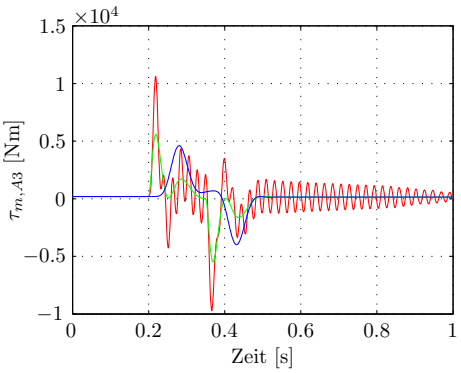
(a) Abweichung $\epsilon_{pos} = \|\boldsymbol{r}_{t,ref} - \boldsymbol{r}_{t,m}\|^2$ der Last von der Solltrajektorie $\mathfrak{T}$.



(b) Vergleich der, über die inversen Modelle, berechneten Vorsteuermomente $\tau_{m,A1}$ für Achse A1.



(c) Vergleich der, über die inversen Modelle, berechneten Vorsteuermomente $\tau_{m,A2}$ für Achse A2.



(d) Vergleich der, über die inversen Modelle, berechneten Vorsteuermomente $\tau_{m,A3}$ für Achse A3.

Abbildung 3.20: Vergleich der drei verschiedenen inversen Modelle, zur Berechnung der Vorsteuergrößen $\boldsymbol{\Sigma}_{ff}$ für das Beispiel-System. Das Streckenmodell, inklusive den drei PID-Motor Reglern, ist jeweils identisch. Rot: Konfiguration 1. Grün: Konfiguration 2. Blau: Konfiguration 3 (in Bild 3.20(a) um Faktor 100 verkleinert dargestellt).

Am Ende dieses Balkens befindet sich die Last m. Auf das System wirkt die Gravitation in negativer $\boldsymbol{e}_y^{(i)}$ Richtung. Ziel ist es, die Last m, möglichst exakt, entlang der Trajektorie $\mathfrak{T}$ zu bewegen. Eine Übersicht des Testmodells zeigt Abb. 3.19(b).

Durch eine Analyse dieses System mit Hilfe mehrerer Linearisierungen an unterschiedlichen Arbeitspunkten erkennt man, dass Übertragungs-Nullstellen mit positivem Realteil im System vorhanden sind, wenn die Motorpositionen oder Motormomente als Eingang und die Position $\boldsymbol{r}_{t,m}$ der Last m als Ausgang gewählt werden. Es ist daher notwendig approximative inverse Modelle für die Vorsteuerung zu verwenden. Um den Effekt der Vorsteuerungen auf das Systemverhalten des geregelten Systems zu verdeutlichen wurde drei verschiedene inverse Modelle verglichen:

- **Konfiguration 1:** Im inversen Modell wird die Elastizität der Antriebsstränge und der Struktur berücksichtigt. Um ein stabiles inverses Modell zu erhalten, wird das Modell in zwei Teilmodelle aufgeteilt. Das erste Teilmodell dient zum Berech-

nen der Motorgrößen für die Achsen A1 und A2. In diesem Teilmodell werden die Gleichungen der Schwinge durch Gl. (3.73) approximiert. Zudem wird die modale Dämpfung erhöht ($\boldsymbol{D}_e^* = \boldsymbol{D}_e + \boldsymbol{\Delta D}$). In Teilmodell eins wird, um Kopplungseffekte zu vermeiden, der Balken als Starrkörper modelliert. Das zweite Teilmodell dient zum Berechnen der Motorgrößen von A3. Hierin wird die Schwinge als starr angenommen, der Arm als elastischer Balken. Die Gleichungen des Balkens werden ebenfalls durch Gl. (3.73) mit erhöhter Dämpfung $\boldsymbol{D}_e^*$ approximiert und nur der erste Biegemode berücksichtigt. Zudem wird ein virtueller Ausgang auf der Höhe $\xi = 0,9$ verwendet (siehe Abschnitt 3.5.1).

- **Konfiguration 2:** Im inversen Modell wird die Elastizität der Antriebsstränge berücksichtigt. Die Struktur wird durch Starrkörper modelliert. Durch die starre Struktur tritt keine instabile Nulldynamik auf.

- **Konfiguration 3:** Im inversen Modell wird keine Elastizität berücksichtigt. Antriebsstrang und Struktur bestehen nur aus Starrkörpern.

Die Invertierung der einzelnen Modelle erfolgt jeweils durch das Vertauschen von Ein- und Ausgängen in *Modelica/Dymola*, wie in Kapitel 3.4 beschrieben. Da die Bewegungsgleichungen hierfür differenziert werden müssen (für Konfiguration 1 dreimal um stetige Momentenverläufe zu erreichen), muss die Trajektorie $\mathfrak{T}$ differenzierbar sein. Sie wird daher über Bessel-Tiefpassfilter gefiltert.

Abb. 3.20 zeigt einige Ergebnisse für die verschieden Konfigurationen für die inversen Modelle als Vorsteuerung $\boldsymbol{\Sigma}_{ff}$. Für alle drei Varianten wurde jeweils das identische (nicht vereinfachte) Streckenmodell verwendet. Die Parameter der PID-Regler wurden über eine Optimierung ermittelt und sind jeweils identisch. Störungen sind im Modell vernachlässigt und alle Parameter werden als exakt bekannt angenommen. Wie man Abb. 3.20(a) entnehmen kann, sinkt der Fehler $\epsilon_{pos} = \|\boldsymbol{r}_{t,ref} - \boldsymbol{r}_{t,m}\|^2$ beim Folgen der Trajektorie $\mathfrak{T}$ deutlich, je mehr Elastizitäten in $\boldsymbol{\Sigma}_{ff}$ berücksichtigt werden. Den größten Effekt macht die Kompensation der Elastizität der Getriebe aus. Durch die Kompensation der strukturelastischen Effekte kann der Fehler jedoch noch einmal deutlich reduziert werden. Ohne eine Kompensation jeglicher Elastizitäten (Konfiguration 3), treten sehr große Abweichungen auf.

Die Kompensation der strukturelastischen Effekte führt jedoch dazu, dass auch nach dem Ende der kommandierten Bewegung (nach etwa $0,5s$ in den Abbildungen), Stellgrößenänderungen aufgebracht werden müssen, um die Schwingungen zu dämpfen (siehe Abb. 3.20(b) bis 3.20(d)). Dies ist auch der Grund, warum eine erhöhte modale Dämpfung $\boldsymbol{D}_e^*$ im inversen System sinnvoll ist, da sie extrem lang wirkende Stellgrößenänderungen (mit sehr kleinen Amplituden) verhindert, die im Normalfall, aufgrund der zusätzlichen Dämpfung durch den Regler, nicht nötig sind oder nur noch einen sehr geringen Einfluss auf die Güte der Bewegung haben.

3.8 Approximative Inverse Modelle des strukturelastischen Roboters

Mit Hilfe der vorgestellten Methoden und Algorithmen ist es möglich approximative inverse Modelle für Roboter zu erstellen. Im Folgenden wird aus den verschiedenen Varianten ein Algorithmus zur Generierung approximativer inverser Modelle gewählt, der sich als robust und für eine Anwendung in Echtzeit geeignet gezeigt hat.

Neben der Berechnungsvorschrift des Algorithmus wird hierbei auch auf die Stabilität des Algorithmus bzw. der Begrenztheit der berechneten Größen eingegangen. Dies ist eine wichtige Voraussetzung um den Algorithmus am realen Roboter verwenden zu können.

Danach werden zwei Varianten der approximativen inversen Modelle für den KR16 und den KR210 vorgestellt und Simulationsergebnisse für diese Modelle gezeigt. In Kapitel 6 werden dann auch Ergebnisse und Messungen für die Anwendung dieser (echtzeitfähigen) inversen Modelle am realen Roboter dargestellt.

Grundlage für die Erstellung ist zunächst immer ein nichtlineares Streckenmodell. Wie in diesem Kapitel beschrieben wurde, müssen diese Synthesemodelle für die Invertierung modifiziert werden, damit diese invertierbar werden. Die wichtigsten Punkte hierzu werden noch einmal kurz zusammengefasst:

- Ausschließliche Verwendung von (eindeutig) invertierbaren, stetig differenzierbaren Kennlinien und Funktionen.
- Eingangssignale und Gleichungen des Systems müssen dem Algorithmus von Pantelides [49] entsprechend oft differenzierbar sein. Alternativ können Filter der entsprechenden Ordnung verwendet werden um differenzierbare Signale zu erzeugen.
- Ein- und Ausgänge müssen so gewählt werden, dass die daraus resultierende Nulldynamik des Systems (global) stabil ist. Kann diese Bedingung nicht erfüllt werden, so müssen Approximationsansätze für das Modell getroffen werden, so dass diese Bedingung erfüllt ist.
- Ist die Invertierung nicht eindeutig (z. B. aufgrund der inversen Kinematik) so müssen zusätzliche Bedingungen oder Näherungen eingeführt werden, damit eine Lösung gefunden werden kann.

Wie bereits bei den in Kapitel 2 vorgestellten Modellen bietet es sich an, für die Invertierung der Robotermodelle, die elastische Struktur des Roboters und die elastischen Antriebsstränge des Roboters getrennt zu betrachten.

3.8.1 Aufbau des Algorithmus zur Berechnung eines approximativen strukturelastischen inversen Modells

Eingangssignal für den Algorithmus ist ein zeitlicher Verlauf der Orientierung und Position der Last am TCP des Roboters (Sollstellung). Dieser kann in der Form einer

homogenen Matrix ${}^{L}_{0}\boldsymbol{T}_{soll}(t)$, als Funktion der Zeit, dargestellt werden. Mit Hilfe von ${}^{L}_{0}\boldsymbol{T}_{soll}$ können, über eine inverse Kinematik des starren Roboters, hieraus direkt die abtriebsseitigen Winkel $\boldsymbol{\varphi}_{a,r}$ für alle 6 Achsen des Roboters berechnet werden. Diese Winkel entsprechen einer Vorgabe für eine perfekt starre Kinematik. Die Differenzierbarkeit der Winkel wird über Besselfilter erreicht, deren minimale Ordnung anhand des Algorithmus von Pantelides [49] bestimmt werden kann, welche sich aus der Komplexität der gewählten Antriebsstrang-Modellierung ergibt.

Die Verformungen der Struktur werden über die in Abschnitt 3.5.2 vorgestellte Approximation der Verformungen berechnet. Hierzu werden zunächst – basierend auf einem Starrkörper-Parallelmodell der Roboter-Struktur – die Schnittmomente an den Positionen der elastischen Gelenke berechnet. Aus diesen Momenten werden dann wie in Abschnitt 3.5.2 beschrieben, die entsprechenden Verformungen $\hat{\boldsymbol{\varphi}}_e$ der Struktur über die Differentialgleichungen der Feder-Dämpfer-Elemente berechnet.

Über den in Abschnitt 3.6.3 vorgestellten DLS-Algorithmus werden dann die korrigierten abtriebsseitigen Winkel $\boldsymbol{\varphi}_{a,e}$ des strukturelastischen Models berechnet. Um Rechenzeit zu sparen, und die Komplexität zu begrenzen, wird für den DLS-Algorithmus dafür eine teillinearisierte Jacobimatrix $\boldsymbol{J}_f(\hat{\boldsymbol{\varphi}}_e, \boldsymbol{\varphi}_a)$ verwendet. Dies bedeutet, dass für die Jacobimatrix eine Taylorentwicklung erster Ordnung für die strukturelastischen Freiheitsgrade $\hat{\boldsymbol{\varphi}}_e$ vorgenommen wird, während die Gelenkwinkel $\boldsymbol{\varphi}_a$ der Achsen nicht linearisiert werden.

Da die elastischen Verformungen des Roboters klein sind ist dieser Schritt gerechtfertigt, da die zusätzlichen Freiheitsgrade der elastischen Gelenke ansonsten zu einer extremen Vergrößerung der einzelnen Komponenten der Jacobimatrix $\boldsymbol{J}_f$ führt, (durch eine Vielzahl von verschachtelten Sinus und Kosinus-Termen) ohne eine wesentliche Verbesserung in der Genauigkeit des DLS-Algorithmus zu erreichen.

Um zusätzlich Rechenzeit einzusparen wird nur ein Iterationsschritt für den DLS-Algorithmus für jedes Zeitintervall der numerischen Integration durchgeführt. Dies hilft dabei das Modell in Echtzeit auszuwerten[15], da durch diese Maßnahme in jedem Schritt in etwa konstante Rechenzeit für den DLS-Algorithmus benötigt wird. Die Auswertung in einem Schritt führt auch dazu, das die inverse Kinematik direkt als geschlossene Formel, als Teil der Systemgleichungen eingeht, und dadurch direkt differenzierbar bleibt, was für iterative Algorithmen nicht gewährleistet wäre. Durch die hohe Taktrate in der Auswertung des inversen Modells bleibt der Fehler der durch diese Auswertung der Korrekturterme $\Delta\boldsymbol{\varphi}_a$ in einem Schritt entsteht sehr klein. Mit Hilfe der korrigierten abtriebsseitigen Winkel $\boldsymbol{\varphi}_{a,e} = \boldsymbol{\varphi}_{a,r} + \Delta\boldsymbol{\varphi}_a$ und den Verformungen der elastischen Struktur $\hat{\boldsymbol{\varphi}}_e$, können dann in einem zweiten Schritt die benötigten abtriebsseitigen Momente $\hat{\boldsymbol{\tau}}_{a,e}$ für die Antriebsstränge des Roboters berechnet werden. Hierzu müssen die Bewegungsgleichungen des elastischen Roboters gelöst werden, wobei die Verformungen durch $\hat{\boldsymbol{\varphi}}_e$ und die korrigierten Winkel abtriebsseitigen Winkel $\boldsymbol{\varphi}_{a,e}$ berücksichtigt werden. Durch die Invertierung der Antriebsstränge welche – aufgrund der stabilen

[15] Würde die Iteration erst nach einer gewissen Genauigkeit abgebrochen, so kann keine genaue Vorhersage für die benötigte Rechenzeit getroffen werden. Es müsste für eine Implementierung in Echtzeit immer eine konservative „Worst Case“ - Annahme getroffen werden. Alternativ könnte die Anzahl der Iterationen auch auf einen fixen Wert beschränkt werden, der durch die verfügbare Rechenzeit bestimmt wird. Für jede Iteration muss allerdings sowohl die Jacobimatrix $\boldsymbol{J}_f$ als auch die Vorwärtskinematik des elastischen Roboters neu berechnet werden.

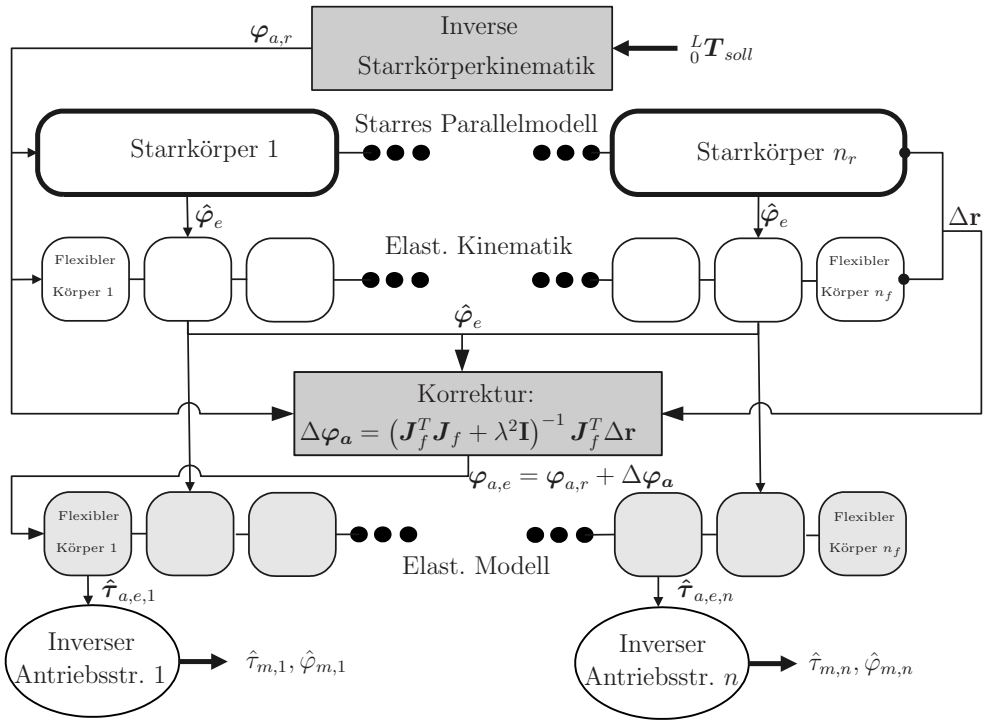


Abbildung 3.21: Schematische Darstellung des Algorithmus für das approximative strukturelastische inverse Modell (ASIM). Ausgehend von dem gewünschten Verlauf von Position und Orientierung der Last am TCP des Roboters ${}^{L}_{0}\boldsymbol{T}_{soll}$ werden über eine inverse Starrkörperkinematik die dazugehörigen Winkel $\boldsymbol{\varphi}_{a,r}$ des starren Systems berechnet. Hierdurch können Schnittkräfte im starren Parallelmodell berechnet werden, über welche approximativ die elastischen Verformungen $\hat{\boldsymbol{\varphi}}_e$ der Struktur bestimmt werden können. Über die Kinematik des elastisch verformten Systems kann die Abweichung $\Delta\boldsymbol{r}$ der Last von ${}^{L}_{0}\boldsymbol{T}_{soll}$ durch die Verformungen bestimmt werden. Mit Hilfe des DLS-Algorithmus können dann Korrekturwinkel $\Delta\boldsymbol{\varphi}_a$ berechnet werden. Anschließend werden die Gelenk-Schnittkräfte $\hat{\boldsymbol{\tau}}_{a,e}$ der Antriebsstränge, welche sich aufgrund der korrigierten Winkel $\boldsymbol{\varphi}_{a,e}$ und der Verformung der Struktur ergeben, berechnet. Über diese Größen können über invertierte Antriebsstrangmodelle die motorseitigen Größen $\hat{\boldsymbol{\tau}}_m$ und $\hat{\boldsymbol{\varphi}}_m$ berechnet werden. Die Invertierung wird in Abschnitt 3.8.2 genauer vorgestellt.

Nulldynamik – exakt erfolgt, können dann die motorseitigen Größen $\hat{\boldsymbol{\varphi}}_m$, $\dot{\hat{\boldsymbol{\varphi}}}_m$ und $\hat{\boldsymbol{\tau}}_m$ des inversen Modells berechnet werden, welche gleichzeitig die Ausgänge des Systems darstellen.

Diese Größen können zur Vorsteuerung des Roboters an einen Regler übergeben werden, welcher eventuelle Abweichungen und Modellfehler korrigiert und das System stabilisiert und dämpft.

Eine schematische Übersicht über den Algorithmus zeigt Abb. 3.21. Im nächsten Abschnitt 3.8.2 wird noch einmal im Detail auf die Berechnungsvorschrift des Algorithmus eingegangen, wobei auch auf die Stabilität des Algorithmus eingegangen wird.

3.8.2 Berechnungsvorschrift und Stabilitätsnachweis des approximativen inversen Modells

Die Berechnung zur approximativen Inversen des strukturelastischen Roboters lässt sich in mehrere Teilschritte unterteilen, die im Einzelnen näher analysiert werden sollen.

Die Struktur der Bewegungsgleichungen das approximierten Robotermodells (mit n_a Gelenken und n_e strukturelastischen Freiheitsgraden, sowie $n_m \equiv n_a$ Antriebsstrang-Freiheitsgraden) lassen sich in Form von Gleichung (3.74) schreiben.

Die Null-Matrizen in der Massenmatrix in Gl. (3.74) ergeben sich durch die Vernachlässigung der Kreiselkräfte, welche durch die Motor-Rotoren auf die Kinematik (Struktur) des Roboters wirken würden. Wie bereits beschrieben ist diese Vereinfachung für die untersuchten Roboter gerechtfertigt, da hierbei hohe Übersetzungen in den Getrieben vorhanden sind, was den Einfluss der Rotormassen und Trägheiten der Elektromotoren, auf die Dynamik des Roboters, stark reduziert (siehe auch [15]).

Zudem führt eine Berücksichtigung der Kopplungsterme, wie in [77] für ein Starrkörpermodell mit elastischen Antrieben gezeigt wurde, zu einer deutlicher Erhöhung der benötigten Differentiation der Bewegungsgleichungen bei der Aufstellung der inversen Dynamik. Im allgemeinen Fall (ohne Dämpfung) müssen die Bewegungsgleichungen bei Berücksichtigung dieser Kopplungsterme $2n_m$ mal differenziert werden um die motorseitigen Größen berechnen zu können. Dies ist in der Praxis nicht akzeptabel, da durch eine $2n_m$-fache Differenzierung Gleichungssysteme einstehen würden, die nicht mehr in Echtzeit berechenbar sind.

$$\underbrace{\begin{pmatrix} \boldsymbol{M}_{aa} & \boldsymbol{M}_{ae} & \boldsymbol{0} \\ \boldsymbol{M}_{ea} & \boldsymbol{M}_{ee} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{\Theta} \end{pmatrix}}_{\boldsymbol{M}} \underbrace{\begin{pmatrix} \ddot{\boldsymbol{\varphi}}_a \\ \ddot{\boldsymbol{\varphi}}_e \\ \ddot{\boldsymbol{\varphi}}_m \end{pmatrix}}_{\ddot{\boldsymbol{\varphi}}} + \underbrace{\begin{pmatrix} \boldsymbol{h}_a \\ \boldsymbol{h}_e \\ \boldsymbol{h}_m \end{pmatrix}}_{\boldsymbol{h}} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{\tau}_m \end{pmatrix} \tag{3.74a}$$

$$\boldsymbol{M}_{aa} = \boldsymbol{M}_{aa}(\boldsymbol{\varphi}_a, \boldsymbol{\varphi}_e) \tag{3.74b}$$

$$\boldsymbol{M}_{ae} = \boldsymbol{M}_{ea}^T = \boldsymbol{M}_{ae}(\boldsymbol{\varphi}_a, \boldsymbol{\varphi}_e) \tag{3.74c}$$

$$\boldsymbol{M}_{ee} = \boldsymbol{M}_{ee}(\boldsymbol{\varphi}_a, \boldsymbol{\varphi}_e) \tag{3.74d}$$

$$\boldsymbol{h}_a = \boldsymbol{h}_a(\boldsymbol{\varphi}_a, \dot{\boldsymbol{\varphi}}_a, \boldsymbol{\varphi}_e, \dot{\boldsymbol{\varphi}}_e, \boldsymbol{\varphi}_m, \dot{\boldsymbol{\varphi}}_m) \tag{3.74e}$$

$$\boldsymbol{h}_e = \boldsymbol{h}_e(\boldsymbol{\varphi}_a, \dot{\boldsymbol{\varphi}}_a, \boldsymbol{\varphi}_e, \dot{\boldsymbol{\varphi}}_e) \tag{3.74f}$$

$$\boldsymbol{h}_m = \boldsymbol{h}_m(\boldsymbol{\varphi}_a, \dot{\boldsymbol{\varphi}}_a, \boldsymbol{\varphi}_m, \dot{\boldsymbol{\varphi}}_m) \tag{3.74g}$$

$$\boldsymbol{\Theta} = \mathrm{diag}(\Theta_i) \text{ mit } i = 1, \ldots, n_m \tag{3.74h}$$

$$\boldsymbol{\varphi}_a = \begin{pmatrix} \varphi_{a,1} & \cdots & \varphi_{a,n_a} \end{pmatrix}^T \tag{3.74i}$$

$$\boldsymbol{\varphi}_e = \begin{pmatrix} \varphi_{e,1} & \cdots & \varphi_{e,n_e} \end{pmatrix}^T \tag{3.74j}$$

$$\boldsymbol{\varphi}_m = \begin{pmatrix} \varphi_{m,1} & \cdots & \varphi_{m,n_m} \end{pmatrix}^T \tag{3.74k}$$

$$\boldsymbol{\tau}_m = \begin{pmatrix} \tau_{m,1} & \cdots & \tau_{m,n_m} \end{pmatrix}^T \tag{3.74l}$$

Um die Begrenztheit der berechneten Größen des inversen Modells im Sinne einer BIBO[16]-Stabilität (siehe hierzu z. B. [78] und [79]) zeigen zu können müssen zunächst

[16]BIBO - *Bounded-Input Bounded-Output*. Beschränkte Eingangsgrößen führen immer auf beschränkte Ausgangsgrößen des Systems

einige Voraussetzungen an das Modell und die Eingangsgrößen des inversen Systems gestellt werden. Auch wenn die BIBO-Stabilität ursprünglich ein Stabilitätskriterium für lineare Systeme ist, so kann sie auch für nichtlineare Systeme angewendet werden. Um das inverse System einsetzen zu können muss sichergestellt werden, dass für beschränkte Solltrajektorien der Last am TCP ${}^{L}_{0}\boldsymbol{T}_{soll}(t)$ auch beschränkte Ausgangssignale für die (approximativen) motorseitigen Größen $\hat{\boldsymbol{\tau}}_m, \hat{\boldsymbol{\varphi}}_m$ und $\dot{\hat{\boldsymbol{\varphi}}}_m$ berechnet werden.

Wir setzen daher voraus, dass die Solltrajektorie für den (ideal) starren Roboter über eine inverse Kinematik des starren Roboters in (ideale) abtriebsseitige Starrkörperwinkel $\boldsymbol{\varphi}_{a,r}$ umgerechnet werden können und die sich hieraus ergebenden Winkel $\boldsymbol{\varphi}_{a,r}$ und zeitlichen Ableitungen $\frac{d^i}{dt^i}\boldsymbol{\varphi}_{a,r}, i = 1, 2, \ldots$ stetig sind und in C^∞ liegen, daher beliebig oft differenzierbar sind. Zudem müssen $\boldsymbol{\varphi}_{a,r}$ und alle Ableitungen $\dot{\boldsymbol{\varphi}}_{a,r}, \ddot{\boldsymbol{\varphi}}_{a,r}, \boldsymbol{\varphi}^{(3)}_{a,r}, \ldots$ beschränkt sein (Gl. (3.75)).

$$\|\boldsymbol{\varphi}^{(j)}_{a,r}\| < \infty \text{ für } j = 0, 1, 2, \ldots \tag{3.75}$$

Anm.: Für die praktische Realisierung genügt es, wenn die Winkel dem Index ν nach Abschnitt 3.4 des Systems entsprechend differenzierbar sind.

Zudem muss vorausgesetzt werden, dass alle physikalischen Parameter $\boldsymbol{p} \in \mathbb{R}^+$ des Systems, größer als Null und beschränkt sind. Dies ist für alle physikalisch sinnvollen Systeme gegeben.

Im ersten Schritt werden über die Approximation nach Abschnitt 3.5.2 die approximierten Gelenkmomente $\hat{\boldsymbol{\tau}}_e$ der elastischen Freiheitsgrade nach Gl. (3.76) berechnet. Herzu wird Gl. (3.74) vereinfacht, indem die elastischen Verformungen zu Null gesetzt werden, und die elastisch angekoppelten Antriebsstränge sowie der Term $\boldsymbol{M}_{ee}$ vernachlässigt werden.

$$\underbrace{\boldsymbol{M}_{ea}|_{(\boldsymbol{\varphi}_a \equiv \boldsymbol{\varphi}_{a,r} \wedge \boldsymbol{\varphi}_e \equiv \boldsymbol{0})}}_{\boldsymbol{M}_{ea,r}} \ddot{\boldsymbol{\varphi}}_{a,r} + \underbrace{\boldsymbol{h}^*_e|_{(\boldsymbol{\varphi}_a \equiv \boldsymbol{\varphi}_{a,r} \wedge \boldsymbol{\varphi}_e \equiv \boldsymbol{0})}}_{\boldsymbol{h}_{e,r}} = -\hat{\boldsymbol{\tau}}_e \tag{3.76}$$

Die Eigenschaften der Matrizen $\boldsymbol{M}$ und der Vektoren $\boldsymbol{h}$ von strukturelastischen Robotern wurde in [80] genauer untersucht und Normbegrenzungen der Matrizen und Vektoren hergeleitet (Gl. (3.77)), wobei σ_{M1}, σ_{M2} und σ_{C1} beschränkte Konstanten aus $\mathbb{R}^+$ sind.

$$\sigma_{M1} \leq \|\boldsymbol{M}(\boldsymbol{\varphi})\| \leq \sigma_{M2} < \infty \tag{3.77a}$$

$$\|\boldsymbol{h}(\boldsymbol{\varphi}, \dot{\boldsymbol{\varphi}})\| \leq \sigma_{C1}\|\dot{\boldsymbol{\varphi}}^2\| < \infty \; \forall \|\dot{\boldsymbol{\varphi}}\| < \infty \tag{3.77b}$$

Wie man direkt erkennt ergeben sich für beschränkte und stetige Verläufe von $\boldsymbol{\varphi}_{a,r}$ (und dessen Ableitungen), auch beschränkte, approximierte Gelenkmomente $\hat{\boldsymbol{\tau}}_e$, da sich in den Matrizen $\boldsymbol{M}_{ea,r}$ und $\boldsymbol{h}_{e,r}$, bei welchen es sich lediglich um vereinfachte Teilkomponenten von $\boldsymbol{M}$ und $\boldsymbol{h}$ handelt, nach Gl. (3.77) nur beschränkte Terme befinden, wenn die Parameter $\boldsymbol{p}$ des Systems beschränkt sind.

Wie in Abschnitt 3.5.2 gezeigt, können mit Hilfe der Differentialgleichungen 1. Ordnung (Gl. (3.78)) approximativ, die sich aus den Momenten $\hat{\boldsymbol{\tau}}_e$ resultierenden Verformungen

der Struktur des Roboters bestimmt werden.

$$\hat{\boldsymbol{\tau}}_e = \boldsymbol{K}_e\hat{\boldsymbol{\varphi}}_e + \boldsymbol{D}_e\dot{\hat{\boldsymbol{\varphi}}}_e \tag{3.78a}$$

$$\hat{\boldsymbol{\tau}}_e = \begin{pmatrix} \hat{\tau}_{e,1} & \hat{\tau}_{e,2} & \cdots & \hat{\tau}_{e,n_e} \end{pmatrix} \tag{3.78b}$$

$$\boldsymbol{K}_e = \begin{pmatrix} c_{e,1} & 0 & \ddots & 0 \\ 0 & c_{e,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & c_{e,n_e} \end{pmatrix} \tag{3.78c}$$

$$\boldsymbol{D}_e = \begin{pmatrix} d_{e,1} & 0 & \ddots & 0 \\ 0 & d_{e,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & d_{e,n_e} \end{pmatrix} \tag{3.78d}$$

Anhand von Gl. (3.78) kann man erkennen, dass es sich um n_e lineare, entkoppelte Systeme erster Ordnung handelt. Wie man durch eine einfache Umformung sehen kann (Gl. (3.79)), liegen die Eigenwerte jeweils bei $\frac{-c_{e,i}}{d_{e,i}}$ und sind somit – unter den Voraussetzungen an die Parameter $\boldsymbol{p}$ des Systems – stabil und für beschränkte Gelenkmomente $\hat{\tau}_{e,i}$ sind auch die sich ergebenden, Winkel $\hat{\varphi}_{e,i}$ beschränkt.

$$\dot{\hat{\varphi}}_{e,i} = \frac{-c_{e,i}}{d_{e,i}}\hat{\varphi}_{e,i} + \frac{1}{d_{e,i}}\hat{\tau}_{e,i} \tag{3.79}$$

Die Stabilitätsaussagen können auch auf nichtlineare Federkennlinien $c_{e,i} = c_{e,i}(\hat{\varphi}_{e,i})$ erweitert werden (wie etwa nach Gl. (2.7)), solange die Kennlinie stetig und differenzierbar ist. Für diesen Fall muss eine Stabilität im Sinne von Lyapunov (siehe z. B. [81, 82] und [83]) verwendet werden. Als Lyapunov-Funktion kann hierbei die Stammfunktion der Federenergie gewählt werden. Für die nichtlineare Federkennlinie nach Gl. (2.7) ergibt sich die Lyapunov-Funktion $V(\hat{\varphi}_{e,i})$ nach Gl. (3.80).

$$V(\hat{\varphi}_{e,i}) = \int_0^{\hat{\varphi}_{e,i}} \xi c_{e,i}(\xi)\mathrm{d}\xi \tag{3.80a}$$

$$V(\hat{\varphi}_{e,i}) = c_{0,i}\left(\frac{1}{4}\gamma\hat{\varphi}_{e,i}^4 + \frac{1}{2}\hat{\varphi}_{e,i}^2\right) > 0 \ \forall\hat{\varphi}_{e,i} \neq 0 \tag{3.80b}$$

$$\dot{V}(\hat{\varphi}_{e,i}) = c_{0,i}\left(\gamma\hat{\varphi}_{e,i}^3 + \hat{\varphi}_{e,i}\right)\frac{-c_{e,i}(\hat{\varphi}_{e,i})}{d_{e,i}}\hat{\varphi}_{e,i} \tag{3.80c}$$

$$\dot{V}(\hat{\varphi}_{e,i}) = -\frac{c_{0,i}^2\gamma^2\hat{\varphi}_{e,i}^6}{d_{e,i}} - \frac{2c_{0,i}^2\gamma^2\hat{\varphi}_{e,i}^4}{d_{e,i}} - \frac{c_{0,i}^2\hat{\varphi}_{e,i}^2}{d_{e,i}} < 0 \ \forall\hat{\varphi}_{e,i} \neq 0 \tag{3.80d}$$

Wie in [83] bewiesen wurde, kann für ein nichtlineares System in der Form von Gleichung (3.81) gezeigt werden, dass falls eine stabile Ruhelage im Sinne von Lyapunov existiert auch die Beschränktheit der Ausgangssignale bei beschränkten Eingangssignalen gegeben ist (BIBO-Stabilität). Hierzu müssen folgende Voraussetzungen erfüllt sein:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}) \tag{3.81a}$$

$$\boldsymbol{y} = \boldsymbol{h}_f(t, \boldsymbol{x}, \boldsymbol{u}) \tag{3.81b}$$

$\boldsymbol{x} \in \mathbb{R}^n, u \in \mathbb{R}^p, \boldsymbol{f} : [0,\infty) \times \mathbb{R}^n \times \mathbb{R}^p \Rightarrow \mathbb{R}^n$ ist stetig differenzierbar und $\boldsymbol{h}_f : [0,\infty) \times \mathbb{R}^n \times \mathbb{R}^p \Rightarrow \mathbb{R}^q$ ist kontinuierlich. Das System besitzt eine Ruhelage für $\boldsymbol{x} = 0$ und $\boldsymbol{u} = \boldsymbol{0}$, daher Gl. (3.82) ist erfüllt.

$$\boldsymbol{f}(t, \boldsymbol{0}, \boldsymbol{0}) = \boldsymbol{0} \; \forall t \geq 0 \tag{3.82}$$

Die Jacobimatrizen $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}$ bei $\boldsymbol{u} = \boldsymbol{0}$ sowie $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}$ müssen global beschränkt sein und Gl. (3.83) muss

$$\parallel \boldsymbol{h}_f(t, \boldsymbol{x}, \boldsymbol{u}) \parallel \leq k_1 \parallel \boldsymbol{x} \parallel + k_2 \parallel \boldsymbol{u} \parallel + k_3 \tag{3.83}$$

für frei wählbare, nichtnegative Konstanten k_1, k_2 und k_3 erfüllt sein. Unter diesen Voraussetzungen existieren für alle $\parallel \boldsymbol{x}(0) \parallel \leq \eta$ Konstanten $\gamma > 0$ und $\beta = \beta(\eta, k_3)$, so dass Gleichung (3.84) erfüllt wird.

$$\sup_{t \geq 0} \parallel \boldsymbol{y}(t) \parallel \leq \gamma \sup_{t \geq 0} \parallel \boldsymbol{u}(t) \parallel + \beta \tag{3.84}$$

Diese Bedingungen werden für das Teilsystem nach Gl. (3.79) mit nichtlinearer Federkennlinie $c_{e,i} = c_{e,i}(\hat{\varphi}_{e,i})$ nur dann erfüllt, wenn die Kennlinie beschränkt ist, also gilt $c_{e,i}(\hat{\varphi}_{e,i}) \in (0, c_{max,i}]$. Diese Bedingung ist für die nichtlineare Kennlinie nach Gl. (2.7) nicht direkt erfüllt. Sie kann jedoch erfüllt werden, indem eine stetig differenzierbare Begrenzer-Funktion analog zu Gl. (3.71) auf den nichtlinearen Term der Federkennlinie angewendet wird. Da hierfür eine hohe Grenzen $c_{max,i} > c_{0,i}$ gewählt werden kann, welche im Normalfall nicht erreicht wird spielt dies aus praktischer Sicht keine Rolle. Die Kennlinie ist damit aber stetig und stetig differenzierbar sowie beschränkt, womit auch $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}$ bei $\boldsymbol{u} = \boldsymbol{0}$ global beschränkt ist. Die Lyapunov-Funktion $V(\hat{\varphi}_{e,i})$ wird hierdurch zwar komplizierter (sie ist aufgrund der Länge der Terme hier nicht explizit angegeben), die Anforderungen sind aber immer noch erfüllt. Im Grenzfall geht das System dann über in ein lineares System mit $c_{e,i}(\hat{\varphi}_{e,i}) = c_{max,i}$.

Mit Hilfe der Vorwärtskinematik ${}^L_0\boldsymbol{T}^{(e)}$ des Roboters mit elastischen Gelenken und Strukturfreiheitsgraden kann die neue Position und Orientierung der Last $\mathbf{r}_{L,e}$ des verformten Roboters bestimmt werden. Aus der Differenz zur Solltrajektorie der Last am TCP[17] $\mathbf{r}_{L,r}$ kann der kartesische Fehler $\Delta\boldsymbol{r}$ berechnet werden (Gl. (3.85)). Vereinfachend kann hierbei nur der kartesische Positionsfehler korrigiert werden, wobei dann kleine Orientierungsfehler zugelassen werden. Im allgemeinen Fall enthält $\boldsymbol{r}_{L,e}$ aber sowohl Informationen über die Position und die Orientierung (siehe Gl. (3.57)).

$$\Delta\boldsymbol{r} = \mathbf{r}_{L,e} - \mathbf{r}_{L,r} \tag{3.85}$$

Wie in Abschnitt 3.6.3 beschrieben, können nun mit Hilfe der DLS-Inversen, die entsprechenden Korrekturwinkel $\Delta\boldsymbol{\varphi}_a$ berechnet werden (Gl. (3.86)).

$$\Delta\boldsymbol{\varphi_a} = \left(\boldsymbol{J}_f^T \boldsymbol{J}_f + \lambda^2 \mathbf{I}\right)^{-1} \boldsymbol{J}_f^T \Delta\mathbf{r} \tag{3.86}$$

Die Jacobimatrix $\boldsymbol{J}_f = \boldsymbol{J}_f(\boldsymbol{\varphi}_{a,r}, \hat{\boldsymbol{\varphi}}_e)$ kann für bestimmte Kombinationen von $\boldsymbol{\varphi}_{a,r}$ und $\hat{\boldsymbol{\varphi}}_e$ singulär werden. Wie in [68] hergeleitet, kann die DLS-Inverse auch in der Form von Gl. (3.87) dargestellt werden.

$$\left(\boldsymbol{J}_f^T \boldsymbol{J}_f + \lambda^2 \mathbf{I}\right)^{-1} \boldsymbol{J}_f^T = \boldsymbol{J}_f^T \left(\boldsymbol{J}_f \boldsymbol{J}_f^T + \lambda^2 \boldsymbol{I}\right)^{-1} = \sum_{i=1}^{r} \frac{\sigma_i}{\sigma_i^2 + \lambda^2} \boldsymbol{v}_i \boldsymbol{u}_i^T \tag{3.87}$$

[17] Vereinfachend kann auch, die sich hieraus ergebende, Solltrajektorie des Handwurzelpunktes gewählt werden.

Eine singuläre Jacobimatrix $\boldsymbol{J}_f$ entspricht einem Rangabfall, was bedeutet, dass einige oder alle Singulärwerte σ_i von $\boldsymbol{J}_f$ zu Null werden. Nach Voraussetzung an die Stetigkeit und Begrenztheit der Eingangsgrößen, kann auch die Beschränktheit der Jacobimatrix $\boldsymbol{J}_f$ angenommen werde. Die Singulärwerte σ_i von $\boldsymbol{J}_f$ entsprechen den Wurzeln der Eigenwerte $\sqrt{\lambda_{E,i}}$ der Matrix $\boldsymbol{J}_f^T\boldsymbol{J}_f$ (siehe [84]).

$$\text{eig}\left(\boldsymbol{J}_f^T\boldsymbol{J}_f\right) \Rightarrow \sigma_i = \sqrt{\lambda_{E,i}}\; i = 1, 2, \ldots r \tag{3.88}$$

Nach [85] und [86] kann die Lage der Eigenwerte (Spektrum) einer Matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ mit Elementen a_{ij} über die Vereinigung der Gerschgorin-Kreise $\overline{S}_i$ nach Gl. (3.89) abgeschätzt werden.

$$\overline{S}_i := \overline{S}\left(a_{ii}, \sum_{j=1, j\neq i}^{n} |a_{ij}|\right) \text{ für } i = 1, 2, \ldots, n \tag{3.89a}$$

$$\text{Spektrum von } \boldsymbol{A} \in \bigcup_{\imath=1}^{n} \overline{S}_i \tag{3.89b}$$

Für eine beschränkte Matrix $\boldsymbol{J}_f$ und damit auch beschränkte Matrix $\boldsymbol{J}_f^T\boldsymbol{J}_f$ sind daher auch die Eigenwerte $\lambda_{E,i}$ und damit die Singulärwerte σ_i beschränkt.

Wie man an Gl. (3.65) erkennen kann, wird durch den Dämpfungsterm λ stets eine beschränkte Abweichung $\Delta\mathbf{r}$ auch zu beschränkten Korrekturwinkeln $\Delta\boldsymbol{\varphi}_a$ führen, da nach Definition der Singulärwertzerlegung $\|\boldsymbol{v}_i\| = 1$ und $\|\boldsymbol{u}_i\| = 1$ gilt (siehe [65, 66]).

In der Praxis können diese Winkel, trotz ihrer Beschränktheit, zu groß werden um sinnvoll am realen Roboter aufgeschaltet zu werden (was sehr hohe Gelenkgeschwindigkeiten zur Folge haben würde), weshalb zur zusätzlichen Absicherung eine, bereits in Abschnitt 3.6.3 beschriebene, stetig differenzierbare Begrenzer-Funktion nach Gl. (3.71) verwendet werden kann. Der Ausdruck nach Gl. (3.86) ist zudem stetig differenzierbar, was für die Berechnung der antriebsseitigen Größen entscheidend ist, da die Jacobimatrix $\boldsymbol{J}_f$ aus Ableitungen der Transformationsmatrix ${}_0^L\boldsymbol{T}^{(e)}$ des Roboters gewonnen wird. Transformationsmatrizen des Roboters bestehen aus Verkettungen von trigonometrischen Termen, welche beliebig oft differenzierbar sind.

Mit den berechneten Größen der elastischen Verformungen $\hat{\boldsymbol{\varphi}}_e$ und den, um die Korrektur beaufschlagen, abtriebsseitigen Winkeln $\boldsymbol{\varphi}_{a,e} = \boldsymbol{\varphi}_{a,r} + \Delta\boldsymbol{\varphi}_a$ und den zeitlichen Ableitungen dieser Größen, können nun die (approximativen) motorseitigen Größen d.h. $\hat{\boldsymbol{\tau}}_m, \hat{\boldsymbol{\varphi}}_m, \dot{\hat{\boldsymbol{\varphi}}}_m$ der Antriebsstränge berechnet werden. Hierzu werden die entsprechenden Größen in die Bewegungsgleichungen (3.74) eingesetzt, was auf Gl. (3.90) führt.

$$\boldsymbol{\Theta}\ddot{\hat{\boldsymbol{\varphi}}}_m + \boldsymbol{h}_m|_{\left(\boldsymbol{\varphi}_a \equiv \boldsymbol{\varphi}_{a,e}\right)} = \hat{\boldsymbol{\tau}}_{\boldsymbol{m}} \tag{3.90a}$$

$$\begin{aligned}\boldsymbol{M}_{aa}|_{\left(\boldsymbol{\varphi}_a \equiv \boldsymbol{\varphi}_{a,e} \wedge \boldsymbol{\varphi}_e \equiv \hat{\boldsymbol{\varphi}}_e\right)} \ddot{\boldsymbol{\varphi}}_{a,e} + \boldsymbol{M}_{ae}|_{\left(\boldsymbol{\varphi}_a \equiv \boldsymbol{\varphi}_{a,e} \wedge \boldsymbol{\varphi}_e \equiv \hat{\boldsymbol{\varphi}}_e\right)} \ddot{\hat{\boldsymbol{\varphi}}}_e \\ + \boldsymbol{h}_a|_{\left(\boldsymbol{\varphi}_a \equiv \boldsymbol{\varphi}_{a,e} \wedge \boldsymbol{\varphi}_e \equiv \hat{\boldsymbol{\varphi}}_e\right)} = \mathbf{0}\end{aligned} \tag{3.90b}$$

Das Gleichungssystem (3.90) ist über Indexreduktion nach Abschnitt 3.4 bezüglich der motorseitigen Größen lösbar, wofür allerdings auch höher Ableitungen der der Terme $\boldsymbol{M}_{ae}|_{\left(\boldsymbol{\varphi}_a \equiv \boldsymbol{\varphi}_{a,e} \wedge \boldsymbol{\varphi}_e \equiv \hat{\boldsymbol{\varphi}}_e\right)}$, $\boldsymbol{M}_{aa}|_{\left(\boldsymbol{\varphi}_a \equiv \boldsymbol{\varphi}_{a,e} \wedge \boldsymbol{\varphi}_e \equiv \hat{\boldsymbol{\varphi}}_e\right)}$, $\boldsymbol{h}_a|_{\left(\boldsymbol{\varphi}_a \equiv \boldsymbol{\varphi}_{a,e} \wedge \boldsymbol{\varphi}_e \equiv \hat{\boldsymbol{\varphi}}_e\right)}$, $\boldsymbol{h}_m|_{\left(\boldsymbol{\varphi}_a \equiv \boldsymbol{\varphi}_{a,e} \wedge \boldsymbol{\varphi}_e \equiv \hat{\boldsymbol{\varphi}}_e\right)}$ aus Gl. (3.90) sowie von $\boldsymbol{M}_{ea,r}$, $\boldsymbol{h}_{e,r}$ aus der Approximation der Verformungen nach

Gl. (3.76) und Ableitungen von $\boldsymbol{J}_f$, $\Delta\boldsymbol{r}$ und λ (für die Kompensation der Verformungen) sowie den darin vorkommenden Winkeln, benötigt werden. Nach den Voraussetzungen an die Modelle sind diese Terme alle stetig differenzierbar (die ist der Grund für die Approximationen in Kapitel 2 dieser Arbeit und die Filterung von σ_{min}), so dass diese Ableitungen symbolisch berechenbar sind.

Zusätzlich zu den Gleichungen (3.90), kann auch die Motordynamik nach Gl. (2.2) berücksichtigt werden, wodurch sich die Ordnung der Differentiation um eins erhöht.

Zur Modellierung ist es hilfreich das abtriebsseitige Schnittmoment zwischen Antriebssträngen und elastischer, kompensierter Kinematik $\hat{\boldsymbol{\tau}}_{a,e}$ einzuführen welches sich ergibt wenn in $\boldsymbol{h}_a$ die motorseitigen Winkel mit den korrigierten abtriebsseitigen Winkeln $\boldsymbol{\varphi}_{a,e}$ gleichgesetzt werden. Es kann über Gl. (3.91) berechnet werden. In *Modelica* ist dies direkt über relative Sensorelemente zwischen kompensierter Kinematik und den Modulen der Antriebsstränge berechenbar, und ermöglicht es, dass die inversen Antriebsstränge einfach austauschbar sind.

$$\begin{pmatrix} \boldsymbol{M}_{aa}|_{(\boldsymbol{\varphi}_a \equiv \boldsymbol{\varphi}_{a,e} \wedge \boldsymbol{\varphi}_e \equiv \hat{\boldsymbol{\varphi}}_e)} & \boldsymbol{M}_{ae}|_{(\boldsymbol{\varphi}_a \equiv \boldsymbol{\varphi}_{a,e} \wedge \boldsymbol{\varphi}_e \equiv \hat{\boldsymbol{\varphi}}_e)} \end{pmatrix} \begin{pmatrix} \ddot{\boldsymbol{\varphi}}_{a,e} \\ \ddot{\hat{\boldsymbol{\varphi}}}_e \end{pmatrix} + \boldsymbol{h}_a|_{(\boldsymbol{\varphi}_m \equiv \varphi_{a,e} \wedge \boldsymbol{\varphi}_a \equiv \boldsymbol{\varphi}_{a,e} \wedge \boldsymbol{\varphi}_e \equiv \hat{\boldsymbol{\varphi}}_e)} = \hat{\boldsymbol{\tau}}_{a,e} \tag{3.91}$$

Auch bei dieser Berechnung führen beschränkte Verläufe von $\boldsymbol{\varphi}_{a,e}$ und $\hat{\boldsymbol{\varphi}}_e$ sowie deren Ableitungen und den Voraussetzungen an die Parameter $\boldsymbol{p}$ des Systems zu beschränkten Verläufen von $\hat{\boldsymbol{\tau}}_{a,e}$.

Mit Hilfe von $\hat{\boldsymbol{\tau}}_{a,e}$ sowie $\boldsymbol{\varphi}_{a,e}$ kann nun die inverse Antriebsstrangdynamik, bezüglich der motorseitigen Größen, gelöst werden.

Hierzu werden ebenfalls höhere Ableitungen von $\boldsymbol{\varphi}_{a,e}$ und $\hat{\boldsymbol{\tau}}_{a,e}$ benötigt, weshalb auch hierbei die Bewegungsgleichung (3.91) differenziert werden müssen.

Wie bereits in [10] gezeigt wurde, ist die inverse Dynamik der Antriebsstränge für stetig differenzierbare, nichtlineare Funktion der Reibung, der Getriebesteifigkeit und des Wirkungsgrades mit einer Obergrenze der Steigung möglich d.h. Approximationen wie sie in Kapitel 2 beschrieben wurden, wenn im Falle von unbeschränkten Steigung eine stetig differenzierbare Begrenzer-Funktion nach Gl. (3.71) verwendet wird.

Die inverse Antriebsstrangdynamik lässt sich für jeden der $i = 1, \ldots, n_m$ Antriebe über eine nichtlineare Zustandsgleichung darstellen (diese kann über die Indexreduktion hergeleitet werden).

Ein- und Ausgangsgrößen sowie Zustände für alle n_m dieser Komponente können nach Gl. (3.92) gewählt werden[18] (der Index für die einzelnen Antriebsstränge der Achsen wird ab hier weggelassen).

$$\boldsymbol{u} = \begin{pmatrix} u_1 & u_2 & u_3 & u_4 & u_5 \end{pmatrix}^T = \begin{pmatrix} \varphi_{a,e} & \dot{\varphi}_{a,e} & \ddot{\varphi}_{a,e} & \hat{\tau}_{a,e} & \dot{\hat{\tau}}_{a,e} \end{pmatrix}^T \tag{3.92a}$$

$$x = \hat{\varphi}_m \tag{3.92b}$$

$$\boldsymbol{y} = \begin{pmatrix} y_1 & y_2 & y_3 & y_4 \end{pmatrix}^T = \begin{pmatrix} \hat{\varphi}_m & \dot{\hat{\varphi}}_m & \ddot{\hat{\varphi}}_m & \hat{\tau}_m \end{pmatrix}^T \tag{3.92c}$$

Die Gleichungen ergeben sich im allgemeinen Fall nach Gl. (3.93) wenn lineare Dämpfung (Parameter d) und eine stetig differenzierbare Approximation für das Reibmoment τ_{reib}

[18] Alternativ kann auch der Relativwinkel $(\varphi_{a,e} - \hat{\varphi}_m)$ als Zustand gewählt werden, was numerisch günstiger sein kann.

und das Getriebemoment τ_{getr} angenommen wird.

$$f(t, \boldsymbol{u}, x) = \dot{x} = \frac{1}{d}\left(u_4 + \tau_{getr}\right) + u_2 \tag{3.93a}$$
$$h_{f,1}(t, x, \boldsymbol{u}) = y_1 = x \tag{3.93b}$$
$$h_{f,2}(t, x, \boldsymbol{u}) = y_2 = \dot{x} \tag{3.93c}$$
$$h_{f,3}(t, x, \boldsymbol{u}) = y_3 = \frac{1}{d}\left(u_5 + \dot{\tau}_{getr}\right) + u_3 \tag{3.93d}$$
$$h_{f,4}(t, x, \boldsymbol{u}) = y_4 = \Theta y_3 - \tau_{getr} - d(u_2 - \dot{x}) + \tau_{reib} \tag{3.93e}$$

Für eine nichtlineare Getriebesteifigkeit nach Gl. (2.9) und eine Reibungsmodellierung nach Gl. (2.3), welche die Bedingungen erfüllen, können die motorseitigen Größen (für jeden Antrieb $i = 1, \ldots, n_m$) über Gl. (3.93) direkt bestimmt werden, die benötigten Terme sind hierzu in Gl. (3.94) dargestellt (analog können auch komplexere Antriebsstrangmodelle, sofern die Gleichungen stetig differenzierbar und eindeutig umkehrbar sind, per Indexreduktion invertiert werden, was jedoch auf längliche Gleichungen führt, weshalb sie hier nicht dargestellt sind).

$$\begin{aligned}\tau_{getr,\gamma,lim} = \big(&c_\gamma - c_\gamma(\operatorname{atan2}((c_\gamma - c_{max})s)\frac{1}{\pi} + 1 - \operatorname{atan2}(c_\gamma s)\frac{1}{\pi}) \\ &+ c_{max}(\operatorname{atan2}((c_\gamma - c_{max})s)\frac{1}{\pi} + \frac{1}{2})\big)\varphi_{rel}\end{aligned} \tag{3.94a}$$

$$\begin{aligned}\dot{\tau}_{getr,\gamma,lim} = \big(&2c_0\gamma\varphi_{rel}\dot{\varphi}_{rel} - 2c_0\gamma\varphi_{rel}\dot{\varphi}_{rel}(\operatorname{atan2}((c_\gamma - c_{max})s)\frac{1}{\pi} + 1 \\ &- \operatorname{atan2}(c_\gamma s)\frac{1}{\pi}) - c_\gamma(2c_0\gamma\varphi_{rel}\dot{\varphi}_{rel}s/(1 + (c_\gamma - c_{max})^2 s^2)\frac{1}{\pi} \\ &- 2c_0\gamma\varphi_{rel}\dot{\varphi}_{rel}s/(1 + c_0^2(1 + \gamma\varphi_{rel}^2)^2 s^2)\frac{1}{\pi}) \\ &+ 2c_{max}c_0\gamma\varphi_{rel}\dot{\varphi}_{rel}s/(1 + (c_\gamma - c_{max})^2 s^2)\frac{1}{\pi})\varphi_{rel} \\ &+ (c_\gamma - c_\gamma(\operatorname{atan2}((c_\gamma - c_{max})s)\frac{1}{\pi} + 1 - \operatorname{atan2}(c_\gamma s)\frac{1}{\pi}) \\ &+ c_{max}(\operatorname{atan2}((c_\gamma - c_{max})s)\frac{1}{\pi} + \frac{1}{2})\big)\dot{\varphi}_{rel}\end{aligned} \tag{3.94b}$$

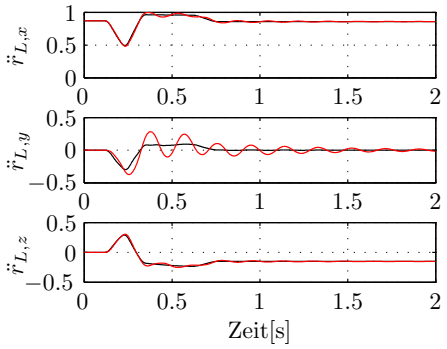
$$c_\gamma = c_0(1 + \gamma(\varphi_{rel}^2)) \tag{3.94c}$$
$$\varphi_{rel} = u_1 - x \tag{3.94d}$$
$$\dot{\varphi}_{rel} = u_2 - \dot{x} \tag{3.94e}$$
$$\tau_{reib,exp} = \tau_v\dot{x} + \frac{2\tau_h}{1 + e^{-s\dot{x}}} - \tau_h \tag{3.94f}$$

Wie in [10] anhand einer Input-to-State Analyse (siehe [87]) gezeigt wurde, führt die Inversion des Antriebsstrangs auf beschränkte Ergebnisse (bei beschränkten Eingängen), wenn die Parameter $\boldsymbol{p}$ die getroffenen Voraussetzungen erfüllen und sowohl die Reibkennlinie als auch die Federkennlinie durch den Ursprung verlaufen und die Federkennlinie (unbegrenzt) streng monoton steigt und eine maximale Steigung besitzt. Dies ist durch die verwendeten Approximationen erfüllt, wenn im Falle einer Kennlinie mit unbegrenzter Steigung die Kennlinie durch eine stetig differenzierbare Begrenzer-Funktion nach Gl. (3.71) modifiziert wird.

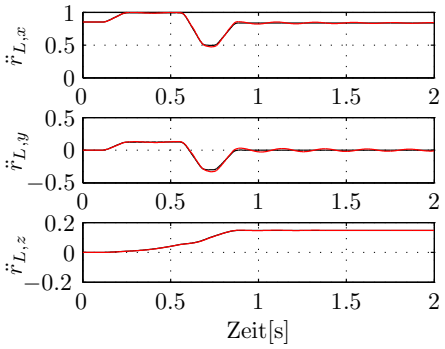
3.8.3 Untersuchung des inversen Modells des Roboters

Für die Roboter KR210 und KR16 wurde jeweils ein approximatives strukturelastisches inverses Modell (ASIM) basierend auf dem in Abschnitt 3.8.2 vorgestellten Verfahren aufgebaut. Das inverse Modell des KR210 soll hier stellvertretend vorgestellt werden. Analoge Ergebnisse lassen sich auch für den KR16 zeigen.

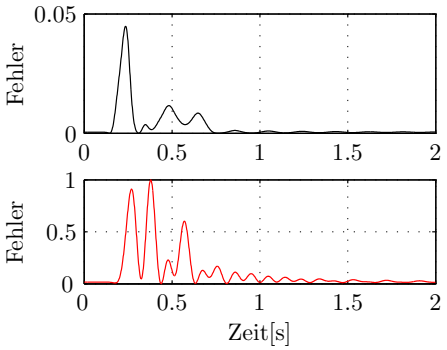
Die Parameter des Modells werden mit Hilfe des in Kapitel 4 vorgestellten Verfahrens bestimmt.



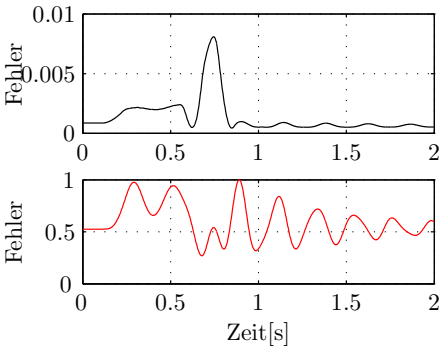
(a) Vergleich der translatorischen Komponenten von $\ddot{\mathbf{r}}_{L,t}$ in der Kanonenstellung (normiert auf Maximalwert) über der Zeit in s. (Mit Strukturstеifigkeit: Schwarz, ohne: Rot)



(b) Vergleich der translatorischen Komponenten von $\ddot{\mathbf{r}}_{L,t}$ in der Strecklage (normiert auf Maximalwert) über der Zeit in s. (Mit Strukturstеifigkeit: Schwarz, ohne: Rot)



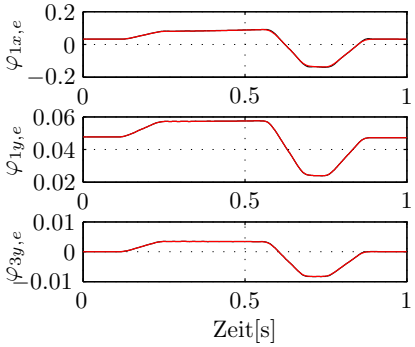
(c) Quadratische Abweichung von $\mathbf{r}_{L,t}$ vom Sollwert in der Kanonenstellung (normiert auf Maximalwert) über der Zeit in s. (Mit Strukturstеifigkeit: Schwarz, ohne: Rot)



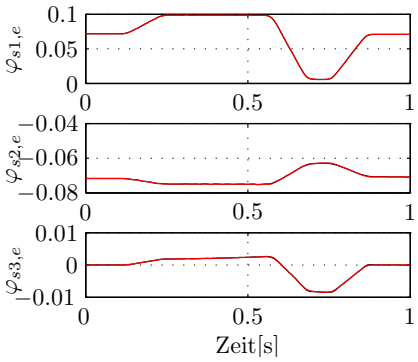
(d) Quadratische Abweichung von $\mathbf{r}_{L,t}$ vom Sollwert in der Strecklage (normiert auf Maximalwert) über der Zeit in s. (Mit Strukturstеifigkeit: Schwarz, ohne: Rot)

Abbildung 3.22: Einfluss der Strukturelastizität auf das dynamisch Verhalten der Strecke für den KR210. Gezeigt ist der Vergleich der translatorischen Beschleunigung $\ddot{\mathbf{r}}_{L,t}$ der Last der Strecke bei Verwendung eines inversen Modells als Vorsteuerung. Gezeigt sind der Vergleich bei Berücksichtigung der Strukturelastizitäten im inversem Modell und ohne Berücksichtigung für eine 5° Dreichachsbewegungen des Roboter in Strecklage und Kanonenstellung, sowie die (quadratische) Abweichung der Last der Strecke von Sollwert. Der Einfluss der Strukturelastizitäten auf das dynamische Verhalten ist stark stellungsabhängig. Durch eine Berücksichtigung ist jedoch eine deutliche Reduzierung der Schwingungen am TPC möglich.

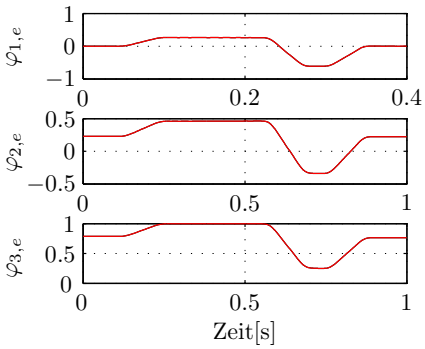
Im Folgenden werden einige Simulationsergebnisse für dieses approximative inverse Modell gezeigt, wobei der Regler vernachlässigt wird und angenommen wird, dass das



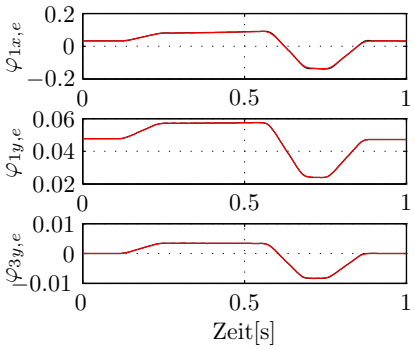
(a) Vergleich der berechneten relativen Verdrehwinkel, für die Kippfreiheitsgrade, von inversem Modell (rot) und der gesteuerten Strecke (schwarz), normiert bezüglich des max. Verdrehwinkel über der Zeit in s. Anregung: Simultane 5° Bewegung der drei Grundachsen aus der Strecklage.



(b) Vergleich der berechneten relativen Verdrehwinkel, für die Freiheitsgrade der Schwinge, von inversem Modell (rot) und der gesteuerten Strecke (schwarz), normiert bezüglich des max. Verdrehwinkel über der Zeit in s. Anregung: Simultane 5° Bewegung der drei Grundachsen aus der Strecklage.

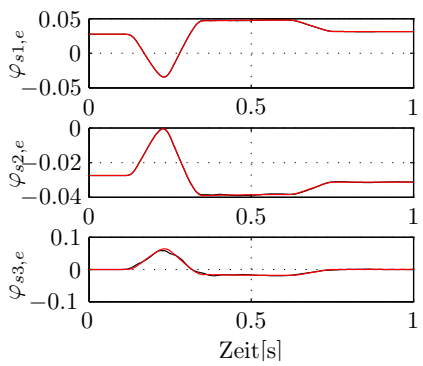


(c) Vergleich der berechneten relativen Verdrehwinkel, für die Grundachsen des Roboters, von inversem Modell (rot) und der gesteuerten Strecke (schwarz), normiert bezüglich des max. Verdrehwinkel über der Zeit in s. Anregung: Simultane 5° Bewegung der drei Grundachsen aus der Strecklage.

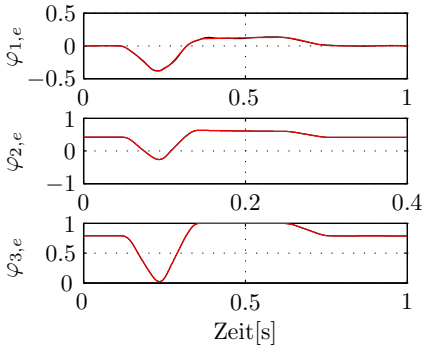


(d) Vergleich der berechneten relativen Verdrehwinkel, für die Kippfreiheitsgrade, von inversem Modell (rot) und der gesteuerten Strecke (schwarz), normiert bezüglich des max. Verdrehwinkel über der Zeit in s. Anregung: Simultane 5° Bewegung der drei Grundachsen aus der Kanonenstellung.

Abbildung 3.23: Ergebnisse der Simulation mit dem approximativen inversen Modell. Gezeigt sind die berechneten Vorsteuergrößen sowie entsprechende Größen der vorgesteuerten Strecke für eine simultane 5° Bewegung der drei Grundachsen des Roboters aus der Strecklage und der Kanonenstellung. Die Daten sind jeweils normiert auf den max. Verdrehwinkel. Die in der Simulation durch die Approximation berechneten Größen entsprechen nahezu exakt den sich in der Strecke ergebenden Verformungen der Struktur.



(a) Vergleich der berechneten relativen Verdrehwinkel, für die Freiheitsgrade der Schwinge, von inversem Modell (rot) und der gesteuerten Strecke (schwarz), normiert bezüglich des max. Verdrehwinkel über der Zeit in s. Anregung: Simultane 5° Bewegung der drei Grundachsen aus der Kanonenstellung.



(b) Vergleich der berechneten relativen Verdrehwinkel, für die Grundachsen des Roboters, von inversem Modell (rot) und der gesteuerten Strecke (schwarz), normiert bezüglich des max. Verdrehwinkel über der Zeit in s. Anregung: Simultane 5° Bewegung der drei Grundachsen aus der Kanonenstellung.

Abbildung 3.24: Ergebnisse der Simulation mit dem approximativen inversen Modell des KR210. Gezeigt sind die berechneten Vorsteuergrößen sowie entsprechende Größen der vorgesteuerten Strecke für eine simultane 5° Bewegung der drei Grundachsen des Roboters aus der Kanonenstellung. Die Daten sind normiert auf den max. Verdrehwinkel. Auch in dieser Simulation führt die Approximation zu einer sehr guten Übereinstimmung bei den berechneten Verformungen.

Streckenmodell der Ausgangsgrößen $\hat{\boldsymbol{\varphi}}_m(t)$ des inversen Modells exakt folgt (perfekter Lageregler). Das Streckenmodell entspricht dabei dem Modell nach Abschnitt 4.5.1. Zudem sind alle Parameter als bekannt angenommen (daher identisch im inversen Modell). Als Last wurde dabei eine Hantel mit $210kg$ gewählt.

Abbildung 3.23 und 3.24 zeigen einige Ergebnisse der Approximation für eine simultane 5° Bewegung der drei Grundachsen des Roboters aus der Strecklage und Kanonenstellung.

Den Abbildungen kann man entnehmen, dass die Approximation über ein Parallelmodell die Verformungen gut wiedergeben kann.

Auch wenn die Verformungen - durch die relativ steife Ausführung der Strukturteile des Roboters - klein sind, so führt eine Vernachlässigung doch zu einem deutlich stärkeren Schwingungsverhalten der Last. In Abbildung 3.22 kann man dies erkennen. In den Abbildungen wird die translatorische Beschleunigung der Last $\ddot{\mathbf{r}}_{L,t}(t)$ der Strecke verglichen, wenn im inversen Modell Strukturelastizitäten vernachlässigt werden (bei ansonsten identischem Modell inklusive elastischen Antriebssträngen).

Kapitel 4

Identifikation von strukturelastischen Robotern

Ziel der Identifikation ist es, alle Parameter für die erstellten Modelle so zu bestimmen, dass die Modelle den realen Roboter möglichst genau abbilden (für die jeweils relevanten Eigenschaften). Aus den so gewonnen Modellen werden anschließend inverse Modelle abgeleitet, welche für eine Steuerung einsetzbar sind.

In der Literatur existiert eine Vielzahl von Identifikationsmöglichkeiten (siehe z. B. [88, 89, 90, 91, 92, 93, 94]). Häufig sind die Ansätze jedoch auf lineare Modelle (z. B. Methoden im Frequenzbereich) beschränkt. Vorhandene Methoden müssen an die Problemstellung angepasst und erweitert werden um ein möglichst gutes Identifikationsergebnis zu erzielen.

4.1 Auswahl der Identifikationsmethode

Die in Kapitel 2 vorgestellten Modellkomponenten dienen als Grundlage für die Identifikation. Diese Modelle besitzen folgende Eigenschaften:

- Nichtlinear
- Parametrisch
- Definiert im Zeitbereich

Diese Eigenschaften sind bei der Identifikation zu berücksichtigen.

Methoden im Frequenzbereich können nur auf lineare Systeme sinnvoll angewendet werden. Wie in [12] gezeigt wurde, ist eine Identifikation nichtlinearer Modelle im Frequenzbereich, durch eine Linearisierung in verschiedenen Arbeitspunkten, möglich. Stark nichtlineare Effekte wie Haftreibung oder Änderungen von (nichtlinearen) Steifigkeiten – welche etwa beim Getriebe messbar sind – können hierdurch allerdings nur unzureichend bestimmt werden. Es wurde daher eine Identifikation im Zeitbereich gewählt.

Kritisch ist auch die hohe Anzahl der Parameter der Modelle, die mit dem Umfang der Komplexität der Modelle stark ansteigt.

Ein effektiver Ansatz ist es daher, möglichst viele Parameter durch direkte Messungen einzeln zu ermitteln, solange diese getrennt messbar sind. Durch die elastische Struktur des Roboters ist dies allerdings nur sehr begrenzt möglich. Parameter des Getriebes von Achse 3 des Roboters können so z. B. nur sehr schwer getrennt von der Steifigkeit der Schwinge und des Getriebes von Achse 2 identifiziert werden. Denn eine entsprechende Anregung des Getriebes von Achse 3, die etwa durch eine kurze (möglichst sprungförmige) Bewegung realisiert werden kann, regt immer auch die Steifigkeiten der anderen Bauteile an.

Verhindert werden kann dies nur durch den Ausbau und einzelnes Vermessen der entsprechenden Komponenten. Dies bedeutet allerdings einen sehr großen Aufwand, zudem ändert sich das dynamische Verhalten der Komponenten nicht unerheblich, wenn sie im Roboter verbaut sind. Für schwer direkt messbare Größen bietet sich daher eine

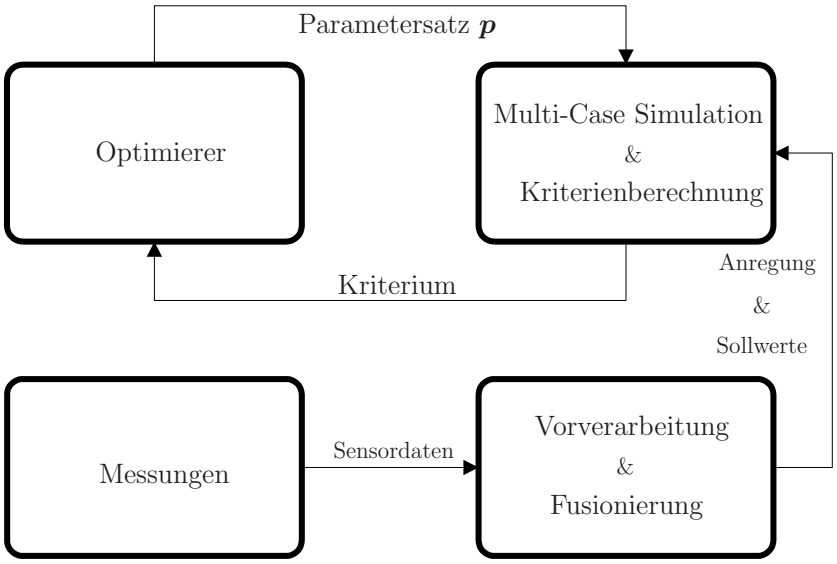


Abbildung 4.1: Ablauf der Parameteridentifikation durch Optimierung.

Identifikation über eine Optimierung an. Hierbei werden die Parameter der Modelle von einem geeigneten Optimierungsalgorithmus variiert, um eine Gütefunktion zu minimieren. Dieses Vorgehen hat den Vorteil, dass mehrere Parameter gleichzeitig bestimmt werden können und nichtlineare Modelle verwendet werden können. Nachteilig ist jedoch, dass das Optimierungsproblem hochgradig nichtlinear ist und nur sehr schwer ein globales Minimum gefunden werden kann, wenn eine große Anzahl von Parametern bestimmt werden soll.

Den grundlegenden Ablauf der Identifikation durch Optimierung zeigt Abbildung 4.1. Messgrößen werden dabei nach einer Vorverarbeitung mit entsprechenden Größen der Simulation über eine Gütefunktion (Kriterienberechnung) verglichen. Der Optimierer variiert dann entsprechend dieser Kriterien die Parameter des Modells, bis die gewünschten Kriterien erreicht sind, bzw. ein Minimum der Gütefunktion gefunden wurde.

4.2 Messungen und Strategien zur Identifikation

Die Parameter in den Modellen können in drei Kategorien eingeteilt werden:

- Statische & Konstruktions-Parameter $\boldsymbol{p}_{stat}$:
  Diese können direkt aus den CAD-Daten des Roboters ermittelt werden. Beispiele sind Längen, Massen und Trägheitsmomente.
- Quasi-statisch, direkt messbare Parameter $\boldsymbol{p}_{q-stat}$:
  Diese Parameter können ohne eine (starke) dynamische Anregung des Roboters bestimmt werden. Hierzu zählen Reibung und Wirkungsgrade.
- Dynamisch relevante Parameter $\boldsymbol{p}_{dyn}$:
  Diese Parameter sind entscheidend für das dynamische Verhalten des Roboters und können nur schwer voneinander entkoppelt bestimmt werden. Eine starke dynamische Anregung ist hilfreich um diese Art der Parameter zu bestimmen. Beispiele sind Steifigkeiten und Dämpfungen.

Parameter des Roboters vom Typ $\boldsymbol{p}_{stat}$ sind alle Massen und Längen des Roboters, sowie die Trägheiten der einzelnen Komponenten. Diese Parameter können ohne eine Anregung des Roboters identifiziert werden und können aus CAD-Daten der Bauteile sowie durch einfache Messungen direkt bestimmt werden. Im Fall vom KR16 und KR210 können diese Daten den Spezifikationen entnommen werden.

Parameter des Typs $\boldsymbol{p}_{q-stat}$ (Kennlinien-Parameter) wie Reibung, Vorspannung und Federstärke des Gewichtsausgleichs können ohne hochdynamische Anregung ermittelt werden. Dadurch können sie leichter getrennt von anderen Parametern des Roboters ermittelt werden.

Die am schwersten zu ermittelnden Parameter des Roboters gehören zur Kategorie der dynamisch relevanten Parameter $\boldsymbol{p}_{dyn}$. Hierbei handelt es sich um die Parameter der Steifigkeiten und Dämpfungen der Achsen und der Struktur des Roboters. Eine Möglichkeit diese Parameter identifizieren zu können ist es, stoßartige Bewegungen mit dem Roboter durchzuführen (typischerweise kurze Bewegungen der entsprechenden Achsen), um dann das Ausschwingverhalten des Roboters zu analysieren. Theoretisch wären auch Anregungen von außen denkbar, wie etwa ein Stoß mit einem Hammer. Diese Form der Anregung ist allerdings, ohne großen technischen Aufwand, nicht exakt reproduzierbar. Erschwerend kommt hinzu, dass der Roboter, je nach Stellung der Achsen, ein deutlich unterschiedliches dynamisches Verhalten aufweist. Zur Identifikation der Parameter von Typ $\boldsymbol{p}_{dyn}$ bietet sich eine Grey-Box-Identifikation[1] über eine Optimierung an, welche durchgeführt wird, nachdem bereits die Parameter von Typ $\boldsymbol{p}_{stat}$ und $\boldsymbol{p}_{q-stat}$ ermittelt sind. Hierdurch ist es möglich die Parameter Anzahl deutlich zu reduzieren und damit die Optimierung zu vereinfachen.

Hierdurch entsteht ein mehrstufiges Identifikationskonzept, welches aus mehreren Teilschritten besteht, welche im Folgenden beschrieben werden sollen.

[1] Hier wurde der Begriff „Grey-Box“ gewählt, da zwar die Struktur der Modelle bekannt ist, diese allerdings nur approximativ der Realität entspricht und damit die Parameter unter Umständen nicht komplett physikalisch interpretierbar sind. In der Literatur wird hierfür auch teilweise der Begriff „White-Box“ genannt, die Übergänge sind jedoch fließend.

Für die Identifikation der Parameter ist zudem entscheidend, welche Sensoren verfügbar sind. Der Roboter selbst besitzt aus Kostengründen nur sehr wenig Sensorik:

- Messung des Iststroms $\boldsymbol{I}_m$ der Motoren der 6 Achsen
- Messung der Motorwinkel $\boldsymbol{\varphi}_m$ der 6 Achsen
- Messung der Motortemperatur $\boldsymbol{T}_m$ der 6 Motoren

Zusätzlich wurden für die Messungen folgende externe Sensoren eingesetzt:

- 3D-Beschleunigungssensoren
- Drehratensensoren
- 6D-Kamera[2] zur Positions- und Orientierungsmessung von Markern

Aus der Messung des Iststroms $\boldsymbol{I}_m$ der einzelnen Achsen können über eine statische Kennlinie die Motormomente $\boldsymbol{\tau}_m$ bestimmt werden, unter Vernachlässigung der – im Vergleich zur Mechanik sehr schnellen – Motordynamik. Soll die Motordynamik berücksichtigt werden kann Gl. (2.2) verwendet werden.

4.2.1 Bestimmung der Reibung und des Haltemoments des Gewichtsausgleichs (GWA)

Zunächst werden Messungen zur Bestimmung quasi-statisch direkt messbare Parameter $\boldsymbol{p}_{q-stat}$ durchgeführt. Die Reibung kann durch Messungen ermittelt werden, bei denen die Geschwindigkeit der entsprechenden Achse konstant gehalten wird. Nach dem Beschleunigungsvorgang und einer gewissen Abklingzeit, kann die Reibung dann getrennt von den elastischen Parametern ermittelt werden. Hierzu eignen sich besonders Trajektorien, welche die mechanischen Schwingungen möglichst wenig anregen. Dies kann durch sehr stark (Tiefpass-) gefilterte Trajektorien erreicht werden. Insbesondere die (stellungsabhängigen) Resonanzfrequenzen des Roboters dürfen hierbei nicht angeregt werden.

Um die Haftreibung zu messen ist es notwendig, den Motorstrom langsam zu erhöhen, bis sich die entsprechende Achse bewegt. Dies ist jedoch nur für eine ungeregelte Achse möglich. Bei Achsen, auf welche die Schwerkraft wirkt, ist dies nicht möglich, da bereits ein Haltemoment aufgebracht werden muss, welches nicht perfekt eingestellt werden kann. Es ist daher meist nur möglich eine Näherung für die Haftreibung zu ermitteln, welche durch die Messung des Achsmoments bei extrem geringen Geschwindigkeiten bestimmt werden kann. Einflüsse von Schwerkraft und Trägheiten müssen dann über ein Modell (für langsame Geschwindigkeiten genügt ein Starrkörpermodell des Roboters) herausgerechnet werden, um zu dem Reibmoment zu gelangen.

[2] 6D-Kamera: Über drei CCD-Kameras kann die Position und Orientierung von Infrarot-Dioden, welche am Roboter angebracht sind, gemessen werden.

Alternativ können die Antriebsstränge über einen zusätzlichen Prüfstand, welcher lediglich aus Motor oder Motor und Getriebe besteht, getrennt vermessen werden. Dadurch können Reibungen und Motorzeitkonstanten direkt ermittelt werden.

Da die Reibung stark geschwindigkeitsabhängig ist, müssen verschiedene Messungen für unterschiedliche Geschwindigkeiten durchgeführt werden. Geeignet sind hierbei Fahrten mit konstanter Geschwindigkeit, da bei diesen alle Terme in den Bewegungsgleichungen – vorausgesetzt alle Schwingungen sind abgeklungen – bis auf die Gravitation und Reibung keinen oder nur geringen Einfluss haben. Dies muss für alle Achsen einzeln durchgeführt werden. Damit überprüft werden kann, ob wirklich alle Getriebeschwingungen abgeklungen sind, wenn die Reibung bestimmt werden soll, werden zusätzlich Drehratensensoren an den Strukturteilen direkt hinter der jeweiligen Achse angebracht. Getriebe- und Strukturschwingungen sind in den Messungen unerwünscht, da dann aufwendigere Modelle für die Identifikation von $\boldsymbol{p}_{q-stat}$ benötigt werden würden.

Soll zudem noch die Temperaturabhängigkeit der Reibung modelliert werden, müssen zusätzlich für verschiedene Temperaturen (z. B. 1°C Raster) Messungen in unterschiedlichen Geschwindigkeiten vorgenommen werden. Dies bedeutet bei 6 Achsen des Roboters einen erheblichen Aufwand. Die Temperaturen müssen entweder durch eine Klimakammer oder durch Warmfahrten (durch schnelle zyklische Bewegungen) generiert werden. Bei einem feinen Temperaturraster ist der Aufwand für diese Messungen für industrielle Applikationen ungeeignet. Reibmessungen wurden daher nur für die Standardbetriebstemperatur durchgeführt.

Bei den Messungen zur Bestimmung der Reibung werden am Roboter Iststrom $\boldsymbol{I}_m$ und Motorposition $\boldsymbol{\varphi}_m$ aufgezeichnet. In der Simulation wird dann ein Starrkörpermodell verwendet mit den Winkeln $\boldsymbol{\varphi}_m$ als Eingangsgröße. Bei diesem motorseitigen inversen Starrkörpermodell $\left(\boldsymbol{\Sigma}_{mot}^{m,(r)}\right)^{-1}$ sind die Motormoment $\hat{\boldsymbol{\tau}}_m$ Ausgangsgrößen. Das Modell für den Antriebsstrang besteht dabei lediglich aus einer Motorträgheit (umgerechnet auf die Abtriebsseite des Getriebes) und dem Reibmodell, dessen Parameter optimiert werden sollen. Über eine statische Kennlinie kann der Motorstrom in ein Motormoment umgerechnet werden. Nur bei sehr geringer Motorgeschwindigkeit, muss ein dynamisches Motormodell verwendet werden (Ausgang des Modells ist dann $\boldsymbol{I}_m$).

Als Gütefunktion $\mathfrak{G}$ für die Optimierung der Reibung dient Gleichung (4.1).

$$\mathfrak{G}_{A,i} = \begin{cases} (I_{m,i}k_{I\tau} - \hat{\tau}_{m,i})^2, & \text{wenn } |\dot{\varphi}_{m,ist,i}| > \dot{\varphi}_{m,i,min} \wedge t > t_{min} \\ 0, & \text{sonst} \end{cases} \tag{4.1a}$$

$$\mathfrak{G} = \int_0^{t_{end}} \left(\sum_{i=1}^{6} \mathfrak{G}_{A,i} \right) dt \tag{4.1b}$$

Hierin steht $\mathfrak{G}_{A,i}$ für das Gütekriterium für die i einzelnen Achsen. $\dot{\varphi}_{m,i,min}$ ist die minimale Geschwindigkeit ab der ein Kriterium gebildet wird (um die Beschleunigungsphase und nicht aktive Achsen herauszufiltern), t_{min} ist die Startzeit ab der ein Kriterium gebildet wird, um Einschwingvorgänge nicht berücksichtigen zu müssen. Die Integration dient dazu einen einzelnen Wert der Gütefunktion für den Optimierer zu bestimmen. Die Endzeit der Simulation, bzw. der Messung ist mit t_{end} bezeichnet.

Die einzelnen Geschwindigkeiten werden dabei als unterschiedliche *Cases*[3] für die Optimierung verwendet. Optimiert wird letztlich die Summe der einzelnen Kriterien $\mathfrak{G}$ über alle *Cases*.

Die Vorspannung des Gewichtsausgleichs kann direkt aus dem Haltemoment der zweiten Achse berechnet werden. Die (stellungsabhängige) Kraft der Feder durch langsames (damit die Dynamik des Roboters nicht angeregt wird) Verfahren der Achse 2 und Messung des Moments von Achse 2. Auch hierbei müssen wieder (über Modelle) der Einfluss der Schwerkraft, Reibung und der Trägheit kompensiert werden.

4.2.2 Grey-Box-Identifikation der Steifigkeiten und Dämpfungen

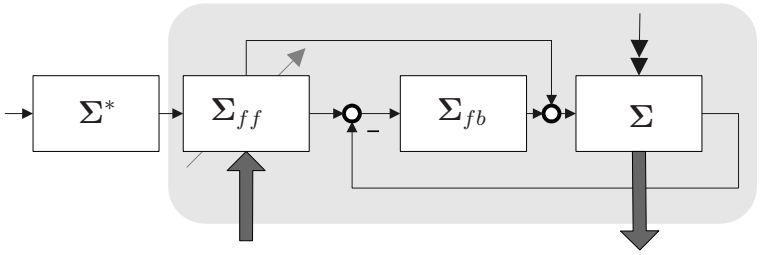
Die Bestimmung der dynamisch relevante Parameter $\boldsymbol{p}_{dyn}$ ist deutlich aufwendiger als die Bestimmung der Reibung, da hierbei, wie bereits beschrieben, starke Verkopplungen der Parameter auftreten und diese nicht getrennt identifiziert werden können. Zudem hat hierbei der Regler des Roboters einen deutlichen Einfluss auf den Verlauf der Messgrößen.

In unterschiedlichen Versuchen hat sich gezeigt, dass es für eine Optimierung der Parameter hilfreich ist möglichst viele aussagekräftige Messgrößen aufzuzeichnen und mit simulierten Messgrößen zu vergleichen. Mit nur wenigen (wichtigen) aufgezeichneten Messgrößen kann zwar das globale Verhalten des Roboters auch schon recht genau bestimmt werden, allerdings kann der Einfluss der einzelnen Parameter schlechter getrennt bestimmt werden, da ein identisches globales Verhalten meist durch verschiedene Parameter-Kombinationen erreicht werden kann. Dies zeigt sich, wenn eine Sensitivitätsanalyse der Modelle durchgeführt wird. Sensitivitätsanalysen können direkt mit Hilfe des Programms „Multi-Objective Parameter Synthesis“ (kurz *MOPS*, siehe auch Abschnitt 4.5 und A.7) durchgeführt werden. Hierbei werden für vorgegebene Parameterbereiche $\mathbf{p} \in [\mathbf{p}_{min}, \mathbf{p}_{max}]$ Gradienten $\frac{\partial \mathfrak{G}}{\partial \mathbf{p}}$ bezüglich des gewählten Kriteriums $\mathfrak{G}$ berechnet. Hierdurch kann der Einfluss der einzelnen Parameter auf das Kriterium bestimmt werden, wodurch es auch möglich ist lineare Abhängigkeiten der Parameter bezüglich des Kriteriums festzustellen.

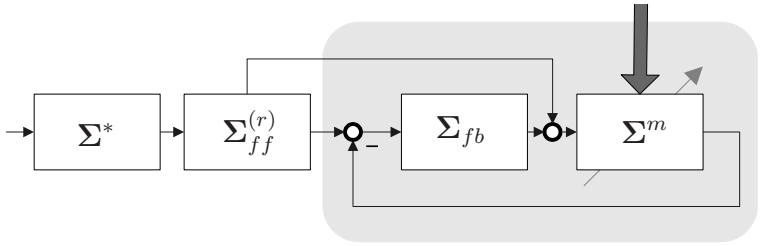
Zur Identifikation der Modelle gibt es viele verschiedene Varianten und Konstellationen. Zudem ist es möglich, zunächst die Parameter eines Vorwärtsmodells $\boldsymbol{\Sigma}^m$ der realen Strecke $\boldsymbol{\Sigma}$ zu bestimmen oder direkt die Parameter des inversen Streckenmodells $(\boldsymbol{\Sigma}^m)^{-1}$, bzw. der hieraus abgeleiteten Vorsteuerung $\boldsymbol{\Sigma}_{ff}$ (siehe hierzu auch Kapitel 3 und 5). Eine Übersicht über mögliche (und sinnvolle) Strukturen zur Identifikation zeigt Abb. 4.2.

Einige Varianten sollen im Folgenden behandelt werden. Es wird auch kurz auf ihre Vor- und Nachteile eingegangen. Grundsätzlich muss hierbei beachtet werden, dass Messungen am Roboter nur mit einem Regler durchgeführt werden können (bis auf Achse 1), da der Roboter aufgrund der Schwerkraft kein stabiles System darstellt.

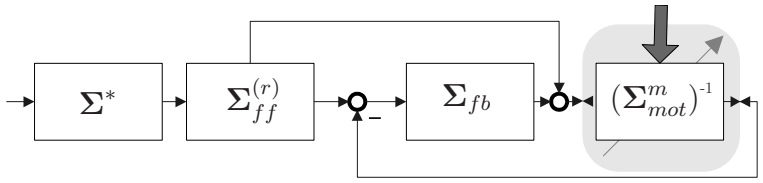
[3]Unter dem Begriff *Case* wird hierbei eine Messung, bzw. Simulation des Roboters aus einer Startpose unter einer bestimmten Anregung verstanden. Unterschiedliche *Cases* unterscheiden sich demnach in der Startpose und/oder Anregung. Sie dienen dazu, dass Parameter eines Modells möglichst global gültig sind und nicht nur für einen bestimmten Arbeitspunkt. Mehr hierzu im Anhang unter A.7.



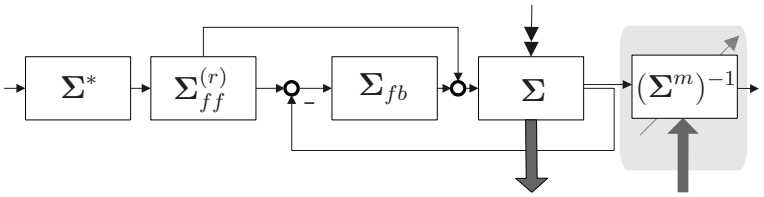
(a) Direkte Identifikation der Parameter einer Vorsteuerung $\mathbf{\Sigma}_{ff}$ auf Basis eines inversen Streckenmodells im Experiment am Roboter.



(b) Identifikation der Parameter eines Streckenmodells $\mathbf{\Sigma}^m$ inklusive Regler in der Simulation.



(c) Identifikation der Parameter eines motorseitigen inversen Streckenmodells „freigeschnitten“ vom Regelkreis in der Simulation.



(d) Identifikation der Parameter eines inversen Streckenmodells $(\mathbf{\Sigma}^m)^{-1}$ direkt am Roboter oder in der Simulation (dann mit $\mathbf{\Sigma}^m$ anstelle der Strecke $\mathbf{\Sigma}$).

Abbildung 4.2: Mögliche Strukturen zur Identifikation von Modellparametern. $\mathbf{\Sigma}^*$ steht hierbei für einen Vorfilter, $\mathbf{\Sigma}_{ff}$ für eine Vorsteuerung bzw. $\mathbf{\Sigma}_{ff}^{(r)}$ für eine vereinfachte (Starrkörper-) Vorsteuerung, $\mathbf{\Sigma}_{fb}$ für einen (Mehrachs-) Regler, $\mathbf{\Sigma}$ für die reale (gestörte) Regelstrecke und $\mathbf{\Sigma}^m$ für ein Streckenmodell. Die breiteren Pfeile stehen für die Verwendung von Messdaten, welche entweder gewonnen werden (Pfeil zeigt vom Block weg), oder zur Kriteriengenerierung (Pfeil zeigt zum Block hin) verwendet werden. Der grau hinterlegte Bereich deutet das für die Optimierung „freigeschnittene“ System an (Grey-Box).

Für die Identifikation eignen sich hierbei einfache Regler mit schwacher Dämpfung. Um den Roboter nicht zu extrem anzuregen und Beschädigung von Getriebe und Motor zu vermeiden, kann zudem eine einfache (leicht zu modellierende) Vorsteuerung für die Messungen verwendet werden. Hierzu eignet sich eine Starrkörpervorsteuerung $\mathbf{\Sigma}_{ff}^{(r)}$. Die Vorsteuerung trägt dazu bei, zu extreme Anregungen zu vermeiden, ohne die dynamischen Effekte der Elastizitäten des Roboters zu stark zu unterdrücken. Die Starrkörpervorsteuerung $\mathbf{\Sigma}_{ff}^{(r)}$ besteht aus einem inversen Modell der als starr angenommenen Struktur des Roboters sowie der Motorträgheiten und berücksichtigt keine Elastizitäten und dynamischen Antriebsstrangseffekte.

Dies ist vor allem bei den größeren Robotern sinnvoll, da hier sehr große Kräfte wirken können. Bei kleineren Robotern kann die Vorsteuerung auch weggelassen werden, ohne eine Beschädigung des Roboters durch zu starke Anregung zu riskieren.

1. **Direkte Optimierung der Parameter der Vorsteuerung im Experiment** (Abb. 4.2(a)): Eingangsgrößen für die Vorsteuerung $\mathbf{\Sigma}_{ff}$, welche auf einem inversen Modell des Roboters basiert, sind die über den Vorfilter $\mathbf{\Sigma}^*$ gefilterten Trajektorien. Messungen werden direkt am Roboter erstellt und als Kriterium aufbereitet und zur Optimierung von $\mathbf{\Sigma}_{ff}$ verwendet. Als Kriterium dienen hierbei die Abweichungen von der Bahn, bzw. das Schwingungsverhalten des Roboters beim Positionieren.
   Vor- und Nachteile:

   - (+) Als Kriterium ist direkt das Fahrverhalten des Roboters bewertbar.
   - (+) Direkt sichtbares Ergebnis an der realen Strecke.
   - (−) Jede einzelne Auswertung des Kriteriums stellt ein eigenes Experiment am Roboter dar, welches mit individuellen (statistischen) Fehlern behaftet ist. Daher können gradientenbasierte Optimierungsverfahren nur sehr beschränkt eingesetzt werden.
   - (−) Extrem zeitaufwendig selbst als „Hardware in the Loop“ (HIL) Optimierung.
   - (−) Parametergrenzen für die Optimierung müssen sehr eng gewählt werden, da der Roboter durch extrem falsche Parameter beschädigt werden kann (durch zu starke Anregung).

2. **Optimierung des Streckenmodells mit Regler in der Simulation** (Abb. 4.2(b)): Eingangsgrößen für das Simulationsmodell sind die Ausgangsgrößen der Vorsteuerung $\mathbf{\Sigma}_{ff}^{(r)}$. Die vom Streckenmodell $\mathbf{\Sigma}^m$ generierten virtuellen Messgrößen werden mit realen Messdaten verglichen, um zu einem Kriterium zu gelangen. Der Regler wird in der Simulation mitberücksichtigt.
   Vor- und Nachteile:

   - (+) Dynamik des Reglers wird in der Simulation komplett berücksichtigt.
   - (+) Simulation entspricht der Realität sehr genau.
   - (−) Exaktes Reglermodell muss existieren. Da der Regler real als diskretes System vorliegt muss dieses in der Simulation beachtet werden oder der Regler muss als kontinuierliches Modell approximiert werden (gemischt diskret

kontinuierliche Modelle wären zwar realisierbar, aber sie benötigen deutlich mehr Rechenzeit).

- (−) Bei schlechten Parameterwerten (bzw. Startwerten) kann der Regler das System nicht mehr stabilisieren, wodurch extreme Abweichungen bei den Kriterien auftreten (schlecht für gradientenbasierte Optimierungsverfahren).

3. **Optimierung des Streckenmodells „freigeschnitten“ vom Regelkreis in der Simulation** (Abb. 4.2(c)): Eine Alternative zur Simulation der Strecke mit Regler ist es, statt der Motormomente $\boldsymbol{\tau}_{m,ist}$ die gemessenen Motoristpositionen $\boldsymbol{\varphi}_{m,ist}$ als Eingang für ein motorseitiges inverses Streckenmodell $(\boldsymbol{\Sigma}^m_{mot})^{-1}$ zu verwenden. Bei diesem inversen Modell sind Motormomente $\hat{\boldsymbol{\tau}}_m$ dann Teil der Ausgangsgrößen.

   Das motorseitige inverse Streckenmodell $(\boldsymbol{\Sigma}^m_{mot})^{-1}$ ist ohne Regler $\boldsymbol{\Sigma}_{fb}$ in der Simulation stabil und der Regler kann vernachlässigt werden. Weitere Details zur Generierung des inversen Modells finden sich in in Abschnitt 3.4. In $\boldsymbol{\varphi}_{m,ist}$ ist indirekt die Dynamik des Reglers bereits enthalten und die Messdaten, welche mit Regler gewonnen wurden, können zur Optimierung verwendet werden.
   Vor- und Nachteile:

   - (+) Kein Reglermodell nötig.
   - (+) Minimal mögliche Rechenzeit für die Simulation, da kein Regler mitberechnet werden muss und $\boldsymbol{\varphi}_m$ sowie $\dot{\boldsymbol{\varphi}}_m$ keine Zustände mehr sind.
   - (+) Streckenmodell stabil (robust gegenüber den Parametern).
   - (−) $\boldsymbol{\varphi}_{m,ist}$ muss zweimal stetig differenzierbar sein, was nur mit Hilfe einer Filterung der Daten möglich ist, wodurch die Dynamik verfälscht wird.
   - (−) Getriebedämpfungen können weniger genau bestimmt werden (was sich in Optimierung gezeigt hat), da sich ungenaue Dämpfungsparameter durch Vorgabe von $\boldsymbol{\varphi}_{m,ist}$ weniger stark auf die Gesamtdynamik des Roboters auswirken.

4. **Optimierung des inversen Streckenmodells in der Simulation**
   (Abb. 4.2(d)): Als Eingangssignale für ein inverses Streckenmodell $(\boldsymbol{\Sigma}^m)^{-1}$ dient die gemessene TCP- bzw. HWP-Position und Orientierung (welche mit einer externen 6D-Kamera aufgezeichnet werden). Die Ausgangssignale und virtuellen Messdaten können dann mit den an der Strecke gemessene Werten verglichen werden, um zu einem Kriterium zu gelangen. Anmerkung.: $(\boldsymbol{\Sigma}^m)^{-1}$ unterscheidet sich in diesem Fall natürlich gegenüber dem motorseitigen inversen Modell $(\boldsymbol{\Sigma}^m_{mot})^{-1}$, welches zur Optimierung des Streckenmodells „freigeschnitten“ vom Regelkreis in der Simulation verwendet wird.
   Vor- und Nachteile:

   - (+) Direkte Identifikation des inversen Modells, aus welchem eine Vorsteuerung abgeleitet werden kann.
   - (+) Approx. inverses Streckenmodell immer stabil, auch für grob falsche Parameter.
   - (+) Kein Reglermodell notwendig.

- (−) Da eine exakte Invertierung des Streckenmodells nicht möglich ist (siehe Kap. 3) müssen Approximationsfehler beachtet werden, wodurch eventuell falsche Parameter identifiziert werden.
- (−) Die Eingangssignale des inversen Modells müssen mehrfach stetig differenzierbar sein (siehe Kap. 3). Dies ist für Messdaten einer 6D-Kamera nur mit Hilfe einer Filterung der Daten möglich, wodurch die Dynamik verfälscht wird.
- (−) Sichtbereich einer 6D-Kamera ist klein, es sind daher aufwendige Messungen notwendig, da die Kamera für unterschiedliche Stellungen des Roboters neu platziert und justiert werden muss.

Aus diesen Möglichkeiten hat sich eine Kombination aus den unterschiedlichen Strategien als am effektivsten herausgestellt:

1. Zunächst werden die Parameter des Streckenmodells $\mathbf{\Sigma}^m$ über eine Optimierung „freigeschnitten“ vom Regelkreis in der Simulation auf Basis von $(\mathbf{\Sigma}^m_{mot})^{-1}$ bestimmt. Hierbei kann aufgrund der geringen Rechenzeit und den Stabilitätseigenschaften ein großer Parameterbereich für die Optimierung verwendet werden.

2. Mit den aus dem ersten Schritt ermittelten Parameter (mit welchen das System bereits sehr genau beschrieben werden kann) wird eine neue Optimierung des Streckenmodells, mit Regler in der Simulation und enger gesetztem Parameterbereich (um die Stabilität zu gewährleisten) durchgeführt, um Dämpfungsterme genauer bestimmen zu können.

3. Anschließend kann das so gewonnene Streckenmodell approx. invertiert werden (siehe Kapitel 3). Daraus kann eine Vorsteuerung $\mathbf{\Sigma}_{ff}$ generiert werden. Die kann über eine HIL[4]-Optimierung am Roboter noch einmal mit einem sehr engen Parameterbereich – um die bereits gewonnen Parameter – feinjustiert werden.

Das direkte Optimieren des approx. inversen Modells hat sich aufgrund der hohen Differentiationsanforderungen an die 6D-Kamera-Messdaten als nicht praktikabel erwiesen.

Für den ersten Schritt muss die Motor-Istposition mit einem Filter von mindestens zweiter Ordnung gefiltert werden, um die benötigten Ableitung bereitzustellen. Dies ist aber schon aus numerischer Sicht sinnvoll, da es sich bei der Motor-Istposition um gemessene Sensorgrößen handelt, die mit einem Messrauschen behaftet sind. Eine leichte zeitliche Verschiebung des Messsignals durch die Filter muss dafür in Kauf genommen werden, die abhängig von der Filterordnung n_f und der Eckfrequenz f_c des Filters ist. Die Filter sollten daher mit möglichst niedriger Ordnung und möglichst hoher Eckfrequenz – wie numerisch möglich – dimensioniert werden.

Um diese Problem zu verringern kann bei der Datenaufbereitung der Messdaten eine kausal/akausale (vorwärts/rückwärts) Filterung (siehe [95, 96]) der Motor-Istpositions-Daten vorgenommen werden. In Kombination mit einer numerischen Differenzierung

[4]Hardware in the loop

(Gl. (4.2)) über einen zentralen Differenzenquotient (Fehler in $\mathcal{O}(\Delta t^2)$)

$$\frac{\partial f}{\partial t}(t) \approx \frac{f(t+\Delta t) - f(t-\Delta t)}{2\Delta t} \tag{4.2}$$

können bereits bei der Datenaufbereitung die beiden benötigten Ableitungen berechnet werden und eine Phasenverschiebung vermieden werden. Dennoch muss durch die (notwendige) Filterung eine gewisse Verzerrung der Daten in Kauf genommen werden.

Um das Rauschen der Messwerte zusätzlich zu reduzieren, werden alle Messungen dreimal durchgeführt und die Messwerte gemittelt. Damit ein möglichst vollständiges Bild der Dynamik des Roboters gefunden werden kann werden alle verfügbaren Sensoren eingesetzt.

Durch die hohe Wiederholgenauigkeit der Roboter ist es möglich – für eine identische Anregung – gut reproduzierbare Ausschwingvorgänge zu messen. Für aufeinanderfolgende Messungen, mit identischer Anregung, führt dies zu nahezu identischen Regel- und Messgrößen. Hierdurch ist es möglich, durch wiederholtes Messen bei identischer Anregung, wobei die Position der Sensoren am Roboter verändert wird, beliebig viele „virtuelle“ Sensoren zu realisieren. Hierdurch können umfangreiche Messdatensätze erstellt werden, obwohl nur eine begrenzte Anzahl an (realen) Sensoren verfügbar ist.

Um die Messgrößen der „virtuellen“ Sensoren, bzw. unterschiedlichen Messungen mit identischer Anregung, zu fusionieren kann über eine Kreuzkorrelation der Sollwerte der Motorgeschwindigkeit $\dot{\boldsymbol{\varphi}}_{m,soll}$ der Zeitversatz der Messdaten ermittelt werden. Dies ermöglicht es die Datensätze möglichst exakt zusammen zu führen, um dadurch einen umfangreichen Datensatz zu erhalten, der die Information von allen „virtuellen“ Sensoren enthält.

Der Kreuzkorrelationsvektor $\hat{\mathbf{R}}_{xy}$ von zwei Messvektoren $\mathbf{x}$ und $\mathbf{y}$ der Länge N (Samplezeit T_s) ist dabei wie in Gl. (4.3) definiert. $\hat{\mathbf{R}}_{xy}$ stellt eine Approximation der wahren Kreuzkorrelation $\mathbf{R}_{xy}$ dar, welche nur für unendliche Datenmengen exakt definiert ist.

$$\hat{\mathbf{R}}_{xy}(m) \;=\; \begin{cases} \sum_{n=0}^{N-m-1} x_{n+m} y_n, & m \geq 0 \\ \hat{\mathbf{R}}_{xy}(-m), & m < 0 \end{cases} \tag{4.3}$$

Über das Maximum des Kreuzkorrelationsvektors $\hat{\mathbf{R}}_{xy}$ kann dann der Zeitversatz (Offset) der Messsignale bestimmt werden. Für weitere Details zu Kreuzkorrelation sei auf [97] verwiesen.

Alle Messungen müssen in unterschiedlichen Achskonfigurationen und für unterschiedliche Bewegungen des Roboters (sog. *Cases*) wiederholt werden, damit ein global gültiges Modell gefunden werden kann.

Eine Übersicht, wie die Datensätze für die Optimierung der dynamischen Parameter gewonnen werden, zeigt Abbildung 4.3.

Die N_s Sensorkonfigurationen sind bei den Messungen so gewählt, dass für jedes, für die Dynamik des Roboters wichtige Strukturteil, also Karussell, Schwinge und Arm des Roboters Drehratensensoren orthogonal zueinander in allen drei Raumrichtungen angebracht wurden. Für die Schwinge wurde dies zusätzlich in drei verschiedenen Höhen bezüglich Achse 2 durchgeführt. Zusätzlich wurde ein 3D-Beschleunigungssensor am Handwurzelpunkt des Roboters befestigt.

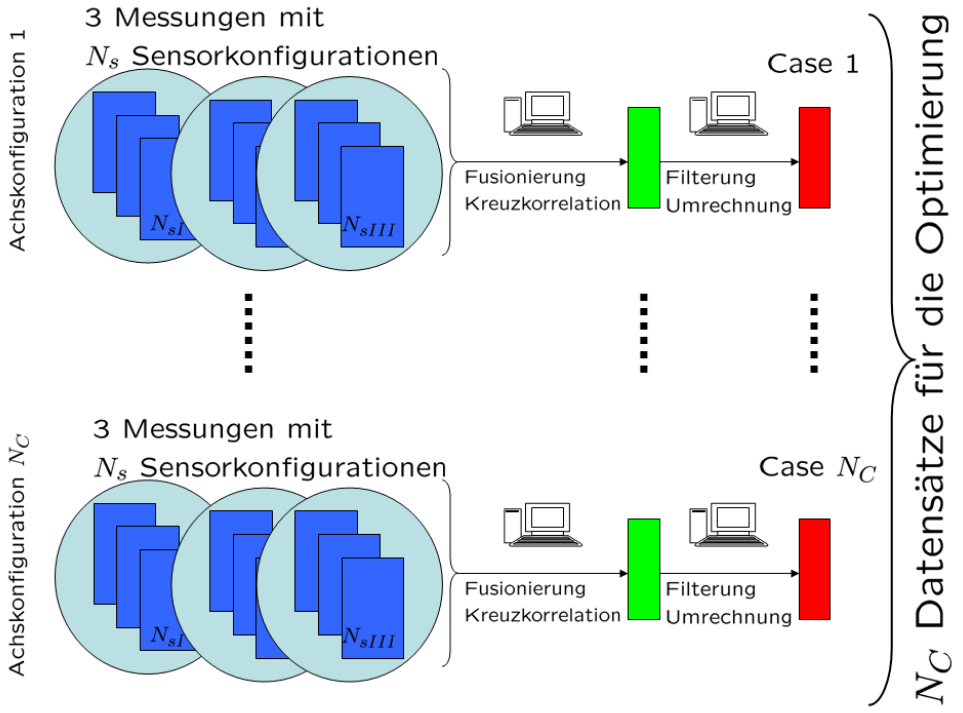


Abbildung 4.3: Aufbereitung der Messdaten für die Optimierung.

Da die Präzision der Drehratensensoren und des Beschleunigungssensors zur Erfassung der Dynamik des Roboters höher als die der Kamera ist, wurden Kameradaten nur zur Verifikation verwendet, da sie keinen zusätzlichen Informationsgehalt liefern.

Für die Anregung des Roboters wurden Mehrachsbewegungen mit unterschiedlicher Verfahrlänge gewählt. Besonders kurze PTP-Bewegung der drei Grundachsen stellen eine starke Anregung des Roboters dar. Dies führt zu sehr steilen Beschleunigungsverläufen wodurch ein hoher Frequenzbereich der Robotermechanik angeregt wird. Theoretisch wäre auch ein Sprung oder Impuls denkbar, dies ist in der Praxis aber nur schwer realisierbar ohne eine Beschädigung des Roboters zu riskieren. Es muss daher mit kurzen PTP-Bewegungen ein Kompromiss eingegangen werden, um den Roboter möglichst stark Anzuregen, ohne ihn jedoch zu beschädigen.

Identische Anregungen wurden auch für die drei Handachsen des Roboters durchgeführt. Die Hand (Achse 4 bis 6) des Roboters ist aber relativ starr, so dass die Handachsen nur einen sehr geringen Beitrag zum dynamischen Verhalten des Roboters liefern. Durch die Fusionierung der so gewonnen Messdaten, zusammen mit den Sensordaten der Sensoren, die direkt im Roboter verbaut sind, entstehen sehr umfangreiche Datensätze. Diese können in aufbereiteter Form für eine Optimierung eines Robotermodells verwendet werden.

Analog zur Optimierung der Reibung muss auch zur Bestimmung der dynamischen Parameter eine Gütefunktion $\mathfrak{G}_k$ für den Optimierer, für jeden einzelnen *Case* aufgestellt werden, wobei wieder die Summe der Gütefunktionen über alle *Cases*, $\mathfrak{G}$ minimiert wird.

Im Simulationsmodell werden hierzu Sensoren an denselben Orten wie bei den realen Messungen angebracht (bzw. die entsprechenden Größen werden anhand der Zustände

berechnet). Die Differenz zwischen den simulierten Messungen und den realen Messwerten werden hierbei zur Bildung der Gütefunktion verwendet. Gleichung (4.4) beschreibt die hierbei verwendete Gütefunktion, wobei $\hat{(.)}$ für die simulierten Messgrößen steht. Die Faktoren $k_\omega, k_{\tau A,i}$ sowie $k_{\ddot{r},L}$ sind dabei Skalierungsfaktoren, welche benötigt werden, um die einzelnen Fehler auf etwa gleiche Größenordnungen zu skalieren.

$$\mathfrak{G}_{\tau,i} = \begin{cases} k_{\tau A,i}(\tau_{m,i} - \hat{\tau}_{m,i})^2, & \text{wenn } |\dot{\varphi}_{m,ist,i}| > \dot{\varphi}_{m,i,min} \\ 0, & \text{sonst} \end{cases} \tag{4.4a}$$

$$\mathfrak{G}_{\omega,j} = k_\omega(\omega_j - \hat{\omega}_j)^2 \tag{4.4b}$$

$$\mathfrak{G}_{\ddot{\mathbf{r}},L} = k_{\ddot{\mathbf{r}},L}\|\ddot{\mathbf{r}}_{L,t} - \hat{\ddot{\mathbf{r}}}_{L,t}\|^2 \tag{4.4c}$$

$$\mathfrak{G}_k = \int_0^{t_{end}} \left(\sum_{i=1}^{6} \mathfrak{G}_{\tau,i} + \sum_{j=1}^{N_s n_{sens}} \mathfrak{G}_{\omega,j} + \mathfrak{G}_{\ddot{\mathbf{r}},L} \right) dt \tag{4.4d}$$

$$\mathfrak{G} = \sum_{k=1}^{N_C} \mathfrak{G}_k \tag{4.4e}$$

$\mathfrak{G}_{\tau,i}$ ist hierbei wieder nur für $|\dot{\varphi}_{m,i}| > \dot{\varphi}_{m,i,min}$ definiert, da zur Messung des Moments die Strommessung des Motors herangezogen wird. Sind die Bremsen des Motors geschlossen oder befindet sich die jeweilige Achse i momentan im Zustand der Haftreibung, können keine vernünftigen Werte für die Momente ermittelt werden. Um die Zahl der zu optimierenden Parameter zu reduzieren, werden für die Simulation bereits die gewonnen Parameter-Werte für die quasi-statischen und statischen Parameter verwendet.

4.3 Optimierung der Parameter der Modelle zur Identifikation

Über eine Optimierung der Parameter der Modelle soll nun die entsprechenden Gütefunktionen $\mathfrak{G}$ minimiert werden.

Die Optimierung basiert dabei auf einem Programm mit dem Namen „Multi-Objective Parameter Synthesis“ (kurz *MOPS*). Für eine genauere Beschreibung siehe Abschnitt A.7 im Anhang.

Untersuchungen anhand der erstellten Modelle haben ergeben, dass sich besonders gradientenfreie Optimierungsverfahren, sowie Verfahren die sich parallelisieren lassen, für die Optimierung der Roboter-Parameter eignen. Gradientenbasierte Verfahren neigen bei den stark nichtlinearen Modellen dazu, in lokalen Minima festzustecken und benötigen daher sehr gute Startwerte, welche im Allgemeinen nicht gegeben sind. Eine Parallelisierung ist aufgrund des hohen Rechenaufwands sinnvoll, der durch die große Anzahl an Parametern ersteht. Zudem benötigen globale Optimierungsverfahren deutlich mehr Funktionsauswertungen[5] als lokale gradientenbasierte Verfahren, wodurch die Rechenzeit zusätzlich ansteigt. Am Institut für Robotik und Mechatronik des DLR existiert für die Optimierung ein Linux basierter Cluster mit 30 CPUs (RM-Cluster).

[5]Eine Funktionsauswertung von $\mathfrak{G}(\mathbf{p})$ entspricht hierbei immer einer kompletten Simulation für einen Datensatz.

Die einzelnen Knotenrechner sind Dell Precision 450 Workstations mit folgenden Eigenschaften:

- Prozessortyp: Intel Xeon 2.8 GHz
- Speichergröße: 2 GByte
- Netzanbindung: 100 Mbit

Als besonders effektiv hat sich, bei verschiedenen Optimierungen der Parameter[6] der Modelle, eine Kombination aus Genetischem-Algorithmus mit anschließender Optimierung mit dem Pattern-Algorithmus (siehe [98]) herausgestellt.

Beide Algorithmen sind gradientenfreie Verfahren und benötigen nur den Funktionswert $\mathfrak{G}(\mathbf{p})$ für ihre Suchstrategie. Zudem sind beide Verfahren mit Hilfe von *MOPS* parallelisierbar (siehe hierzu auch [99]). Hierdurch eignen sie sich für eine Optimierung auf dem Cluster.

Der in *MOPS* implementierte Genetische Algorithmus (GA) basiert auf Evolutionsprinzipien und stellt eine globale Optimierungstechnik dar. Im Folgenden soll er kurz beschrieben werden, für weitere Details sei auf [100, 101, 102] verwiesen. Der Algorithmus besteht aus mehreren Teilschritten:

- Initialisierung
- Evaluation
- Selektion
- Rekombination
- Mutation

Zunächst werden in der Initialisierung zufällig n_p verschiedene Tunervektoren $\mathbf{p}$ generiert, wobei die einzelnen Tunervariablen innerhalb der vorzugebenden Grenzen liegen $\mathbf{p} \in [\mathbf{p}_{min}, \mathbf{p}_{max}]$. Die Auswahl dieser Grenzen hat einen entscheidenden Einfluss auf die Güte des Algorithmus. Werden die Grenzen für die Parameter zu groß gewählt, ist der mögliche Parameterraum zu groß und das Suchverfahren hat nur eine sehr geringe Chance ein Optimum zu finden. Ist der Bereich zu klein gewählt, kann nur ein lokales Minimum gefunden werden.

Die Anzahl n_p der in der Initialisierungsphase gewählten Parametervektoren $\mathbf{p}$ wird in Anlehnung an die Biologie als „Populationsgröße“ bezeichnet; ein einzelner Parametersatz als Individuum. Für jeden einzelnen Vektor werden nun die entsprechenden Gütefunktionen $\mathfrak{G}(\mathbf{p})$ über Simulationen berechnet (auch Fitness-Funktion genannt). Danach folgt die Selektion: Hierbei werden zufällige Individuen aus der Population ausgewählt, wobei Individuen mit besserer Gütefunktion mit einer höheren Wahrscheinlichkeit ausgewählt werden. In dem in *MOPS* implementierten Verfahren erfolgt dies über das sog. Tunierverfahren, wobei immer zwei Individuen verglichen werden. Durch

[6] In *MOPS* werden die zu optimierenden Parameter als „Tuner“ bezeichnet. Siehe hierzu auch Abschnitt A.7 im Anhang.

die „Elite“-Option in *MOPS* wird festgelegt, dass in diesem Schritt auch immer das beste bisher gefundene Individuum in den nächsten Schritt übernommen wird.

Nach der Selektion erfolgt die Rekombination. Hierbei werden einzelne Parameter aus den Parametervektoren $\mathbf{p}$ (auch Genome genannt) der verschiedenen Individuen gemischt und aus den neuen Parametervektoren $\mathbf{p}$ eine neue Generation erzeugt (Vermehrung). Verwendet wir hierbei der sog. *uniform crossover*-Algorithmus (siehe [103]).

Anschließend folgt die sog. „Mutation“ der Individuen. Wie hoch die Mutationswahrscheinlichkeit ist, kann in *MOPS* über zwei Parameter eingestellt werden. Bei der Mutation wird dabei zwischen der „Kriech-Mutation“ (engl.: Creep mutation) und der „Sprung-Mutation“ (engl.: Jump mutation) unterschieden. Im ersten Fall handelt es sich um einen lediglich geringe Änderung eines Parameterwertes aus dem Vektor, wohingegen in zweiten Fall auch sehr große Änderungen zugelassen werden.

Aus der Menge der alten und den neuen erzeugten Individuen wird nun eine neue Generation erzeugt und der Algorithmus beginnt wieder bei der Evaluation.

Dies wird so lange wiederholt, bis entweder eine festgelegte Anzahl an Funktionsauswertungen erfolgt ist oder für einige Generationen kein besseres Individuum gefunden werden kann.

Das Verfahren benötigt einen sehr großen Rechenaufwand, da viele Auswertungen von $\mathfrak{G}(\mathbf{p})$ erfolgen müssen. Es hat jedoch den großen Vorteil, nicht in lokalen Minima feststecken zu bleiben. Bei der Optimierung der Parameter der Modelle hat es sich als sehr effektiv herausgestellt.

Nachteilig ist zudem, dass ein gefundenes Minimum nicht genau gefunden wird. Weshalb es vorteilhaft ist, nach dem Genetischen Algorithmus noch ein lokales Suchverfahren zu verwenden.

Je nach Modell hat sich hierbei der SQP [104] oder der Pattern-Algorithmus [98] bewährt. Der SQP-Algorithmus konvergiert deutlich schneller als der Pattern-Algorithmus (super-lineare Konvergenz). Er verwendet allerdings Gradienten, um zu dem Optimum zu gelangen. Exakte Gradienten setzten eine sehr hohe numerische Genauigkeit bei der Optimierung voraus, was sich negativ auf die Rechenzeit auswirkt.

Der Pattern-Algorithmus verwendet lediglich direkte Funktionsauswertungen in seinem Suchalgorithmus und ist numerisch robuster. Er benötigt allerdings deutlich mehr Funktionsauswertungen. Zudem ist es bei Verwendung des Pattern-Algorithmus nach der Optimierung mit dem GA sinnvoll, die Grenzen für die Parameter zu verkleinern, da es das Ziel ist, nur noch eine lokale Verbesserung des bisher gefundenen Optimums $\mathbf{p}^*$ zu finden. Alternativ kann die Startschrittweite des Pattern-Algorithmus verringert werden, wodurch die Suche nur lokal stattfindet.

4.4 Zusammenfassung des Verfahrens zur Identifikation

In den vorangegangenen Abschnitten des Kapitels wurden die durchgeführten Messungen und einzelnen Schritte zur Identifikation der Robotermodelle und der daraus

abgeleiteten inversen Robotermodellen beschrieben, wobei auch auf möglich Varianten des Verfahrens eingegangen wurde.

Im Folgenden soll noch einmal der Prozess, der zur Identifikation der Modelle aus dieser Vielfalt gewählt wurde, in einer kurzen Form anhand von Abb. 4.4 dargestellt werden.

Das Verfahren lässt sich in fünf Schritte einteilen, wobei bereits davon ausgegangen wird, dass ausreichend Information über das zu identifizierenden System in Form von Messungen, wie sie in den vorangegangenen Abschnitten beschrieben wurden, und CAD-Daten vorliegt.

1. Zunächst werden aus CAD-Daten und Tabellen die statischen Parameter $\boldsymbol{p}^*_{stat}$ ermittelt. Diese können als bekannt und damit auch als optimal angenommen werden. Bei Unsicherheiten in den Parametern können kleine Korrekturterme über eine Optimierung auf Basis von statischen Messdaten ermittelt werden.

2. Mit Hilfe von $\boldsymbol{p}^*_{stat}$ und einfachen (starren) Modellen des Roboters $\boldsymbol{\Sigma}^{m,(r)}$ sowie $\left(\boldsymbol{\Sigma}^{m,(r)}_{mot}\right)^{-1}$ (ohne die Parameter $\boldsymbol{p}_{dyn}$) lassen sich die optimalen quasi-statischen Parameter $\boldsymbol{p}^*_{q-stat}$ über eine Optimierung ermitteln. Die Messungen hierzu sind so auszuwählen, dass die Dynamik des Systems und damit die Parameter $\boldsymbol{p}_{dyn}$ keinen (bzw. nur sehr geringen) Einfluss auf die Optimierung haben. Dies kann über stark (Tiefpass) gefilterte Anregungssignale erreicht werden, was über eine Modifikation des Vorfilters $\boldsymbol{\Sigma}^*$ erreicht werden kann.

3. Mit $\boldsymbol{p}^*_{stat}, \boldsymbol{p}^*_{q-stat}$ und geschätzten Startwerten $\boldsymbol{p}^{(0)}_{dyn}$ für die dynamisch relevanten Parameter des Systems, können nun über eine Optimierung in einem ersten Iterationsschritt, optimale Parameter $\boldsymbol{p}^{*,(1)}_{dyn}$ bezüglich des gewählten Kriteriums $\mathfrak{G}$ bestimmt werden. Hierbei wird eine Optimierung bezüglich des vom Regelkreis freigeschnittenen, motorseitigen inversen Systems $(\boldsymbol{\Sigma}^m_{mot})^{-1}$ gewählt. Bei dieser Methode dienen die gemessenen Motorwinkel $\boldsymbol{\varphi}_{m,ist}$ als Eingangsgrößen des Systems. Durch diese Wahl ist das System stets stabil. Durch die Vernachlässigung des Reglers in der Simulation wird zudem eine minimale Rechenzeit für ein Simulationsexperiment erreicht, wodurch ein sehr großer Bereich für die Grenzen um $\boldsymbol{p}^{(0)}_{dyn}$ gewählt werden kann. Um den zulässigen Parameterbereich einzuschränken wird eine Konstante $0 < k_0 < 1$ gewählt, wodurch sich ein zulässiger Parameterbereich von $k_0\boldsymbol{p}^{(0)}_{dyn} \leq \boldsymbol{p}^{(1)}_{dyn} \leq \frac{1}{k_0}\boldsymbol{p}^{(0)}_{dyn}$ für die Parameter der ersten Iteration $\boldsymbol{p}^{(1)}_{dyn}$ ergibt. Hierdurch muss die erste Schätzung von $\boldsymbol{p}^{(0)}_{dyn}$ nicht sehr genau sein, sollte jedoch physikalisch sinnvoll gewählt werden. Dadurch, dass im 4. Schritt eine weiterere Optimierung durchgeführt wird, kann für diesen Schritt auch eine geringe Toleranz[7]für die Integration in der Simulation gewählt werden, wodurch zusätzlich Rechenzeit eingespart werden kann.

4. Über einen weiteren Optimierungsschritt können nun noch einmal genauere Parameter über ein aufwendigeres Modell bestimmt werden. Dieses Modell besteht aus Robotermodell $\boldsymbol{\Sigma}^m$ und Regler $\boldsymbol{\Sigma}_{fb}$ (dessen Struktur und Parameter als bekannt angenommen wird) des Roboters und entspricht damit möglichst genau den durchgeführten Messungen am realen Roboter. Für diese Simulation wird

[7]Sinnvoll ist eine relative Toleranz von 10^{-4} bis 10^{-5} bei der Integration mit dem DASSL-Algorithmus, was zu geringen Rechenzeiten bei akzeptabler numerischer Güte führt.

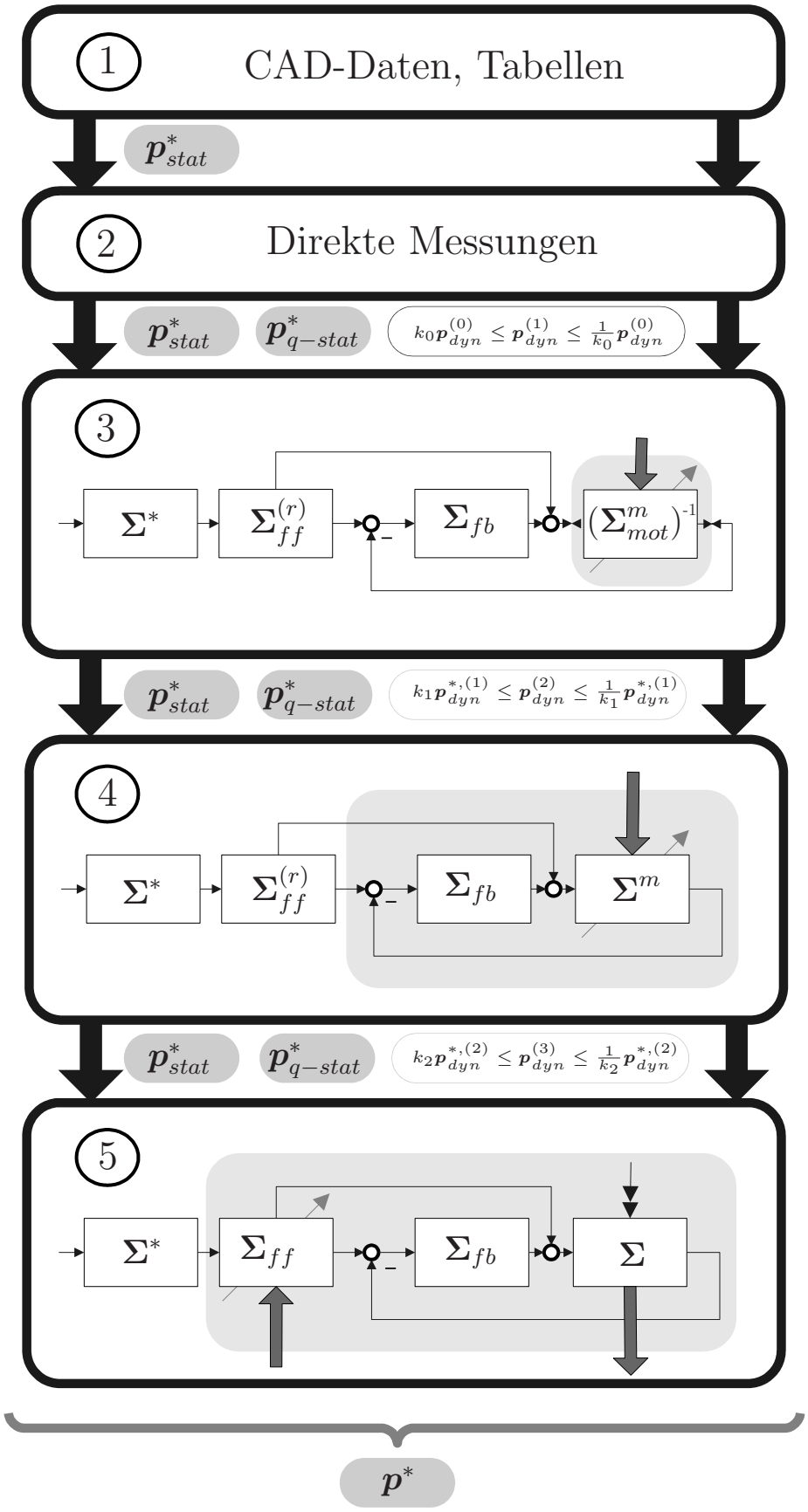


Abbildung 4.4: Zusammenfassung des Verfahrens zur Identifikation der optimalen Parameter $\boldsymbol{p}^*$ der Robotermodelle $\boldsymbol{\Sigma}^m$ und der davon abgeleiteten inversen Robotermodelle, welche als Vorsteuerung $\boldsymbol{\Sigma}_{ff}$ eingesetzt werden, in fünf Schritten.

eine hohe numerische Toleranz[8] gewählt, es werden jedoch die Parameterbereiche für die Optimierung der optimalen Parameter $\boldsymbol{p}_{dyn}^{*,(2)}$ stärker eingeschränkt. Hierzu wird eine Konstante $k_0 < k_1 < 1$ gewählt und der zulässige Bereich für die Parameter beschränkt ($k_1 \boldsymbol{p}_{dyn}^{*,(1)} \leq \boldsymbol{p}_{dyn}^{(2)} \leq \frac{1}{k_1} \boldsymbol{p}_{dyn}^{*,(1)}$).

5. Der fünfte Schritt kann als optional angesehen werden, kann jedoch noch einmal die Leistungsfähigkeit der aus dem Modell $\boldsymbol{\Sigma}^m$ abgeleiteten Vorsteuerung $\boldsymbol{\Sigma}_{ff}$ verbessern. Hierzu werden in einer direkten Optimierung am Roboter (HIL-Optimierung) die Parameter $\boldsymbol{p}_{dyn}^{*,(2)}$ des inversen Modells des Roboters variiert wobei die Grenzen der Parameter der Optimierung noch einmal deutlich eingeschränkt werden, was wieder über die Wahl eine Konstanten $k_1 < k_2 < 1$ geschieht. Die Grenzen werden analog zu Schritt vier zu $k_2 \boldsymbol{p}_{dyn}^{*,(2)} \leq \boldsymbol{p}_{dyn}^{(3)} \leq \frac{1}{k_2} \boldsymbol{p}_{dyn}^{*,(2)}$ festgelegt. Das Kriterium für die Optimierung wird dabei direkt über das Fahrverhalten des realen Roboters bewertet. Hierdurch können die für das Fahrverhalten des Roboters optimalen Parameter $\boldsymbol{p}_{dyn}^{*,(3)}$ bestimmt werden.

Nach der Durchführung dieser vier, bzw. fünf Schritte können die optimalen Parameter $\boldsymbol{p}^* = \{\boldsymbol{p}_{stat}^*, \boldsymbol{p}_{q-stat}^*, \boldsymbol{p}_{dyn}^*\}$ sehr effektiv bestimmt werden. Durch die Aufteilung der Optimierung in mehrere Teilschritte kann ein sehr großer Parameterbereich $\boldsymbol{p} \in [\boldsymbol{p}_{min}, \boldsymbol{p}_{max}]$ mit möglichst geringer Rechenzeit und möglichst wenigen zeitaufwendigen und fehleranfälligen Experimenten am Roboter durchgeführt werden. Durch die starke Nichtlinearität der Modelle kann jedoch auf globale Optimierungsverfahren nicht verzichtet werden, welche inhärent eine hohe Anzahl an Funktionsauswertungen benötigen. Daher ist es sinnvoll, wie bereits beschrieben, Schritt zwei bis vier auf einem Rechen-Cluster durchzuführen.

4.5 Simulationsergebnisse der Identifikation

Für den KR16 und den KR210 wurden, wie in Abschnitt 4.2 beschrieben, Messungen am Roboter durchgeführt und mit Hilfe der aufbereiteten Messdaten die Parameter von verschiedenen Modelle über eine Optimierung auf dem RM-Cluster ermittelt. Im Folgenden sollen einige Ergebnisse für den KR210 vorgestellt werden. Vergleichbare Ergebnisse wurden auch für den Roboter KR16 erzielt.

4.5.1 Modelle für den KR210

Die Modelle des KR210 setzen sich aus den Komponenten, welche in Kapitel 2 vorgestellt wurden zusammen. Bei dem Roboter KR210 besitzt vor allem die Schwinge des Roboters einen starken Einfluss auf die Dynamik des Roboters. Das Modell besteht aus den folgenden Komponenten:

- Motor als Trägheit (unter Vernachlässigung der gyroskop. Kopplungsterme)
- Motorreibung linear mit Haftreibungsapproximation (Gl. (2.3))

[8] Eine relative Toleranz von 10^{-6} bis 10^{-8} bei der Integration mit dem DASSL-Algorithmus führt zu sehr genauen Ergebnissen die sich auch mit höheren Toleranzen nicht mehr ändern.

- Nichtlineare Getriebe für alle 6 Achsen (Gl. (2.7))
- Elastisches Karussell
- Elastische Schwinge
- Elastischer Arm
- Gewichtsausgleich

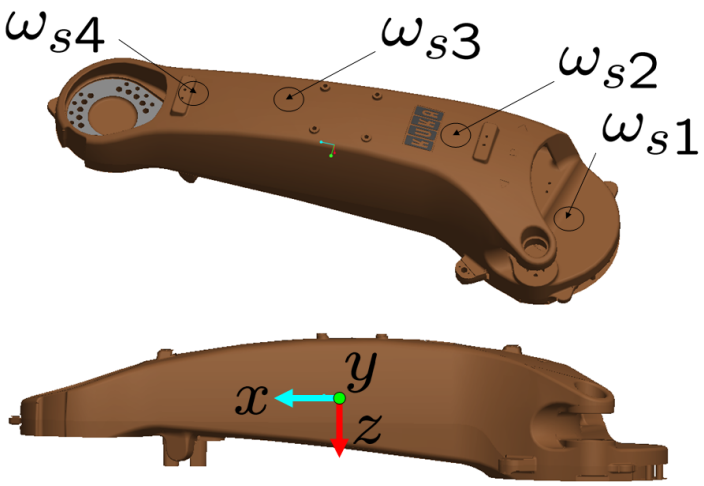


Abbildung 4.5: Sensorpositionierung an der Schwinge des KR210, sowie verwendetes lokales Koordinatensystem.

Für den KR210 wurden mehrere Drehratensensoren an der Schwinge angebracht, um ein umfassendes Bild zu erhalten. Insgesamt wurde an vier unterschiedlichen Stellen Drehratensensoren (jeweils in x, y und z Richtung bezüglich dem jeweiligen lokalen Koordinatensystem) angebracht. Die Positionierung der Sensoren an der Schwinge, sowie das dort verwendete lokale Koordinatensystem, zeigt Abbildung 4.5.

Zusätzlich wurde ein 3D-Beschleunigungssensor am TCP des Roboters angebracht, sowie Drehratensensoren am Karussell und an der Hand (wieder jeweils für die x, y und z-Richtung, im jeweiligen lokalen Koordinatensystems). In den Abbildungen 4.6 und 4.7, sind einige Messungen und entsprechende Simulationsergebnisse für den Roboter in Strecklage ($A1 = 0, A2 = 0, A3 = 0, A4 = 0, A5 = 0, A6 = 0$) dargestellt. Entsprechend in Abbildung 4.8 und 4.9 sowie in Abbildung 4.10 und 4.11 für zwei weiter Stellungen.

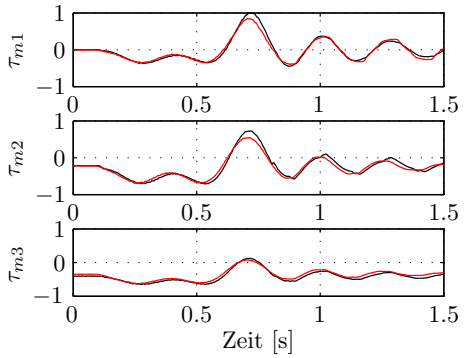
Das sind drei typische Beispiele der neun, für die Messungen verwendeten, Stellungen (mit 5° und 20° Dreichachsbewegungen[9] zur Anregung, also insgesamt 18 *Cases*) welche auch zur Optimierung verwendet wurden.

[9]Mit Dreichachsbewegungen werden hier simultane Bewegungen der drei Grundachsen (A1,A2,A3) des Roboters bezeichnet. Grundachsen haben deutlich stärkeren Einfluss auf die Dynamik des Roboters als die Handachsen des Roboters, welche im Wesentlichen die Aufgabe besitzen die Orientierung des Werkzeugs einzustellen.

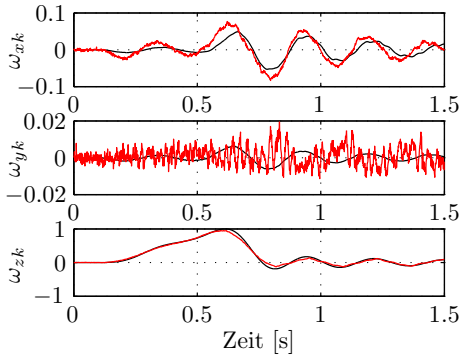
Für den KR210 wurden auch einige weitere Modellvarianten aufgestellt, in denen verschiedene andere Kombinationen der in Kapitel 2 vorgestellten Komponenten untersucht wurden. Hierzu zählen insbesondere Modelle mit Lose im Getriebe (nach Gleichung (2.12)) und drehzahlabhängigem Wirkungsgrad (Gl. (2.10)) sowie Modelle in welchen aufwendigere elastische Modelle für Arm und Schwinge verwendet wurden.

Die Lose der Antriebsstränge des Roboters ist dabei sehr klein, so dass auf die Modellierung verzichtet werden kann. Auch durch aufwendigere Modelle für Arm und Schwinge konnte keine deutliche Verbesserung in der Übereinstimmung mit den Messdaten erzielt werden. Durch die Modellierung eines drehzahlabhängigen Wirkungsgrades der Getriebe nach Gleichung (2.10) konnte eine Verbesserung von etwa 1 bis 4% (je nach *Case*) der Motormomente $\boldsymbol{\tau}_m$ erreicht werden. Numerisch ist die Modellierung jedoch kritisch, da beim Übergang von negativer zur positiver (oder umgekehrt) Kraftflussrichtung sehr steile Gradienten, aufgrund der zur Approximation eingesetzten Exponentialfunktionen, auftreten.

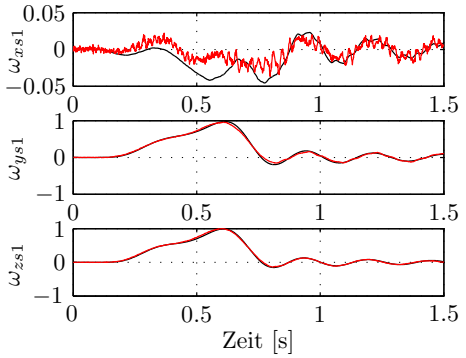
Da es auch hier das Hauptziel ist, Synthese-Modelle für die Invertierung in Echtzeit zu erstellen, stellen diese – relativ einfach gehaltenen – Modelle einen guten Kompromiss zwischen Genauigkeit und Rechenzeit dar.



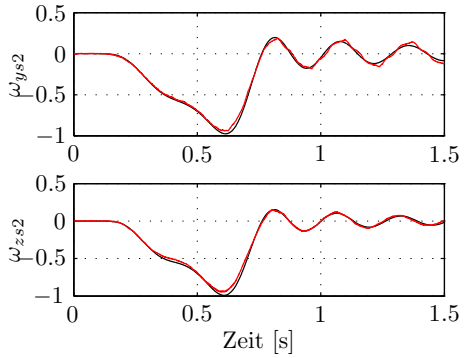
(a) $\tau_{m1}, \tau_{m2}, \tau_{m3}$, normiert auf $\tau_{max} = \max\{|\tau_{m1}|, |\tau_{m2}|, |\tau_{m3}|\}$



(b) $\omega_{xk}, \omega_{yk}, \omega_{zk}$ (Karussell), normiert auf $\omega_{max,k} = \max\{|\omega_{xk}|, |\omega_{yk}|, |\omega_{zk}|\}$

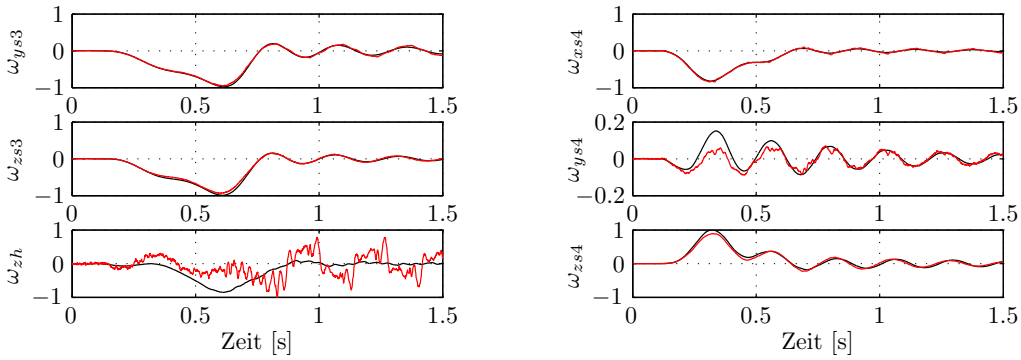


(c) $\omega_{xs1}, \omega_{ys1}, \omega_{zs1}$ (Schwinge Pos. 1), normiert auf $\omega_{max,s}$



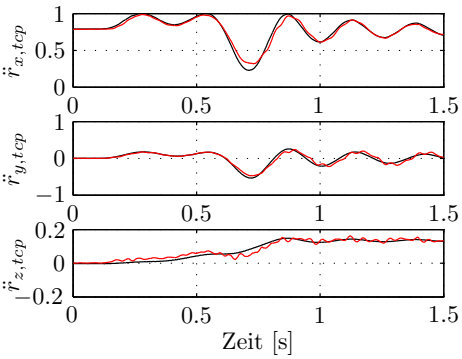
(d) $\omega_{ys2}, \omega_{zs2}$ (Schwinge Pos. 2), normiert auf $\omega_{max,s}$

Abbildung 4.6: Vergleich der gemessenen (rot) und simulierten (schwarz) Größen am KR210. Dargestellt sind die Daten von Drehratensensoren (normiert, Normierungsfaktor $\omega_{max,s} = \max\{|\omega_{xs1}|, ..., |\omega_{xs4}|, |\omega_{ys1}|, ..., |\omega_{ys1}|, |\omega_{zs1}|, ..., |\omega_{zs4}|\}$) sowie der Motormomente (normiert) der Grundachsen, jeweils über der Zeit t aufgetragen, bezüglich des jeweiligen lokalen Koordinatensystems. Anregung: Simultane 5° Bewegung der Grundachsen. Ausgangsstellung: $A1 = 0°, A2 = 0°, A3 = 0°$.

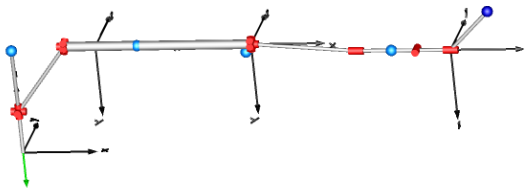


(a) $\omega_{ys3}, \omega_{zs4}$ (Schwinge Pos. 3), normiert auf $\omega_{max,s}$ sowie ω_{hz}, normiert auf $\omega_{h,max} = \max\{|\omega_{hz}|\}$

(b) $\omega_{xs4}, \omega_{ys4}, \omega_{zs4}$ (Schwinge Pos. 4), normiert auf $\omega_{max,s}$

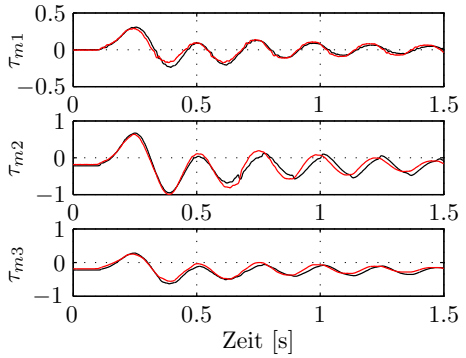


(c) $\ddot{r}_{x,tcp}, \ddot{r}_{y,tcp}, \ddot{r}_{z,tcp}$ (TCP), normiert auf $\ddot{r}_{max,tcp} = \max\{|\ddot{r}_{x,tcp}|, |\ddot{r}_{y,tcp}|, |\ddot{r}_{z,tcp}|\}$

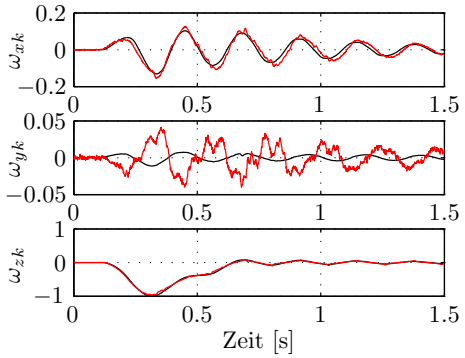


(d) Ausgangsstellung: $A1 = 0°, A2 = 0°, A3 = 0°$

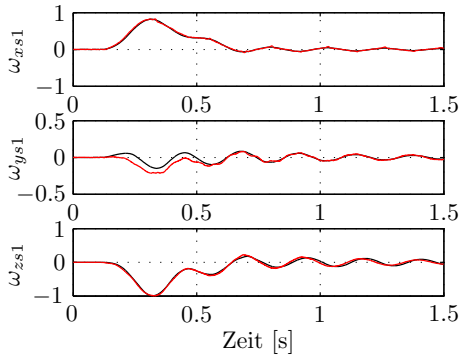
Abbildung 4.7: Vergleich der gemessenen (rot) und simulierten (schwarz) Größen am KR210. Dargestellt sind die Daten von Drehratensensoren (normiert, Normierungsfaktor $\omega_{max,s} = \max\{|\omega_{xs1}|, ..., |\omega_{xs4}|, |\omega_{ys1}|, ..., |\omega_{ys1}|, |\omega_{zs1}|, ..., |\omega_{zs4}|\}$) sowie der Motormomente (normiert) der Grundachsen und Beschleunigungsmessungen am TCP (normiert), jeweils über der Zeit t aufgetragen, bezüglich des jeweiligen lokalen Koordinatensystems. Anregung: Simultane 5° Bewegung der Grundachsen. Ausgangsstellung: $A1 = 0°, A2 = 0°, A3 = 0°$.



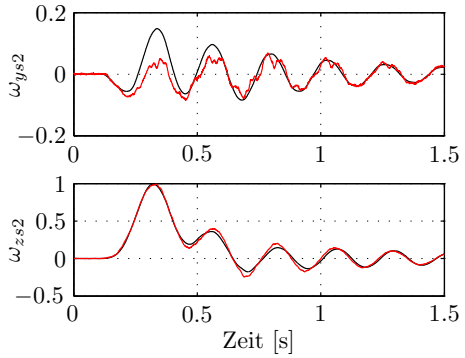
(a) $\tau_{m1}, \tau_{m2}, \tau_{m3}$, normiert auf $\tau_{max} = \max\{|\tau_{m1}|, |\tau_{m2}|, |\tau_{m3}|\}$



(b) $\omega_{xk}, \omega_{yk}, \omega_{zk}$ (Karussell), normiert auf $\omega_{max,k} = \max\{|\omega_{xk}|, |\omega_{yk}|, |\omega_{zk}|\}$

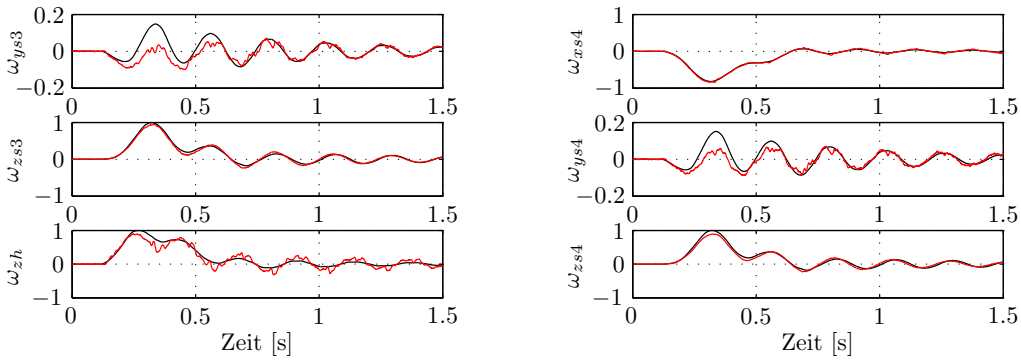


(c) $\omega_{xs1}, \omega_{ys1}, \omega_{zs1}$ (Schwinge Pos. 1), normiert auf $\omega_{max,s}$



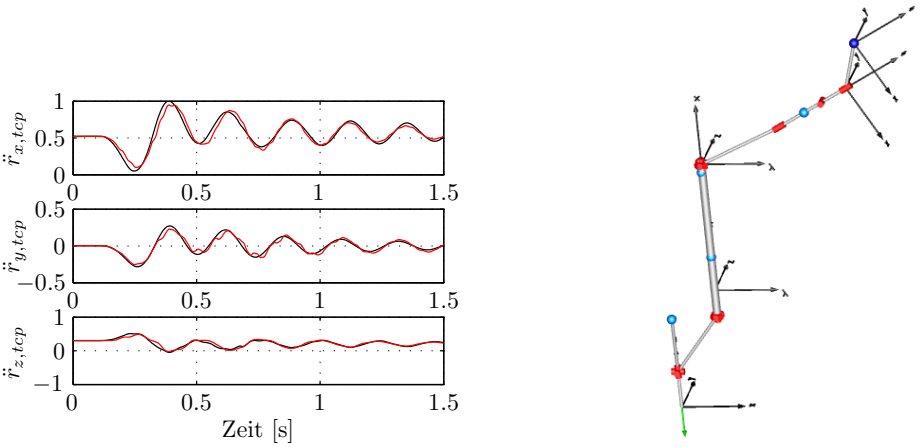
(d) $\omega_{ys2}, \omega_{zs2}$ (Schwinge Pos. 2), normiert auf $\omega_{max,s}$

Abbildung 4.8: Vergleich der gemessenen (rot) und simulierten (schwarz) Größen am KR210. Dargestellt sind die Daten von Drehratensensoren (normiert, Normierungsfaktor $\omega_{max,s} = \max\{|\omega_{xs1}|, ..., |\omega_{xs4}|, |\omega_{ys1}|, ..., |\omega_{ys1}|, |\omega_{zs1}|, ..., |\omega_{zs4}|\}$) sowie der Motormomente (normiert) der Grundachsen, jeweils über der Zeit t aufgetragen, bezüglich des jeweiligen lokalen Koordinatensystems. Anregung: Simultane 5° Bewegung der Grundachsen. Ausgangsstellung: $A1 = 0°, A2 = -90°, A3 = 60°$.



(a) $\omega_{ys3}, \omega_{zs4}$ (Schwinge Pos. 3), normiert auf $\omega_{max,s}$ sowie ω_{hz}, normiert auf $\omega_{h,max} = \max\{|\omega_{hz}|\}$

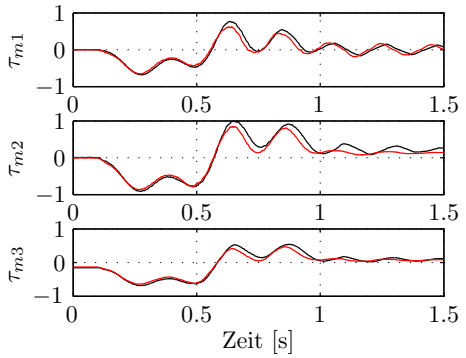
(b) $\omega_{xs4}, \omega_{ys4}, \omega_{zs4}$ (Schwinge Pos. 4), normiert auf $\omega_{max,s}$



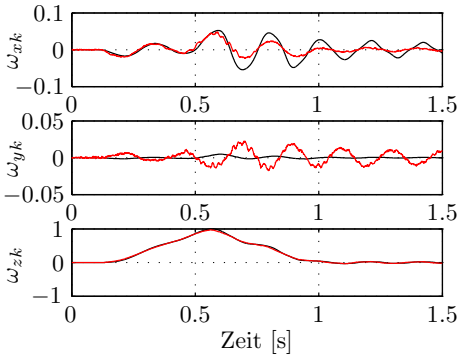
(c) $\ddot{r}_{x,tcp}, \ddot{r}_{y,tcp}, \ddot{r}_{z,tcp}$ (TCP), normiert auf $\ddot{r}_{max,tcp} = \max\{|\ddot{r}_{x,tcp}|, |\ddot{r}_{y,tcp}|, |\ddot{r}_{z,tcp}|\}$

(d) Ausgangsstellung: $A1 = 0°, A2 = -90°, A3 = 60°$

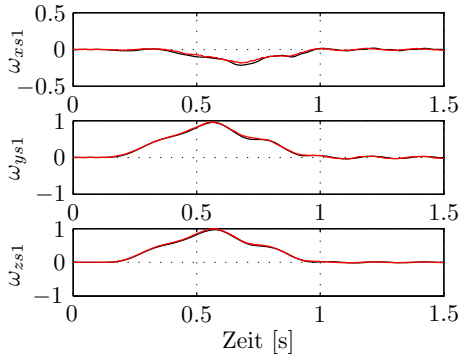
Abbildung 4.9: Vergleich der gemessenen (rot) und simulierten (schwarz) Größen am KR210. Dargestellt sind die Daten von Drehratensensoren (normiert, Normierungsfaktor $\omega_{max,s} = \max\{|\omega_{xs1}|, ..., |\omega_{xs4}|, |\omega_{ys1}|, ..., |\omega_{ys1}|, |\omega_{zs1}|, ..., |\omega_{zs4}|\}$) sowie der Motormomente (normiert) der Grundachsen und Beschleunigungsmessungen am TCP (normiert), jeweils über der Zeit t aufgetragen, bezüglich des jeweiligen lokalen Koordinatensystems. Anregung: Simultane 5° Bewegung der Grundachsen. Ausgangsstellung: $A1 = 0°, A2 = -90°, A3 = 60°$.



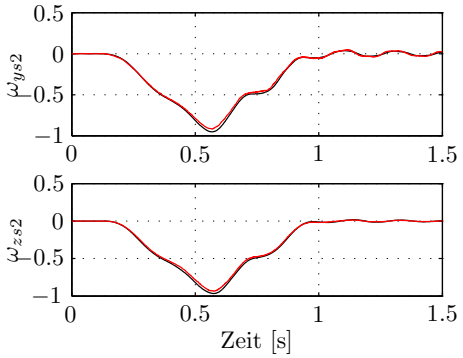
(a) $\tau_{m1}, \tau_{m2}, \tau_{m3}$, normiert auf $\tau_{max} = \max\{|\tau_{m1}|, |\tau_{m2}|, |\tau_{m3}|\}$



(b) $\omega_{xk}, \omega_{yk}, \omega_{zk}$ (Karussell), normiert auf $\omega_{max,k} = \max\{|\omega_{xk}|, |\omega_{yk}|, |\omega_{zk}|\}$

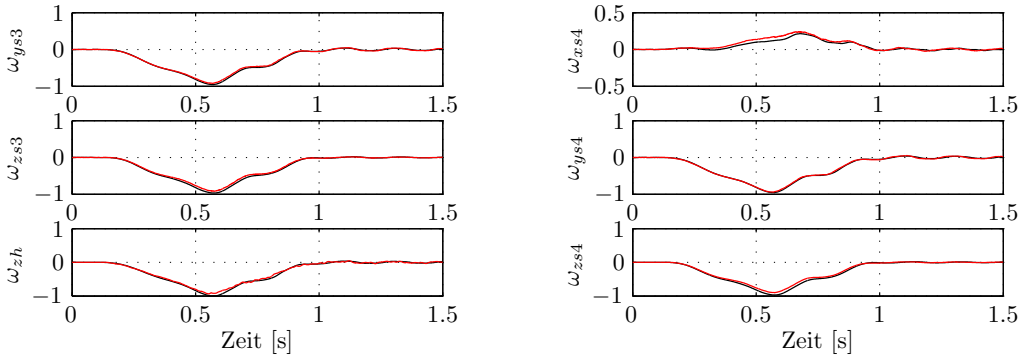


(c) $\omega_{xs1}, \omega_{ys1}, \omega_{zs1}$ (Schwinge Pos. 1), normiert auf $\omega_{max,s}$



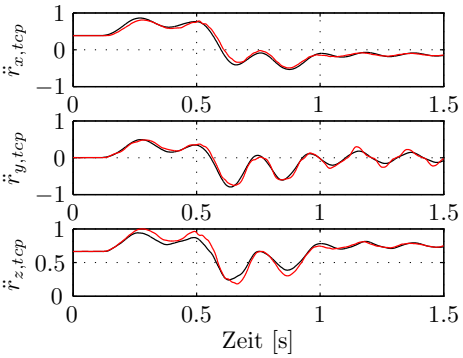
(d) $\omega_{ys2}, \omega_{zs2}$ (Schwinge Pos. 2), normiert auf $\omega_{max,s}$

Abbildung 4.10: Vergleich der gemessenen (rot) und simulierten (schwarz) Größen am KR210. Dargestellt sind die Daten von Drehratensensoren (normiert, Normierungsfaktor $\omega_{max,s} = \max\{|\omega_{xs1}|, ..., |\omega_{xs4}|, |\omega_{ys1}|, ..., |\omega_{ys1}|, |\omega_{zs1}|, ..., |\omega_{zs4}|\}$) sowie der Motormomente (normiert) der Grundachsen, jeweils über der Zeit t aufgetragen, bezüglich des jeweiligen lokalen Koordinatensystems. Anregung: Simultane 5° Bewegung der Grundachsen. Ausgangsstellung: $A1 = 0°, A2 = -90°, A3 = 60°$.

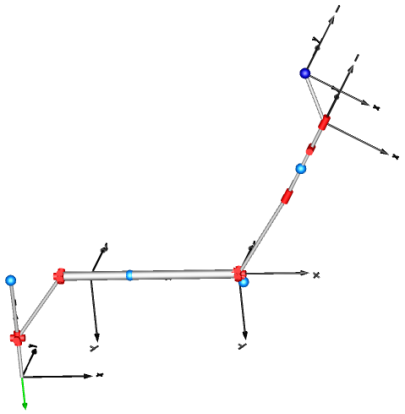


(a) $\omega_{ys3}, \omega_{zs4}$ (Schwinge Pos. 3), normiert auf $\omega_{max,s}$ sowie ω_{hz}, normiert auf $\omega_{h,max} = \max\{|\omega_{hz}|\}$

(b) $\omega_{xs4}, \omega_{ys4}, \omega_{zs4}$ (Schwinge Pos. 4), normiert auf $\omega_{max,s}$



(c) $\ddot{r}_{x,tcp}, \ddot{r}_{y,tcp}, \ddot{r}_{z,tcp}$ (TCP), normiert auf $\ddot{r}_{max,tcp} = \max\{|\ddot{r}_{x,tcp}|, |\ddot{r}_{y,tcp}|, |\ddot{r}_{z,tcp}|\}$



(d) Ausgangsstellung: $A1 = 0°, A2 = 0°, A3 = -60°$

Abbildung 4.11: Vergleich der gemessenen (rot) und simulierten (schwarz) Größen am KR210. Dargestellt sind die Daten von Drehratensensoren (normiert, Normierungsfaktor $\omega_{max,s} = \max\{|\omega_{xs1}|, ..., |\omega_{xs4}|, |\omega_{ys1}|, ..., |\omega_{ys1}|, |\omega_{zs1}|, ..., |\omega_{zs4}|\}$) sowie der Motormomente (normiert) der Grundachsen und Beschleunigungsmessungen am TCP (normiert), jeweils über der Zeit t aufgetragen, bezüglich des jeweiligen lokalen Koordinatensystems. Anregung: Simultane 20° Bewegung der Grundachsen. Ausgangsstellung: $A1 = 0°, A2 = 0°, A3 = -60°$.

Kapitel 5

Regelungskonzepte und Anwendungsmöglichkeiten von inversen Robotermodellen

In diesem Kapitel sollen ausgewählte Konzepte vorgestellt werden wie ein inverses Modell als Teil einer Regelung eines Roboters verwendet werden kann und welche zusätzlichen Anwendungen möglich sind.

5.1 Regelung mit zwei Freiheitsgraden

Eine Möglichkeit, wie ein inverses Modell zur Regelung eines Roboters eingesetzt werden kann, ist das Konzept der zwei Freiheitsgrade (siehe auch [105, 106]). Hierbei dient das inverse Modell als Vorsteuerung. Ein Schema des Konzepts zeigt Abb. 5.1.

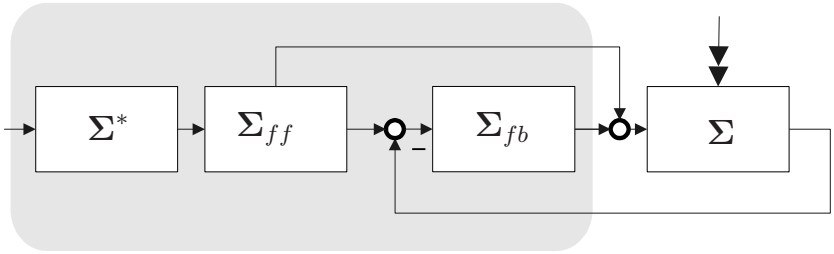


Abbildung 5.1: Konzept der Regelung mit zwei Freiheitsgraden. Die Sollgrößen werden über einen Vorfilter $\mathbf{\Sigma}^*$ aufbereitet und an eine Vorsteuerung $\mathbf{\Sigma}_{ff}$ übergeben. Diese generiert die Soll-Steuergrößen (beim Roboter Vorsteuermomente) und Soll-Messgrößen (beim Roboter Soll-Motorposition und Soll-Motorgeschwindigkeit). Ein Regler $\mathbf{\Sigma}_{fb}$ dient dazu vorhanden Störungen und Modellfehler auszugleichen. $\mathbf{\Sigma}$ stellt die (gestörte) Regelstrecke dar.

Von einer Bahnplanung werden die gewünschten Solltrajektorien an das inverse Modell übergeben. Im inversen Modell werden die Soll-Steuergrößen und Soll-Messgrößen berechnet. Wenn Strecke und inverses Systems exakt übereinstimmen, beim selben Zustand starten, das System stabil ist und es keine Störgrößen gibt, dann ist der Re-

gelfehler mit dieser Struktur bereits nur mit der Vorsteuerung Null und es würde kein Regler benötigt (idealer Fall).

Um dies zu veranschaulichen soll als Beispielsystem ein linearer Zwei-Massen-Torsionsschwinger nach Gl. (5.1) betrachtet werden mit motorseitiger Trägheit Θ_m, abtriebsseitiger Trägheit Θ_a und Konstanten des Feder-Dämpfersystems c, d. Stellgröße ist das Motormoment τ_m und Messgröße der Motorwinkel φ_m. Ziel soll es sein den abtriebsseitigen Winkel φ_a mit einer Regelung mit zwei Freiheitsgraden vorzugeben.

$$\begin{pmatrix} \dot{\varphi}_a \\ \ddot{\varphi}_a \\ \dot{\varphi}_m \\ \ddot{\varphi}_m \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -\frac{c}{\Theta_a} & -\frac{d}{\Theta_a} & \frac{c}{\Theta_a} & \frac{d}{\Theta_a} \\ 0 & 0 & 0 & 1 \\ \frac{c}{\Theta_m} & \frac{d}{\Theta_m} & -\frac{c}{\Theta_m} & -\frac{d}{\Theta_m} \end{pmatrix} \begin{pmatrix} \varphi_a \\ \dot{\varphi}_a \\ \varphi_m \\ \dot{\varphi}_m \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{\Theta_m} \end{pmatrix} \tau_m \tag{5.1}$$

Zur Berechnung des inversen Systems wird die dritte Ableitung von φ_a als Eingang benötigt da der relative Grad des Systems (5.1) von $\tau_m(s)$ nach $\varphi_a(s)$ drei beträgt (siehe Gl. (5.2) mit Laplace-Variablen s).

$$G_{\tau_m \to \varphi_a}(s) = \frac{ds + c}{s^2 \left(\Theta_m \Theta_a s^2 + \Theta_m d s + d\Theta_a s + \Theta_m c + c\Theta_a\right)} \tag{5.2}$$

Mit den in Abschnitt 3.4 vorgestellten Algorithmen lässt sich direkt ein inverses System (Gl. (5.3)) ableiten. Der Zustand τ_{FD} in Gl. (5.3) entspricht dem Feder-Dämpfer-Schnittmoment. Da die Regelgröße φ_a, Stellgröße τ_m und Messgröße φ_m unterschiedlich sind, muss das inverse System sowohl Soll-Stellgröße und Soll-Messgröße berechnen.

$$\begin{pmatrix} \dot{\varphi}_a \\ \ddot{\varphi}_a \\ \dot{\varphi}_m \\ \ddot{\varphi}_m \\ \dot{\tau}_{FD} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{\Theta_a} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & \frac{c}{d} & 0 & -\frac{c}{d} & -\frac{1}{\Theta_a} \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \varphi_a \\ \dot{\varphi}_a \\ \varphi_m \\ \dot{\varphi}_m \\ \tau_{FD} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{\Theta_a}{d} \\ -\Theta_a \end{pmatrix} \dddot{\varphi}_a \tag{5.3a}$$

$$\begin{pmatrix} \tau_m \\ \varphi_m \end{pmatrix} = \begin{pmatrix} 0 & \frac{\Theta_m c}{d} & 0 & -\frac{\Theta_m c}{d} & -\frac{\Theta_m}{\Theta_a} - 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \varphi_a \\ \dot{\varphi}_a \\ \varphi_m \\ \dot{\varphi}_m \\ \tau_{FD} \end{pmatrix} + \begin{pmatrix} \frac{\Theta_a \Theta_m}{d} \\ 0 \end{pmatrix} \dddot{\varphi}_a \tag{5.3b}$$

Die dritte Ableitung von $\varphi_{a,soll}$ kann über einen Vorfilter $\mathbf{\Sigma}^*$ berechnet werden. Für das Beispiel wird ein kritisch gedämpfter Filter dritter Ordnung (wobei $\omega_f = 2\pi f_c$) verwendet (siehe Abschnitt 3.5). Wie man anhand von Gleichung (5.4) erkennt sind die Übertragungsfunktionen $G_{soll}(s)$ von Sollgröße $\varphi_{a,soll}(s)$ zu Soll-Messgröße $\varphi_{m,soll}(s)$ und die Übertragungsfunktion $G_{ist}(s)$ von Regelgröße $\varphi_{a,soll}(s)$ zu Ist-Messgröße $\varphi_{m,ist}(s)$ identisch. Es tritt daher kein Regelfehler auf wenn Strecke und inverses Modell exakt

übereinstimmen.

$$G_{soll}(s) = \underbrace{\frac{{\omega_f}^3}{(s+\omega_f)^3}}_{Filter} \underbrace{s^3}_{Ableitung} \underbrace{\frac{\Theta_a\, s^2 + ds + c}{s^3\,(ds+c)}}_{\ddot{\dddot{\varphi}}_{a,soll} \rightarrow \varphi_{m,soll}} = \frac{{\omega_f}^3\,(\Theta_a\, s^2 + ds + c)}{(s+\omega_f)^3\,(ds+c)} \tag{5.4a}$$

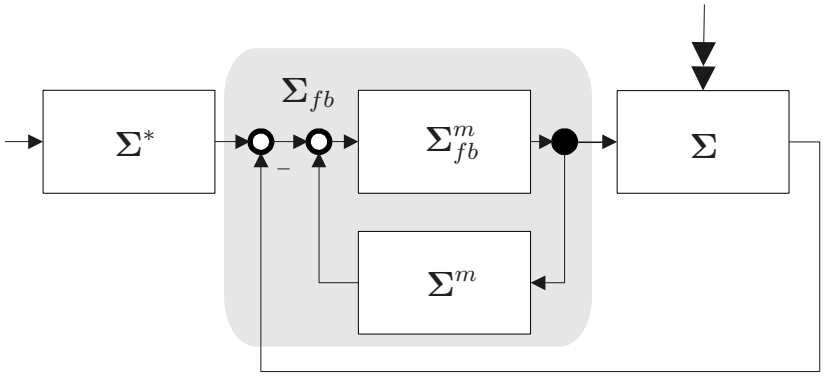
$$G_{ist}(s) = \underbrace{\frac{{\omega_f}^3}{(s+\omega_f)^3}}_{Filter} \underbrace{s^3}_{Ableitung} \underbrace{\frac{\Theta_m\, s^2 \Theta_a + \Theta_m\, sd + \Theta_m\, c + d\Theta_a\, s + c\Theta_a}{s\,(ds+c)}}_{\ddot{\dddot{\varphi}}_{a,soll} \rightarrow \tau_{m,soll}} \tag{5.4b}$$

$$\cdot \underbrace{\frac{\Theta_a\, s^2 + ds + c}{s^2\,(\Theta_m\, s^2 \Theta_a + \Theta_m\, sd + \Theta_m\, c + d\Theta_a\, s + c\Theta_a)}}_{Strecke: \tau_m \rightarrow \varphi_{m,ist}} = \frac{{\omega_f}^3\,(\Theta_a\, s^2 + ds + c)}{(s+\omega_f)^3\,(ds+c)}$$

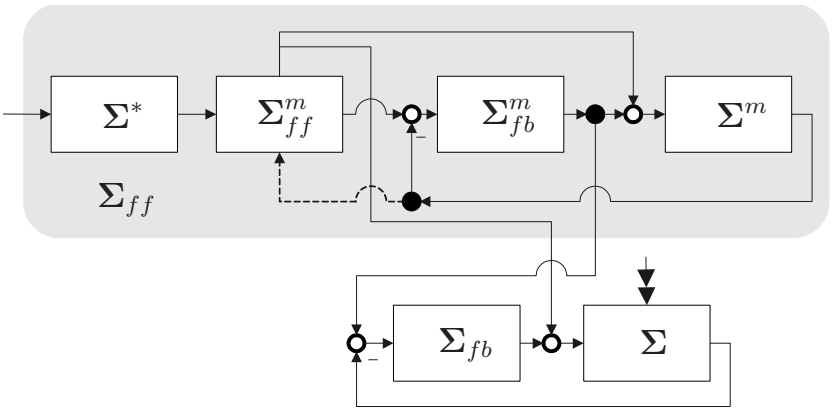
Durch die Verwendung des inversen Modells kann der Regler daher stärker dahingehend ausgelegt werden Störungen zu dämpfen und im Falle eines instabilen Systems, das System zu stabilisieren. Das Führungsverhalten der Regelungsstruktur wird durch die Vorsteuerung bestimmt. Durch die Vorsteuerung können bereits viele nichtlineare Effekte kompensiert werden, was einen Regler zusätzlich entlastet. Hierdurch kann zur Auslegung eines (modellbasierten) Reglers ein einfacheres (z. B. lineares) mathematisches Ersatzmodell des Roboters verwendet werden. Dies erleichtert die Reglersynthese und es können einfache, robuste Regler verwendet werden.

5.2 Regelung innerhalb der Vorsteuerung

Wie in den vorangegangenen Kapiteln beschrieben, sind Modelle von Robotern mit elastischen Strukturteilen, aufgrund ihrer instabilen Nulldynamik, nicht exakt invertierbar. Es können zur Vorsteuerung nur approximative inverse Modelle verwendet werden. Eine Möglichkeit, um den Approximationsfehler in der Vorsteuerung zu verringern ist es, bereits dort eine zusätzliche Rückführungs-Schleife in Kombination mit einem (exakteren) Vorwärtsmodell zu berechnen, daher bereits in der Vorsteuerung eine Regelungsschleife zu verwenden. Der Vorteil dieser Rückführungs-Schleife in der Vorsteuerung ist, dass dort alle System-Größen „virtuell messbar“ (berechenbar) sind und daher nahezu beliebige Regelungsmethoden angewandt werden können. Diese Vorgehensweise ist ähnlich zu dem Regelungskonzepts „Internal Model Control“ (siehe [107]). Beim IMC wird das Streckenmodell $\mathbf{\Sigma}^m$ durch einen modellbasierten Regler $\mathbf{\Sigma}^m_{fb}$ geregelt und dadurch im Prinzip invertiert (unter der Annahme, dass der Regler perfekt ist und keine Abweichungen von Soll- und Istgrößen auftreten). Abbildung 5.2(a) zeigt die Struktur von IMC. Die Vorgehensweise von IMC lässt sich auch auf eine Vorsteuerung $\mathbf{\Sigma}_{ff}$ anwenden. Dies zeigt Abb. 5.2(b). Hierbei wird bereits innerhalb der Vorsteuerung $\mathbf{\Sigma}_{ff}$ das Konzept der zwei Freiheitsgrade angewendet um eventuelle Fehler, welche durch eine approximative Invertierung der Strecke auftreten, durch einen zusätzlichen Regler $\mathbf{\Sigma}^m_{fb}$ auszugleichen. Da dieser Regler rein in der Simulation arbeitet, sind nahezu beliebige Reglertypen einsetzbar, da in der Simulation beliebige Zustandsgrößen berechenbar sind, welche bei der eigentlichen Strecke $\mathbf{\Sigma}$ nicht messbar wären. Durch diese zusätzliche Korrektur der modellbasierten Vorsteuerung $\mathbf{\Sigma}^m_{ff}$ durch den Regler $\mathbf{\Sigma}^m_{fb}$ können kleine Approximationsfehler ausgeglichen werden, oder die Vorsteuerung $\mathbf{\Sigma}^m_{ff}$ kann zusätzlich adaptiert werden (angedeutet durch die gestrichelte



(a) IMC - Internal Model Control: Das Modell der Strecke $\mathbf{\Sigma}^m$ und der modellbasierte Regler $\mathbf{\Sigma}^m_{fb}$ bilden zusammen den Regler $\mathbf{\Sigma}_{fb}$.



(b) Regelung innerhalb der Vorsteuerung: Die Vorsteuerung $\mathbf{\Sigma}_{ff}$ besteht aus einem modellbasierten System das nach dem Konzept der Regelung mit zwei Freiheitsgraden geregelt wird. Die aus dieser modellbasierten Regelung gewonnen Signale dienen zum Adaptieren der Vorsteuerung $\mathbf{\Sigma}^m_{ff}$ sowie zur Generierung der Sollgrößen für das eigentliche (reale) geregelte System, welches aus dem (robusten) Regler $\mathbf{\Sigma}_{fb}$ und der (gestörten) Strecke $\mathbf{\Sigma}$ besteht.

Abbildung 5.2: Unterschiedliche Konzepte zur modellbasierten Regelung, jeweils mit Vorfilter $\mathbf{\Sigma}^*$. Ein hochgestelltes m steht jeweils für modellbasierte oder simulierte Teilsysteme.

Linie in Abb. 5.2(b)), um systematische Fehler zu eliminieren. $\mathbf{\Sigma}^m$ stellt hierbei ein sehr exaktes (ungestörtes) Streckenmodell dar, welches – da es nicht invertiert werden muss – beliebige (nicht differenzierbare) Nichtlinearitäten (wie etwa exakte Haftreibung mit Zustandsumschaltung) oder instabile Nulldynamik enthalten kann.

In der eigentlichen Rückführungs-Schleife der Regelung $\mathbf{\Sigma}_{fb}$ (außerhalb der Vorsteuerung) können lediglich Algorithmen verwendet werden, welche auf die vorhandenen (motorseitigen) Messgrößen ($\boldsymbol{\varphi}_m$ und $\boldsymbol{I}_m$) aufbauen oder es müssen zusätzlich Beobachter verwendet werden, wobei hierbei auch die Beobachtbarkeitsbedingungen für die zusätzlich verwendeten Größen erfüllt sein müssen.

Als Regler in der Vorsteuerung kommen unterschiedlichste Verfahren in Frage. Die Verfahren sollten nach Möglichkeit so gewählt werden, dass sie Vorteil daraus ziehen, dass beliebige Größen messbar sind. Zudem ist eher ihr Führungsverhalten (Effizienz in der Minimierung von Abweichung von den Sollgrößen) von Bedeutung, weniger ihre Robustheit, da keine Störungen auftreten und das dort verwendete (simulierte)

Streckenmodell $\boldsymbol{\Sigma}^m$ als sehr genau bekannt angenommen werden kann.

5.2.1 Inverse Disturbance Observer als kartesischer Regler in der Vorsteuerung

Für einen Regler innerhalb der Vorsteuerung kommen prinzipiell beliebige Verfahren in Frage. Sehr gut geeignet sind etwa Zustandsregler sowie ein Inverse Disturbance Observer (IDOB) (siehe [108]) und/oder kartesische Regelungsansätze oder nichtlineare modellprädiktive(NL-MPC[1]) Regelungsansätze (siehe z. B. [109]).

Verschiedene Verfahren wurden untersucht, eine Variante des IDOB als kartesischer Regler hat sich im untersuchten Fall als gute Lösung erwiesen. Bei der Auslegung eines Zustandsreglers ist es problematisch eine Lösung zu finden, die bei dem stark nichtlinearen Robotermodell in beliebigen Stellungen den Approximationsfehler minimiert und stabil bleibt, wohingegen der IDOB deutlich besser für das stark nichtlineare Modell eingestellt werden kann. Eine Alternative wäre ein adaptiver, stellungsabhängig parametrisierter Zustandsregler (siehe [24] und [10]), wobei kein Beobachter gebraucht wird, da alle Größen berechenbar sind.

Der IDOB stellt eine Modifikation des Disturbance Observers dar. Im Vergleich zur Struktur eines Disturbance Observers (DOB) [110] sind die Blöcke der Regelstrecke und des invertierten nominalen Modells gegeneinander ausgetauscht. Der IDOB benutzt die Eigenschaft der hohen Regelverstärkung mit dem Ziel, den Ausgang der Regelstrecke einem Sollwert anzugleichen.

Der IDOB baut auf einem approximativen inversen Modell $(\boldsymbol{\Sigma}^m)^{-1}$ der Strecke $\boldsymbol{\Sigma}^m$ auf. Fehler von $(\boldsymbol{\Sigma}^m)^{-1}$ werden über eine Rückführungs-Schleife über einen Filter $\boldsymbol{Q}$ zurückgeführt. Der Filter hat die Aufgabe Abweichungen in niederfrequenten Bereich, im Sinne eines „infinity-gain feedbacks“, zurück zu führen, wohingegen hochfrequente Anteile des Fehlers $\boldsymbol{\epsilon}$ gedämpft werden, um die Stabilität des Systems zu gewährleisten. Eine ausführliche Beschreibung des IDOB befindet sich in [108].

Der IDOB wurde für lineare Systeme entwickelt, kann jedoch auch für nichtlineare Systeme angewandt werden. Hierbei ist allerdings keine direkte mathematische Regel für die Eckfrequenzen $f_{c,i}$ der Filter $\boldsymbol{Q} = \{Q_1, \ldots, Q_n\}$ angebbar, wie dies im linearen Fall möglich ist. Da die Güte des approximativen inversen Modells der Strecke $(\boldsymbol{\Sigma}^m)^{-1}$ nicht über den gesamten Arbeitsraums des Roboters identisch ist, müssen die Eckfrequenzen $f_{c,i}$ der einzelnen Filter konservativer als im linearen Fall gewählt werden, um Stabilität im kompletten Arbeitsbereich zu gewährleisten. Da der IDOB in der Vorsteuerung eingesetzt wird, können zusätzliche Störungen vernachlässigt werden. Die Struktur des Systems mit IDOB und Regler $\boldsymbol{\Sigma}_{fb}$ zeigt Abbildung 5.3. Ziel des Aufbaus ist es den TCP exakt nach einer Sollvorgabe zu führen. In der Simulation wurde nur die Position betrachtet, die Orientierung wurde vernachlässigt. Durch die Entkopplung von Position und Orientierung bei einem Roboter kann die Position durch die Grundachsen (Achse 1 bis 3) und die Orientierung ausschließlich durch die Handachsen (Achse 4 bis 6) eingestellt werden (dies ist für den elastischen Roboter nur approximativ erfüllt, aber eine gute Näherung).

[1] Nonlinear Model Predictive Control

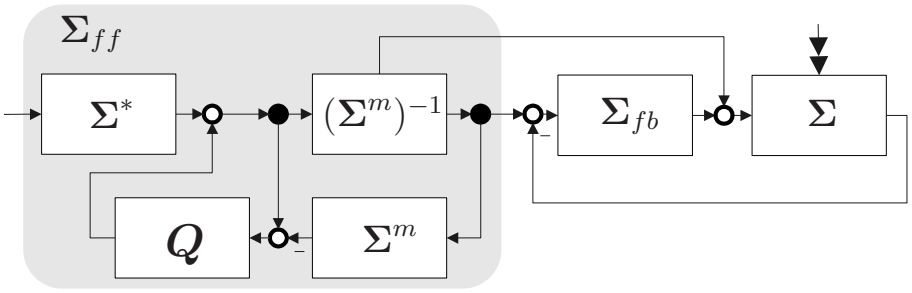


Abbildung 5.3: IDOB als Teil der Vorsteuerung mit simulierter Strecke $\boldsymbol{\Sigma}^m$, realer (gestörter) Strecke $\boldsymbol{\Sigma}$, approximativen inversen Streckenmodell $(\boldsymbol{\Sigma}^m)^{-1}$, sowie einem zusätzlichen Regler $\boldsymbol{\Sigma}_{fb}$ und IDOB Filteranordnung $\boldsymbol{Q}$ und Vorfilter $\boldsymbol{\Sigma}^*$.

Die Sollvorgabe geschieht daher für den Handwurzelpunkt, direkt vor Achse 4. Als „virtuelle Messgrößen“ kommen motorseitige-, abtriebsseitige oder kartesische Größen in Frage. In der Simulation wurden alle drei Varianten getestet. Die besten Ergebnisse ließen sich mit der kartesischen Beschleunigung am Handwurzelpunkt als „virtuelle Messgröße“ erreichen.

Die Beschleunigung ist besonders geeignet, da sie eine sehr empfindliche Messgröße darstellt. Anhand der Beschleunigung $\ddot{\mathbf{r}}_{HWP,t}$ können bereits kleinste Abweichungen erkannt werden. Im IDOB wurden dabei drei Bessel-Filter Q_1, Q_2, Q_3 als Filteranordnung $\boldsymbol{Q}$ verwendet, für die drei räumlichen kartesischen Abweichungen. Es wurden Filter 4. Ordnung gewählt, ihre jeweiligen Eckfrequenzen $f_{c,i}$ wurden durch eine Optimierung[2] ermittelt.

Im Modell erzeugt eine Bahnplanung die Trajektorien[3] für das approximative inverse Modell $(\boldsymbol{\Sigma}^m)^{-1}$. Die hierin berechneten Motormomente $\hat{\boldsymbol{\tau}}_m$ dienen als Eingangsgröße für das Robotermodell $\boldsymbol{\Sigma}^m$. Abweichungen $\boldsymbol{\epsilon}$ der kartesischen Beschleunigung am Handwurzelpunkt (HWP) $\ddot{\mathbf{r}}_{HWP,t}$ zwischen den Sollwerten und den simulierten Größen in $\boldsymbol{\Sigma}^m$ dienen komponentenweise (x, y, z) als Eingangsgrößen der Besselfilter Q_i. Die Ausgänge der Filter werden zweimal integriert, um zu einer Positionskorrektur $\Delta\mathbf{r}_{HWP,t}$ zu gelangen. Diese Korrektur wird im inversen Modell $(\boldsymbol{\Sigma}^m)^{-1}$ zu den Sollwerten addiert, um den Fehler auszugleichen. Die daraus resultierenden korrigierten Motormomente und Motorwinkel dienen dann als neue Sollwerte für das Robotermodell $\boldsymbol{\Sigma}^m$ und die geregelte Strecke $\boldsymbol{\Sigma}$.

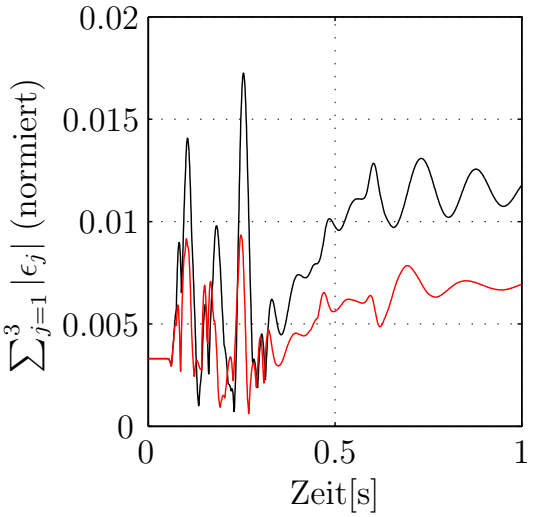
$(\boldsymbol{\Sigma}^m)^{-1}$ in Kombination mit den Filtern Q_i und der Strecke $\boldsymbol{\Sigma}^m$ können als Vorsteuerung für $\boldsymbol{\Sigma}$ angesehen werden.

Abbildung 5.4 zeigt Simulationsergebnisse mit und ohne IDOB. Da in der Simulation die Strecke als exakt bekannt angenommen wird gilt hierbei $\boldsymbol{\Sigma}^m \equiv \boldsymbol{\Sigma}$. Die Abweichungen sind in beiden Fällen sehr klein[4], jedoch kann mit Hilfe des kartesischen IDOB die Abweichung zu den Sollwerten, im Vergleich zur Simulation nur mit dem

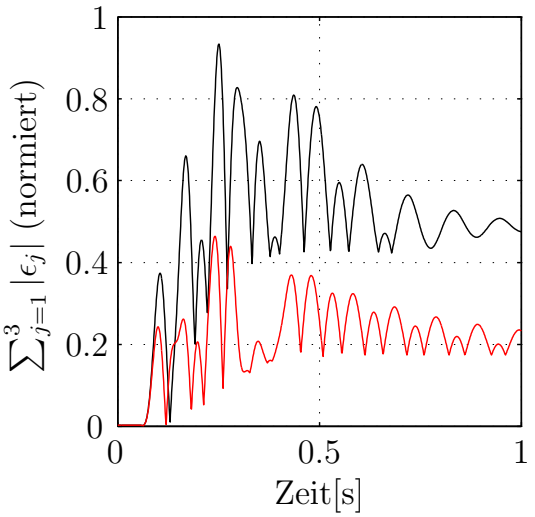
[2]Die Optimierung erfolgte über nichtlineare Simulationen mit *Modelica/Dymola* in Kombination mit *MOPS* (siehe A.7) und dem Pattern-Algorithmus. Gütefunktion war die Summe über den quadratischen kartesischen Positionsfehler für verschiedene Stellungen, über den Arbeitsraum verteilt.

[3]Ideale abtriebsseitige Winkel $\boldsymbol{\varphi}_{a,r}(t)$ für die drei Grundachsen

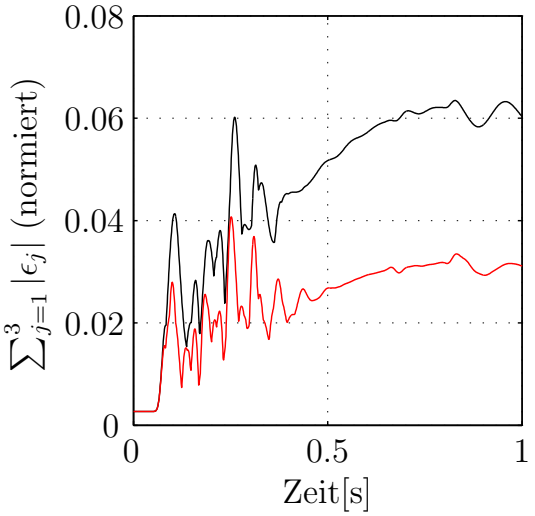
[4]Anmerkung: Für die inverse Kinematik im inversen Robotermodell wurde hierbei die direkte Lösung nach Abschnitt 3.6.1 gewählt und die Sollbahn so gewählt, dass hierfür immer Lösungen existieren. Daher ist die Genauigkeit höher als bei der Verwendung des (robusteren) DLS-Algorithmus.



(a) Normierter Positionsfehler mit (rot) und ohne (schwarz) IDOB in der Vorsteuerung. Anregung simultane 5° Bewegung der Grundachsen aus der Strecklage.



(b) Normierter Positionsfehler mit (rot) und ohne (schwarz) IDOB in der Vorsteuerung. Anregung simultane 5° Bewegung der Grundachsen aus der Kanonenstellung.



(c) Normierter Positionsfehler mit (rot) und ohne (schwarz) IDOB in der Vorsteuerung. Ausgangsstellung {A1:0°,A2:-30°,A3:30°,A4:0°,A5:0°,A6:0°}. Anregung: PTP-Relativbewegung {A1:5°,A2:30°,A3:-30°}.

Abbildung 5.4: Simulationsergebnisse mit und ohne IDOB für simultane Bewegungen der drei Grundachsen (Dreiachsbewegungen) eines elastischen Roboters (KR16) in unterschiedlichen Stellungen. Gezeigt ist die (normierte) Summe der kartesischen Positionsfehler (über alle drei Raumrichtungen) am HWP ($\sum_{j=1}^{j=3} |\epsilon_j|$). Durch die zusätzliche Regelungsschleife in der Vorsteuerung kann der kartesische Positionsfehler gegenüber der reinen Vorsteuerung noch einmal verkleinert werden. Es ist allerdings ein deutlich größerer Rechenaufwand nötig.

approximativen inversen Modell $(\mathbf{\Sigma}^m)^{-1}$, noch einmal verkleinert werden. Als Nachteil muss allerdings die hohe Rechenzeit angemerkt werden, da sowohl das inverse Modell $(\mathbf{\Sigma}^m)^{-1}$ als auch das geregelte Streckenmodell $\mathbf{\Sigma}^m$ simuliert werden muss, um zu der Vorsteuerung $\mathbf{\Sigma}_{ff}$ zu gelangen. Um eine Verbesserung gegenüber der ausschließlichen Verwendung von $(\mathbf{\Sigma}^m)^{-1}$ zu erzielen muss zudem ein sehr genaues Streckenmodell $\mathbf{\Sigma}^m$ verwendet werden. Für heutige Steuerungsrechner übersteigt dies meist die verfügbare Rechenzeit. Alternativ kann diese Methode zur Offline-Berechnung von Trajektorien verwendet werden, bei welchen die Rechenzeit keine Rolle spielt. Oder für zukünftige Vorsteuerungen, bei denen – aller Voraussicht nach – mehr Rechenzeit verfügbar sein wird.

5.3 Trajektorien-Optimierung basierend auf inversen Modellen

Die erstellten Robotermodelle können zur Optimierung von Trajektorien, welche der Roboter z. B. mit einem Werkzeug abfahren soll, verwendet werden. Bei vielen Anwendungen dieser Art ist die Lage der Trajektorie $\mathfrak{T}$ im Raum, bis auf eine geringe Toleranz, fest vorgegeben (z. B. wegen Hindernissen oder aufgrund der Applikation), allerdings ist die Geschwindigkeit auf der Trajektorie frei vorgebbar. Zur Optimierung von Trajektorie dieser Art, basierend auf Robotermodellen, gibt es zwei grundsätzliche Vorgehensweisen:

1. Die Motorwinkel (oder Motormomente) der Roboterachsen, als Funktion der Zeit, sind die zu optimierende Steuerungsfunktion. Als Modell wird ein Vorwärtsmodell $\mathbf{\Sigma}^m$ des Roboters verwendet. In diesem Fall muss die gewünschte Trajektorie, sowie die Monotonie[5] in Form von Nebenbedingungen oder Kriterien der Optimierung erzwungen werden.

2. Die gewünschte Trajektorie wird als Eingang für ein inverses Robotermodell $(\mathbf{\Sigma}^m)^{-1}$ verwendet, wobei lediglich die Geschwindigkeit auf der Bahn optimiert wird. Motorwinkel und Motormomente sind die Ausgänge des inversen Modells. Deren Realisierbarkeit muss über Kriterien oder Nebenbedingungen in der Optimierung sicher gestellt werden.

Die erste Variante bedeutet einen erheblich größeren Freiheitsgrad, der vom Optimierer bewältigt werden muss. Durch die Elastizität des Roboters muss der Optimierer hierbei dafür sorgen, dass die generierten Motorwinkelverläufe den Roboter nicht zu Schwingungen anregen. Diese könnten unter Umständen die Toleranz für Abweichungen von der gewünschten Trajektorie $\mathfrak{T}$ überschreiten. Zudem müssen die Motoren des Roboters in der Lage sein, den gewünschten Verlauf zu realisieren, was durch zusätzliche Nebenbedingungen, bzw. Beschränkungen der Steuerfunktion erreicht werden muss.

Im zweiten Fall wird durch das inverse Robotermodell bereits garantiert, dass alle optimierten Trajektorien innerhalb der Toleranz sind – da diese ja direkt als Eingang für

[5] Der Roboter darf sich nicht auf der Trajektorie rückwärts bewegen

das Modell verwendet werden – so dass der Optimierer lediglich noch auf die Realisierbarkeit der Trajektorien zu achten hat. Dies kann durch geeignete Nebenbedingungen bzw. Kriterien erreicht werden.

Wegen der Vorteile der zweiten Variante wird diese zur Optimierung von Bahnen verwendet.

Die gewünschte Trajektorie $\mathfrak{T}$ wird im Achswinkelraum parametrisiert. Die Trajektorie $\mathfrak{T}$ wird hierzu zunächst über eine inverse Kinematik in Achswinkel $\boldsymbol{\varphi}_{a,r}(t)$ des (ideal starren) Roboters umgerechnet. Die (idealen) Achswinkel als Funktion der Zeit werden über das Bogenmaß $s(t)$ bestimmt. Über den Bahnparameter $s(t)$ können die idealen Gelenkwinkel durch $\boldsymbol{\varphi}_{a,r}(s(t))$ ausgedrückt werden. Dies wird durch Interpolationspolynome realisiert.

Als Steuerungsfunktion wird die Geschwindigkeit auf der Bahn $u(t) = \dot{s}(t)$ verwendet. Um sicherzustellen, dass der Endpunkt erreicht wird muss die Nebenbedingung $s(t_f) = s_f$ erfüllt sein, wobei s_f für das Bogenmaß am Endpunkt der Bahn steht. Als Zielfunktional wählen wir die Endzeit t_f und zusätzlich ein Maß für die Energie der Bewegung $\sum_{i=1}^{n_A} \left(\int_{t_0}^{t_f} |\tau_{m,i}\dot{\varphi}_{m,i}| \, dt \right)$ das dazu führt, dass die Momentenverläufe glätter werden. Zusätzlich werden die Zustandsgrößen $\varphi_{m,i}, \dot{\varphi}_{m,i}, \tau_{m,i}, \tau_{a,i}$ sowie $\dot{\tau}_{a,i}$ für alle $i = 1 \ldots n_A$ für die Bewegung notwendigen Achsen beschränkt, um eine realisierbare Trajektorie zu erzielen. Die mit „a“ gekennzeichneten Größen stehen hierbei für Werte hinter dem Getriebe (abtriebsseitig), die mit „m“ bezeichneten für motorseitige Größen. Die Beschränkung von $\dot{\tau}_{a,i}$ ist notwendig, um die Getriebe nicht durch zu große Momentensprünge zu beschädigen. Die Zustandsgrößen werden durch Integration des inversen Robotermodells berechnet, welches als DAE-System dargestellt werden kann. Als Approximation für $(\boldsymbol{\Sigma}^m)^{-1}$ wird ein approximatives strukturelastisches inverses Modell (ASIM) der Roboters $\boldsymbol{\Sigma}_{ff}^{(ASIM)}$ nach Abschnitt 3.8.1 verwendet.

Zusammenfassend ergibt sich die Optimierungsaufgabe zu Gl. (5.5).

$$min \left\{ \begin{array}{c} t_f \\ \sum_{i=1}^{n_A} \left(\int_{t_0}^{t_f} |\tau_{m,i}\dot{\varphi}_{m,i}| \, dt \right) \end{array} \right\} \tag{5.5a}$$

wobei gilt:

$$\boldsymbol{F}(t, \boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{z}, \boldsymbol{u}) = 0, \boldsymbol{x}(t_0) = \boldsymbol{x_0}, t \in [t_0, t_f] \tag{5.5b}$$

unter den Nebenbedingungen für $i = 1 \ldots n_A$

$$|\dot{\varphi}_{m,i}| \leq \dot{\varphi}_{m,i}^{max} \tag{5.5c}$$

$$|\tau_{m,i}| \leq \tau_{m,i}^{max} \tag{5.5d}$$

$$|\tau_{a,i}| \leq \tau_{a,i}^{max} \tag{5.5e}$$

$$|\dot{\tau}_{a,i}| \leq \dot{\tau}_{a,i}^{max} \tag{5.5f}$$

$$s(t_f) = s_f \tag{5.5g}$$

Um eine endliche Anzahl von Parametern für die Optimierung zu erhalten wird die Steuerfunktion $u(t) = \dot{s}(t)$ durch einen Polygonzug mit $m+1$ Stützstellen zu den Zeitpunkten $\{t_0, t_1, \ldots, t_m = t_f\}$ approximiert. Die Werte $u(t_j)$ mit $j = 0, 1, \ldots, m$ und die Randbedingungen $u(t_0) = u(t_m) = 0$ sind Stützstellen für eine stückweise lineare Interpolation unter der Nebenbedingung $u(t_j) \geq 0, j = 1, \ldots, m-1$. Die Nebenbedingung ist notwendig, damit die Bahn stets in Vorwärtsrichtung abgefahren wird. Eine

stückweise lineare Interpolation wurde gewählt, da (höherdimensionale) Splines zwar zu glätteren Verläufen führen würden, allerdings kann für diese nicht garantiert werden, dass die Bogengeschwindigkeit stets positiv bleibt (Überschwingen). Zudem werden die Trajektorien durch die, vor das inverse Modell geschalteten, Filter $\mathbf{\Sigma}^*$ geglättet.

Ausgangspunkt für die Optimierung sind Bahnen, welche über eine optimale Bahnplanung für einen starren Roboter generiert werden (nach einem Verfahren das auf [111, 112, 113] beruht). Diese müssen zunächst in eine Steuerfunktion $\boldsymbol{\varphi}_{a,r}(t) = \boldsymbol{\varphi}_{a,r}(s(t))$ in Abhängigkeit eines Bahnparameters $s(t)$ umgerechnet werden. $\dot{s}(t)$ wird dann durch einen stückweise linearen Polygonzug approximiert. Die Endpunktbedingung kann für die Optimierung zudem noch in eine günstigere Form gebracht werden (5.6). Sie wird hierzu in ein zu minimierendes Zielfunktional umgewandelt.

$$\left(\frac{s(t_f)}{s_f} - 1\right)^2 \rightarrow min \tag{5.6}$$

Dies dient dazu die Konvergenz der Optimierung zu verbessern.

Die erstellte Optimierungsaufgabe wurde mittels einem SQP[6]-Verfahren [104] mit *MOPS* (siehe hierzu [114, 115] und Anhang A.7) gelöst, wobei es nötig sein kann, dass die Optimierung mehrfach mit leicht geänderten Startwerten neu gestartet werden muss, um ein Feststecken in lokalen Optima zu vermeiden.

Je nach Gewichtung des Zielfunktionals, kann hierdurch entweder eine zeitoptimale oder energieoptimale bzw. pareto-optimale Lösung erreicht werden. Um das Verfahren zu testen wurden verschiedene Trajektorien optimiert, welche auch am Roboter KR16, im realen Experiment, gefahren wurden. Einige Ergebnisse der Optimierung zeigen die folgenden Abbildungen (5.5 und 5.6).

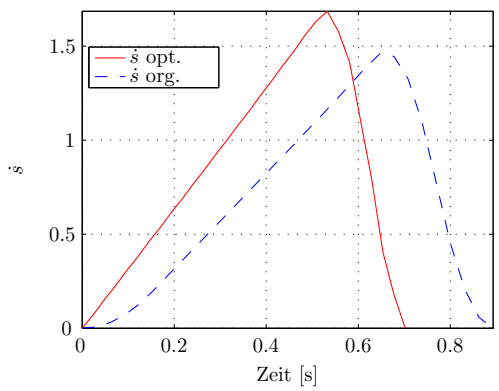
Wie anhand von Abbildung 5.5 zu erkennen ist, kann die Endzeit t_f deutlich reduziert werden ohne die Stellgrößenbeschränkung zu verletzen. Für die Bewegung nach Abbildung 5.5 beträgt die Zeitersparnis 21%. Alternativ kann die Endzeit t_f auch konstant gehalten werden, um die benötigte Energie zum Abfahren der Bahn zu minimieren (Abb. 5.6).

Für die Bewegung (Abb. 5.6) ist die Bahnenergie $\sum_{i=1}^{n_A} \left(\int_{t_0}^{t_f} |\tau_{m,i}\dot{\varphi}_{m,i}| \, dt \right)$ bei festgehaltener Endzeit t_f für die optimierte Bahn um 12% kleiner. Die Optimierungen wurden auch für verschiedene pareto-optimale Varianten und unterschiedliche Bahnen berechnet, welche aus Platzgründen allerdings nicht dargestellt sind. Alle optimierten Trajektorien ließen sich auch am realen Roboter abfahren.

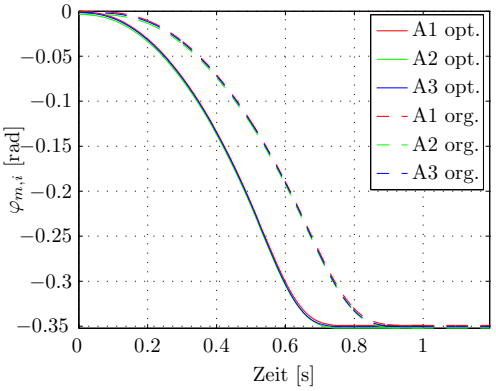
Die Optimierung von Trajektorien ist relativ zeitaufwändig (zum Teil mehrere Stunden). Sie kann allerdings dann sinnvoll eingesetzt werden, wenn eine identische Bahn in einem Prozess sehr oft wiederholt werden muss, wodurch sich (Takt-)Zeit und/oder Energie einsparen lässt. In diesem Fall lohnt sich der einmalige Aufwand der Optimierung.

Das hier vorgestellte Verfahren auf Basis von inversen Modellen, besitzt den großen Vorteil, dass Pfadbeschränkungen bereits direkt durch das inverse Modell erfüllt werden, wodurch keine extrem rechen- bzw. zeitaufwendigen Mehrfachschießverfahren notwen-

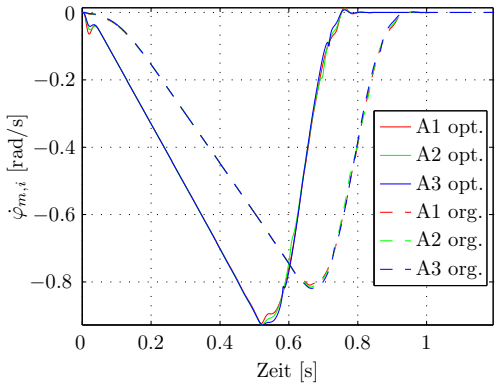
[6] Sequential quadratic programming



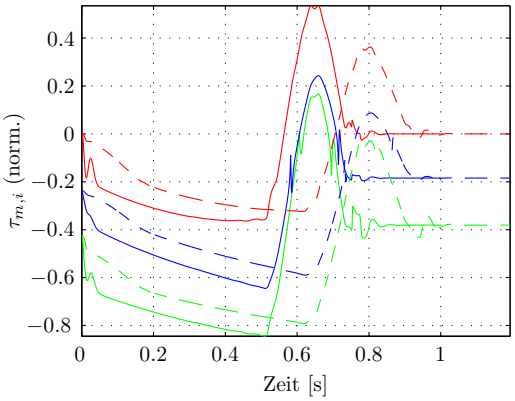
(a) Steuerfunktion $\dot{s}(t)$



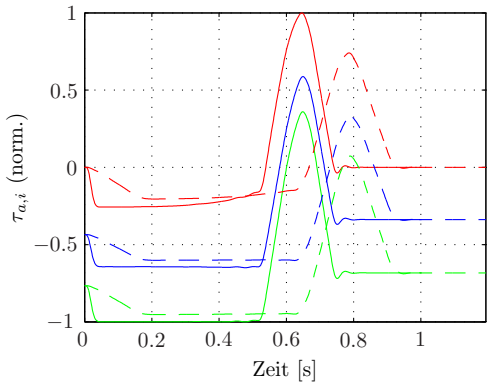
(b) Motorwinkel $\varphi_{m,i}(t)$



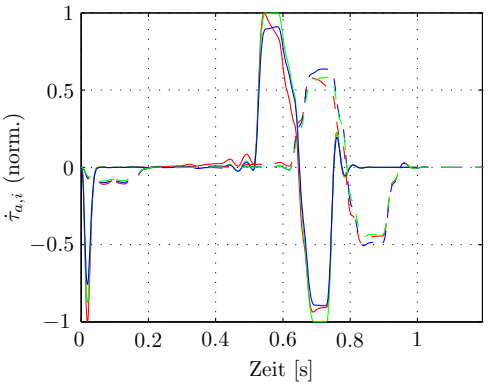
(c) Motorgeschwindigkeit $\dot{\varphi}_{m,i}(t)$



(d) Motormoment $\frac{\tau_{m,i}(t)}{\tau_{m,i}^{max}}$

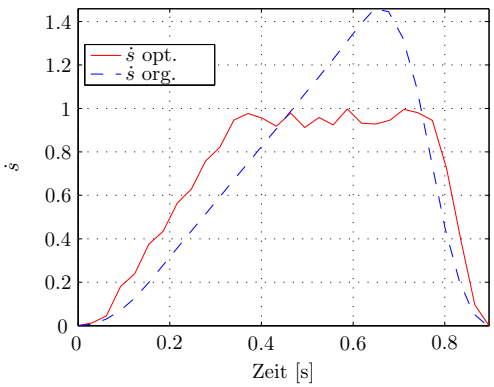


(e) Abtriebsmoment $\frac{\tau_{a,i}(t)}{\tau_{a,i}^{max}}$

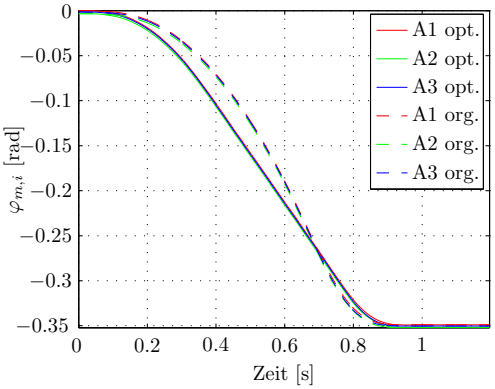


(f) Ableitung Abtriebsmoment $\frac{\dot{\tau}_{a,i}(t)}{\dot{\tau}_{a,i}^{max}}$

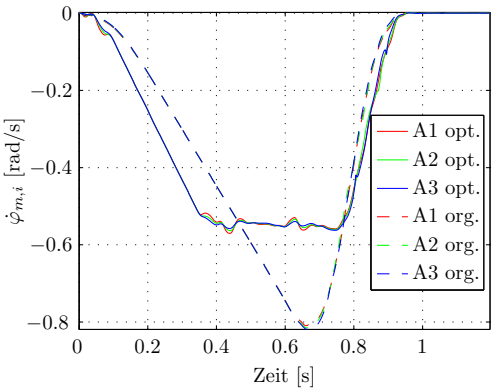
Abbildung 5.5: Optimierung der Endzeit t_f einer simultanen Bewegung der drei Grundachsen (Dreiachsbewegung) um jeweils -20° aus der Strecklage für einen KR16. Farbcodierung: Rot=Achse 1, Grün=Achse 2, Blau=Achse 3. Durchgezogen Linie=Optimierungsergebnis, gestrichelte Linie=Startlösung. Durch die Optimierung ist eine deutlich (für diese Trajektorie beträgt die Zeitersparnis 21%) Verkürzung der Fahrzeit möglich.



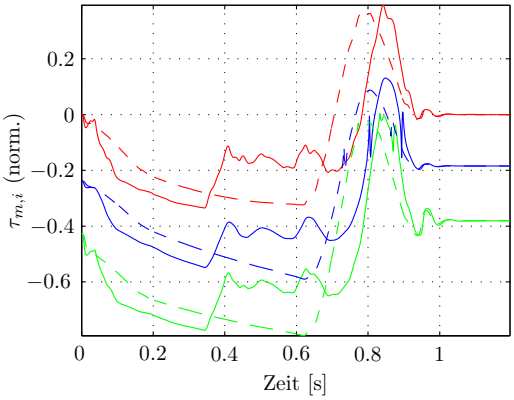
(a) Steuerfunktion $\dot{s}(t)$



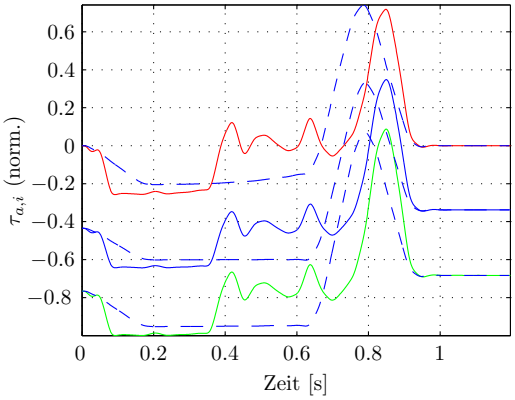
(b) Motorwinkel $\varphi_{m,i}(t)$



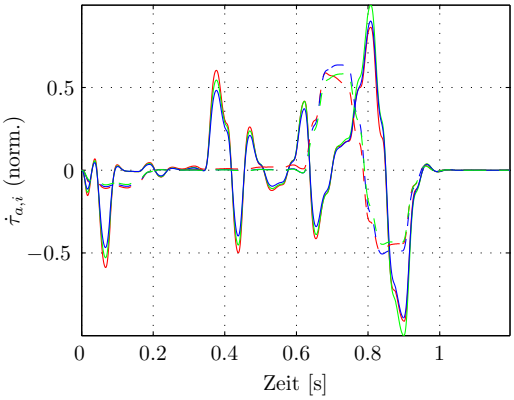
(c) Motorgeschwindigkeit $\dot{\varphi}_{m,i}(t)$



(d) Motormoment $\frac{\tau_{m,i}(t)}{\tau_{m,i}^{max}}$



(e) Abtriebsmoment $\frac{\tau_{a,i}(t)}{\tau_{a,i}^{max}}$



(f) Ableitung Abtriebsmoment $\frac{\dot{\tau}_{a,i}(t)}{\dot{\tau}_{a,i}^{max}}$

Abbildung 5.6: Optimierung der Bahnenergie $\sum_{i=1}^{n_A} \left(\int_{t_0}^{t_f} |\tau_{m,i}\dot{\varphi}_{m,i}| \, dt \right)$ bei festgehaltener Endzeit t_f einer simultanen Bewegung der drei Grundachsen (Dreiachsbewegung) um jeweils -20° aus der Strecklage für einen KR16. Farbcodierung: Rot=Achse 1, Grün=Achse 2, Blau=Achse 3. Durchgezogen Linie=Optimierungsergebnis, gestrichelte Linie=Startlösung. Durch die Optimierung wird die benötigte Bahnenergie deutlich (für diese Bahn um 12%) reduziert, ohne langsamer am Ziel zu sein.

dig sind. Bei einem Mehrfachschießverfahren wäre es nötig an den Diskretisierungs-Gitterpunkten Startwerte für die Zustände als zusätzliche Parameter einzuführen und die Zustandsdifferentialgleichungen damit zu lösen. Zusätzlich wäre es dann nötig, damit insgesamt eine Lösung des Ausgangsproblems entsteht, an den Gitterpunkten die Stetigkeit des Zustandsverlaufes als zusätzliche (implizit definierbare) Nebenbedingung zu fordern.

Dieser Mehraufwand entfällt beim Einsatz inverser Modelle, da das Problem hierdurch in ein Parameter-Optimierungsproblem überführt wird.

5.3.1 Optimierung von PTP-Bewegungen auf Basis von Ruckprofilen mit NL-MPC Korrekturschritt

Für PTP-Bewegung, bei welchen die Trajektorie $\mathfrak{T}$ im Raum nicht fest vorgegeben ist, sondern nur Start- und Zielpose, kann dieser zusätzlicher Freiheitsgrad für die Optimierung genutzt werden um die erzielbare minimale Endzeit t_f zusätzlich zu senken.

Als Parametrierungsansatz für die Bahn eignet sich ein Ruckprofil $\dddot{\varphi}_{opt}\left(t, t_f, \boldsymbol{a}, n\right)$ nach Gl. (5.7) für alle bewegten Achsen welches sich in Abhängigkeit der wählbaren Ordnung n mit dem Parametervektor $\boldsymbol{a} \in \mathbb{R}^{n+1}$ sowie der gewünschten Endzeit t_f ergibt.

$$\dddot{\varphi}_{opt}\left(t, t_f, \boldsymbol{a}, n\right) = \begin{cases} 0 & \frac{t}{t_f} < 0 \\ n\left(a_2 - a_1\right)\frac{t}{t_f} + a_1 & 0 < \frac{t}{t_f} \leq \frac{1}{n} \\ \left(a_3 - a_2\right)\left(\frac{t}{t_f} - \frac{1}{n}\right) + a_2 & \frac{1}{n} < \frac{t}{t_f} \leq \frac{2}{n} \\ \vdots & \vdots \\ \left(a_{n+1} - a_n\right)\left(\frac{t}{t_f} - \frac{n-1}{n}\right) + a_n & \frac{n-1}{n} < \frac{t}{t_f} \leq 1 \\ 0 & \frac{t}{t_f} > 1 \end{cases} \tag{5.7}$$

Durch die stückweise lineare Parametrierung kann dieses Ruckprofil einfach analytische integriert werden (Gl. (5.8)), wodurch Randbedingungen analytisch berechnet werden können.

$$\ddot{\varphi}_{opt,i}\left(t, t_f, \boldsymbol{a}_i, n\right) = \int \dddot{\varphi}_{opt,i}\left(t, t_f, \boldsymbol{a}_i, n\right) dt \tag{5.8a}$$

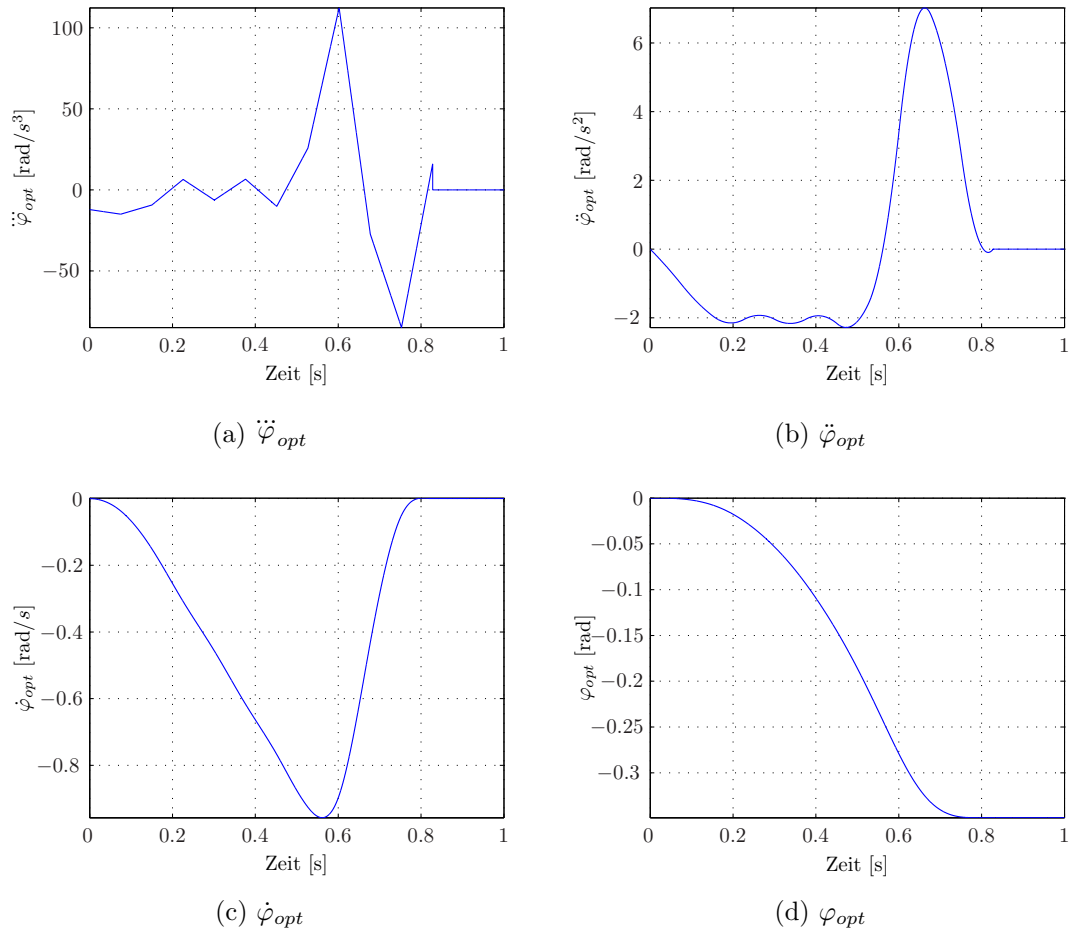
$$\dot{\varphi}_{opt,i}\left(t, t_f, \boldsymbol{a}_i, n\right) = \int \ddot{\varphi}_{opt,i}\left(t, t_f, \boldsymbol{a}_i, n\right) dt \tag{5.8b}$$

$$\varphi_{opt,i}\left(t, t_f, \boldsymbol{a}_i, n\right) = \int \dot{\varphi}_{opt,i}\left(t, t_f, \boldsymbol{a}_i, n\right) dt + \varphi_{opt,start,i} \tag{5.8c}$$

Die zur Optimierung verwendeten Randbedingungen ergeben sich nach Gl. (5.9) mit den zusätzlich vorzugebenden Start- und Endwinkelvektoren $(\boldsymbol{\varphi}_{opt,start}, \boldsymbol{\varphi}_{opt,stop})$.

$$\text{RB: } \boldsymbol{\varphi}_{opt}\left(t = t_f\right) = \boldsymbol{\varphi}_{opt,stop},\ \dot{\boldsymbol{\varphi}}_{opt}\left(t = t_f\right) = 0,\ \ddot{\boldsymbol{\varphi}}_{opt}\left(t = t_f\right) = 0 \tag{5.9}$$

Zur Erfüllung der Randbedingungen (RB) einstehen neue Gleichungen für die Parametervektoren $\boldsymbol{a}_i$. Die Anzahl der Randbedingungen reduziert daher die Anzahl der vom Optimierer frei wählbaren Parameter der Parametervektoren $\boldsymbol{a}_i$. Pro Randbedingung wird eine Komponente von $\boldsymbol{a}_i$ explizit berechnet. Für jede bewegte Achse wird



(a) $\dddot{\varphi}_{opt}$ (b) $\ddot{\varphi}_{opt}$

(c) $\dot{\varphi}_{opt}$ (d) φ_{opt}

Abbildung 5.7: Beispiel für ein Ruckprofil nach Gl. (5.7) mit analytischer Integration nach Gl. (5.8) und der Endzeit $t_f = 0,828$ für eine Verfahrlänge von -20° und Nebenbedingungen nach Gl. (5.9).

ein eigener Parametervektor $\boldsymbol{a}_i$ verwendet, wodurch für jede Achse unterschiedliche Ruck-Profile generiert werden können.

Durch die Integration der Profile entstehen bereits bei der linearen Parametrierung von $\dddot{\varphi}_{opt}$ relativ glatte Verläufe von φ_{opt}, was in der Optimierung von Vorteil ist. Ein Beispiel für ein Ruckprofil zeigt Abb. 5.7.

Diese Profile werden analog zu Abschnitt 5.3 als Eingangsgrößen für ein inverses Robotermodell $(\boldsymbol{\Sigma}^m)^{-1}$ verwendet indem $\boldsymbol{\varphi}_{a,r} \equiv \boldsymbol{\varphi}_{opt}$ gesetzt wird. Als Approximation für $(\boldsymbol{\Sigma}^m)^{-1}$ wird wieder ein approximatives strukturelastisches inverses Modell (ASIM) der Roboters $\boldsymbol{\Sigma}_{ff}^{(ASIM)}$ nach Abschnitt 3.8.1 verwendet. Das Optimierungsproblem das nun gelöst werden muss ergibt sich dann analog zu Gl. (5.5), mit dem Unterschied, dass die Bedingung an die Steuerfunktion $s(t_f) = s_f$ wegfällt und durch die Zwangsbedingungen an die Parametervektoren $\boldsymbol{a}_i$ ersetzt wird, welche sich durch die Randbedingungen aus Gl. (5.9) ergeben. Der Optimierer versucht nun durch eine Variation der verbleibenden Freiheitsgrade in $\boldsymbol{a}_i$ sowie der Endzeit t_f die Nebenbedingungen $|\dot{\varphi}_{m,i}| \leq \dot{\varphi}_{m,i}^{max}, |\tau_{m,i}| \leq \tau_{m,i}^{max}, |\tau_{a,i}| \leq \tau_{a,i}^{max}, |\dot{\tau}_{a,i}| \leq \dot{\tau}_{a,i}^{max}$ zu erfüllen und eine minimale, optimale Endzeit t_f^* zu erreichen. Als Ausgangspunkt für diese Optimierung eignen sich wieder optimale Trajektorien eines starren Roboters anlog zu Abschnitt 5.3. Die

optimale Trajektorie eines starren Roboters kann in ein Ruckprofil umgerechnet werden indem, zusätzlich zu den gewünschten Randbedingungen, neue Gleichungen für die Parametervektoren $\boldsymbol{a}_i$ der Ruckprofile, der einzelnen Achsen, aufgestellt werden. Hierzu wird der optimale Winkelverlauf für einen starren Roboter $\boldsymbol{\varphi}^*_{opt,r}(t)$ zeitdiskret aufgeteilt und analytisch mit dem Startwert des Ruckprofil-Vektors $\boldsymbol{\varphi}^{(0)}_{opt}$ gleichgesetzt um Gleichungen für die $i \cdot n_{f,a} \in \mathbb{N}$ verbleibenden Freiheitsgrade für alle $\boldsymbol{a}_i$ zu erhalten (Gl. (5.10)).

$$\varphi^{(0)}_{opt,i}\left(t = \frac{1t_f}{n_{f,a}+1}, t_f, \boldsymbol{a}_i, n\right) = \varphi^*_{opt,r,i}\left(t = \frac{1t_f}{n_{f,a}}\right) \tag{5.10a}$$

$$\varphi^{(0)}_{opt,i}\left(t = \frac{2t_f}{n_{f,a}+1}, t_f, \boldsymbol{a}_i, n\right) = \varphi^*_{opt,r,i}\left(t = \frac{2t_f}{n_{f,a}}\right) \tag{5.10b}$$

$$\vdots \tag{5.10c}$$

$$\varphi^{(0)}_{opt,i}\left(t = \frac{n_{f,a}t_f}{n_{f,a}+1}, t_f, \boldsymbol{a}_i, n\right) = \varphi^*_{opt,r,i}\left(t = \frac{n_{f,a}t_f}{n_{f,a}+1}\right) \tag{5.10d}$$

In einem zusätzlichen Schritt können die Fehler, welche durch die Verwendung einer approximativen Inversen der Strecke innerhalb der Optimierung entstehen können, durch eine anschließende Optimierung auf Basis eines nicht approximierten Vorwärtsmodell $\boldsymbol{\Sigma}^m$ korrigiert werden. Hierzu eignet sich ein Ansatz inspiriert von der nichtlinearen modellprädiktiven Regelung (NL-MPC[7], siehe z. B. [109] und [116]). Durch die Optimierung im ersten Schritt auf Basis von $\boldsymbol{\Sigma}^{(ASIM)}_{ff}$ wird ein optimaler Verlauf (auf Basis der in $\boldsymbol{\Sigma}^{(ASIM)}_{ff}$ getroffenen Approximationen) der Motormoment Verläufe $\boldsymbol{\tau}^{*,(1)}_m(t)$ berechnet welche als Startlösung für den zweiten Optimierungsschritt dient. Ebenso wird durch den ersten Schritt ein optimaler Verlauf der TCP-Position[8] $\boldsymbol{r}^{*,(1)}_L(t)$ berechnet, welcher als Referenztrajektorie für den NL-MPC Schritt dient (sie ist durch das inverse Modell $\boldsymbol{\Sigma}^{(ASIM)}_{ff}$ schwingungsfrei am Endpunkt). Im nicht approximierten Vorwärtsmodell $\boldsymbol{\Sigma}^m$ wird ein Reibmodell mit Zustandsumschaltung im Haftbereich nach [23] verwendet welche nicht – wie die Approximation nach Gl. (2.3), welche in $\boldsymbol{\Sigma}^{(ASIM)}_{ff}$ verwendet wird – invertierbar ist. Im Vorwärtsmodell des Roboters $\boldsymbol{\Sigma}^m$ ist zudem natürlich keine Approximation der Verformungen der elastischen Strukturelemente nach Abschnitt 3.5.2 nötig.

Da durch den ersten Optimierungs-Schritt $\boldsymbol{\tau}^{*,(1)}_m(t)$ bereits eine sehr gute Startlösung für den zweiten Optimierungsschritt darstellt, genügt eine lineare Parametrierung für den NL-MPC-Korrekturschritt. Der lineare Korrekturvektor der Motormomente (als Funktion der Zeit) wird mit $\boldsymbol{\tau}^{(NLMPC)}_m(t)$ bezeichnet. Die optimalen Verläufe der Motormomente nach dem 2. Korrekturschritt ergeben sich dann als Summe nach Gl. (5.11).

$$\boldsymbol{\tau}^{*,(2)}_m(t) = \boldsymbol{\tau}^{*,(1)}_m(t) + \boldsymbol{\tau}^{(NLMPC)}_m(t) \tag{5.11}$$

Stellhorizont und Vorhersagezeitraum werden identisch zu $N_u \Delta t$ gewählt mit einer wählbaren Zeit-Schrittgröße Δt und $N_u \in \mathbb{N}$ welche dem Problem angepasst[9] zu wählen ist (siehe Abb. 5.8). Das Optimierungsproblem für jeden Zeitschritt k ergibt

[7] Englisch: Nonlinear Predictive Control.

[8] Dies kann auch um die TCP-Orientierung erweitert werden.

[9] Eine sinnvolle Wahl für Δt ist die Taktrate eines typischen Reglers für das System.

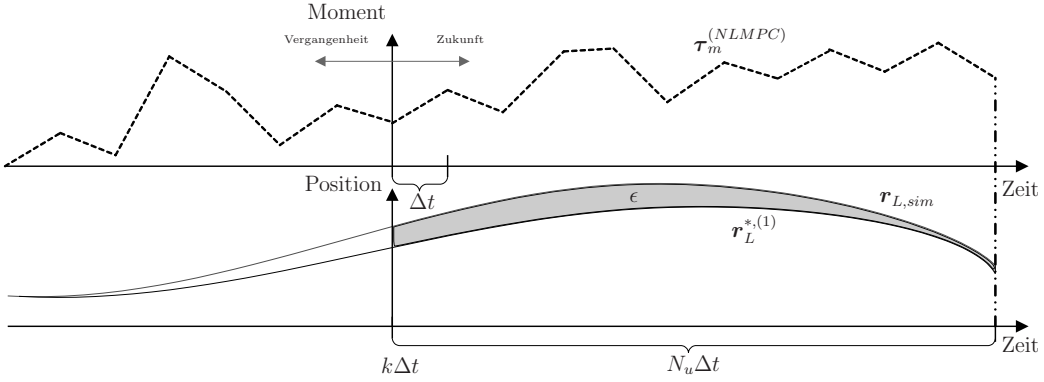


Abbildung 5.8: Konzept des NL-MPC Korrekturschritts: Für jede Achse des Roboters werden N_u lineare Korrekturterme $\boldsymbol{\tau}_m^{(NLMPC)}$ über die Optimierung bestimmt. Ziel ist es die Abweichung ϵ zwischen Referenztrajektorie $\boldsymbol{r}_L^{*,(1)}(t)$ und auf Basis des Vorwärtsmodells $\boldsymbol{\Sigma}^m$ simulierter Trajektorie $\boldsymbol{r}_{L,sim}(t)$ unter Einhaltung der Nebenbedingungen zu minimieren.

sich dann zu Gl. (5.12). Der zweite zu minimierende Term in Gl. (5.12) stellt einen Bestrafungsterm für Stellgrößenänderungen dar. Hierzu muss eine Konstante $\lambda_u \in \mathbb{R}^+$ gewählt werden. Er dient dazu das Ergebnis zu glätten und keine zu extremen Stellgrößenänderungen zu erzeugen. Er wird für die Optimierung als zeitdiskrete Summe berechnet.

$$min \left\{ \begin{array}{c} \epsilon = \int_{k\Delta t}^{(k+N_u)\Delta t} \left\| \boldsymbol{r}_L^{*,(1)}(t) - \boldsymbol{r}_{L,sim}(t) \right\|_2 dt \\ \lambda_u \sum_{j=1}^{N_u} \left\| \boldsymbol{\tau}_m^{(NLMPC)}\left(k + j\Delta t\right) - \boldsymbol{\tau}_m^{(NLMPC)}\left(k + (j-1)\Delta t\right) \right\|_2 \end{array} \right\} \quad (5.12a)$$

wobei gilt:

$$\boldsymbol{F}(t, \boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{z}, \boldsymbol{u}) = 0, \boldsymbol{x}(k\Delta t) = \boldsymbol{x_0}, t \in [k\Delta t, (k + N_u)\,\Delta t] \quad (5.12b)$$

unter den Nebenbedingungen für $i = 1 \ldots n_A$

$$|\dot{\varphi}_{m,i}| \leq \dot{\varphi}_{m,i}^{max} \quad (5.12c)$$

$$|\tau_{m,i}| \leq \tau_{m,i}^{max} \quad (5.12d)$$

$$|\tau_{a,i}| \leq \tau_{a,i}^{max} \quad (5.12e)$$

$$|\dot{\tau}_{a,i}| \leq \dot{\tau}_{a,i}^{max} \quad (5.12f)$$

Analog zu üblichen MPC Verfahren [116] wird nach jedem Optimierungsschritt k um eins erhöht und die Optimierung neu aufgesetzt. Hierzu wird der Zustand $\boldsymbol{x}$ von $\boldsymbol{\Sigma}^m$ auf Basis des vorhergehende Schritts und der optimierten Stellgrößen neu berechnet. Hierdurch wird der aktuelle Stellhorizont und Vorhersagezeitraum in jedem Schritt um Δt verschoben. Durch die Elastizität im System ist es nötig die Optimierung auch für eine Gewisse Zeit nach t_f fortzusetzen, da Schwingungen des TCPs nach erreichen der Endposition zum Zeitpunkt t_f des Modells $\boldsymbol{\Sigma}^m$ ausgeglichen werden müssen (durch Approximationsfehler). Jede Funktionsauswertung in der Optimierung entspricht daher einer Simulation von $\boldsymbol{\Sigma}^m$ vom Zeitpunkt $t = k\Delta t$ bis $t = (k + N_u)\,\Delta t$. Abbildung 5.9 zeigt eine Übersicht für das zweistufige Optimierungskonzept um zu zeitoptimalen PTP-Bewegungen, unter der Einhaltung von Nebenbedingungen, zu gelangen. Abbildung 5.10 zeigt das Ergebnis der zweistufigen Optimierung für eine -20° Dreiachsbewegung aus der Strecklage eines KR16 analog zu Abschnitt 5.3, aber ohne eine fest vorgegebene Trajektorie $\mathfrak{T}$. Durch den zusätzlichen Freiheitsgrad in der Optimierung

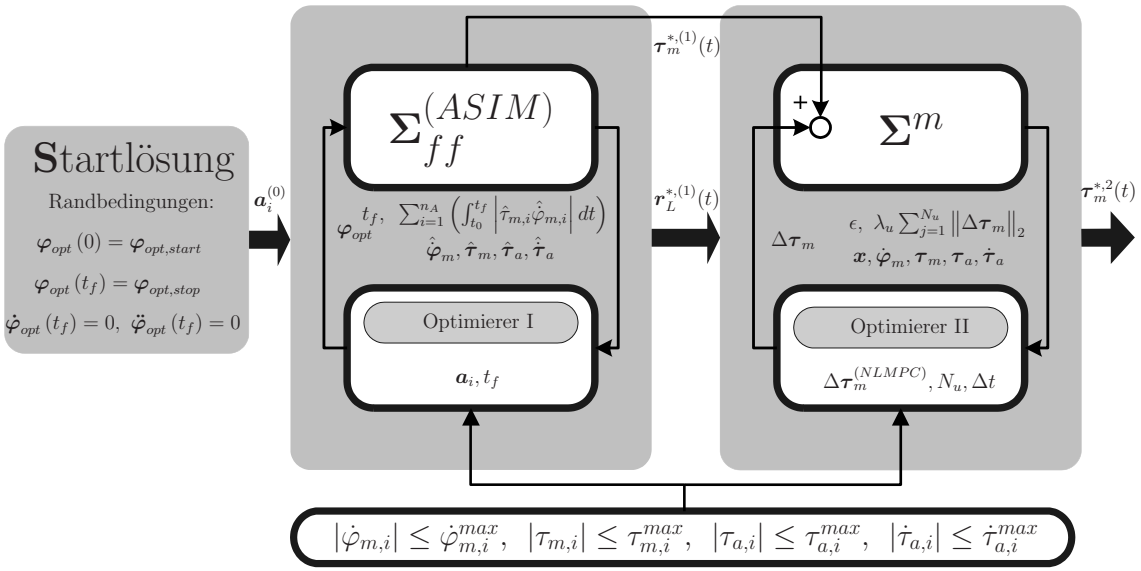
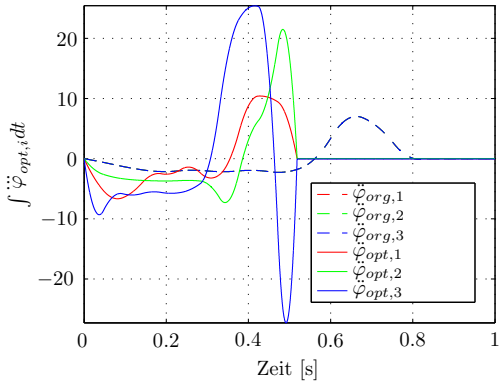
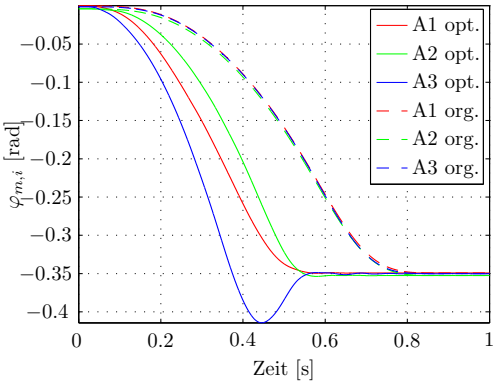


Abbildung 5.9: Zweistufiges Optimierungskonzept: Angefangen von einer Startlösung und gewählten Randbedingungen werden analytisch Startwerte für alle $\boldsymbol{a}_i$ berechnet welche mit $\boldsymbol{a}_i^0$ bezeichnet sind. Im ersten Optimierungsschritt variiert der Optimierer I $\boldsymbol{a}_i$ und t_f und erzeugt hierdurch einen neuen Ruckprofil-Vektor $\boldsymbol{\varphi}_{opt}$ mit dem Ziel Endzeit t_f und ein Energiekriterium unter Einhaltung der Nebenbedingungen zu minimieren. Das Model in der ersten Optimierung ist ein approx. inverses Model $\boldsymbol{\Sigma}_{ff}^{(ASIM)}$. Das Ergebnis dieser Optimierung wird gespeichert und dient als Vorsteuergrößen $\boldsymbol{\tau}_m^{*,(1)}$ und Solltrajektorie $\boldsymbol{r}_L^{*,(1)}$ für den zweiten Optimierungsschritt. In diesem dient ein NL-MPC Ansatz dazu, basierend auf einem nicht approx. Vorwärtsmodell $\boldsymbol{\Sigma}^m$, den Momentenverlauf weiter zu verbessern und die Nebenbedingung exakt einzuhalten.

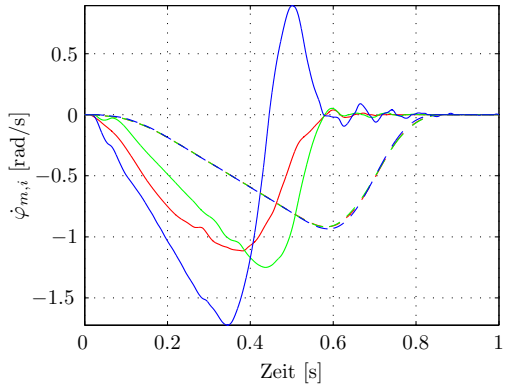
kann die Endzeit t_f um etwa 37% gegenüber der Ausgangslösung reduziert werden, wohingegen mit dem verfahren nach Abschnitt 5.3 ca. 21% erreicht wurden. Dies wird jedoch mit einer deutlich größeren Rechenzeit erkauft und die Start-Trajektorie wird verändert, was in der Nähe von Hindernissen zu Problemen führen kann. Die Optimierung führt jedoch zu einer sehr genauen Einhaltung der Beschränkungen (Nebenbedingungen), welche nicht auf Approximationen basiert.



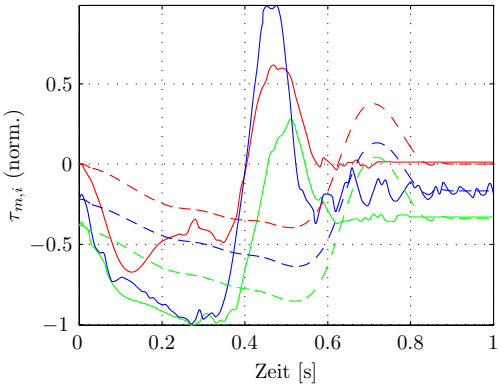
(a) Integral über das Ruckprofil $\int \dddot{\varphi}_{opt,i} dt$



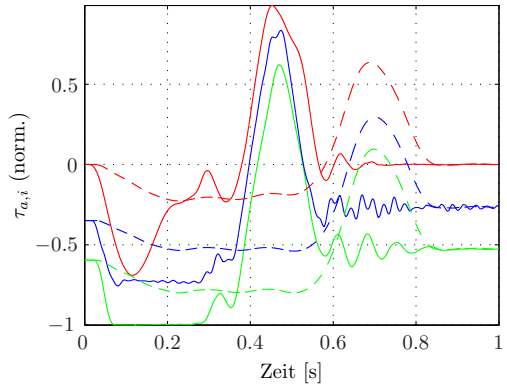
(b) Motorwinkel $\varphi_{m,i}(t)$



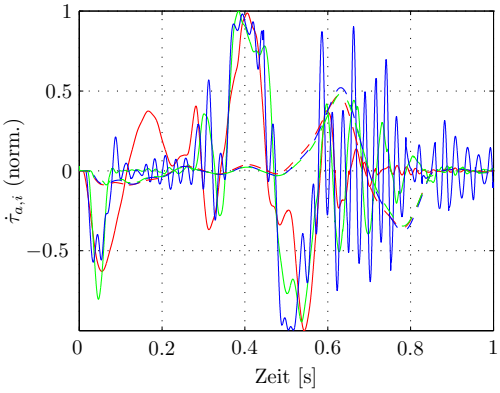
(c) Motorgeschwindigkeit $\dot{\varphi}_{m,i}(t)$



(d) Motormoment $\frac{\tau_{m,i}(t)}{\tau_{m,i}^{max}}$



(e) Abtriebsmoment $\frac{\tau_{a,i}(t)}{\tau_{a,i}^{max}}$



(f) Ableitung Abtriebsmoment $\frac{\dot{\tau}_{a,i}(t)}{\dot{\tau}_{a,i}^{max}}$

Abbildung 5.10: Optimierung der Endzeit t_f auf Basis von Ruckprofilen mit NL-MPC Korrekturschritt für eine simultane Bewegung der drei Grundachsen um jeweils -20° aus der Strecklage für einen KR16. Farbcodierung: Rot=Achse 1, Grün=Achse 2, Blau=Achse 3. Durchgezogen Linie=Optimierungsergebnis, gestrichelte Linie=Startlösung. Da die Trajektorie gegenüber der Optimierung in Abschnitt 5.3 nicht fest vorgegeben ist kann die Endzeit t_f, an welcher der Zielpunkt erreicht wird noch mal deutlich gegenüber der Ausgangslösung reduziert werden (um etwa 37%, wohingegen bei fester Trajektorie 21% erreicht wurden).

Kapitel 6

Experimentelle Ergebnisse

In diesem Kapitel werden experimentelle Ergebnisse vorgestellt, welche mit den erstellten inversen Modellen erzielt wurden. Hierzu wird zunächst kurz beschrieben, wie die Modelle für eine Auswertung in Echtzeit aufbereitet werden und welche Soft- und Hardware hierzu verwendet wurde. Anschließend werden die Ergebnisse von Messungen gezeigt, welche an den Robotern KR16 und KR210 durchgeführt wurden.

6.1 Implementierung der Modelle für die Auswertung in Echtzeit

Um die inversen Modelle im Rahmen einer Regelung des Roboters anwenden zu können müssen diese in Echtzeit berechnet werden können.

Für den Einsatz eines inversen Modells innerhalb einer Vorsteuerung ist es zwar grundsätzlich möglich die Berechnungen für eine gegebene Trajektorie im Voraus (Offline) zu berechnen und sie dann als Solldaten für einen Regler in der Form von Tabellen zu speichern, dies ist jedoch für viele Aufgabenstellungen in der Robotik nicht sinnvoll oder ausreichend.

Trajektorien werden in vielen Anwendungsfällen erst zur Laufzeit erstellt, da in den Roboterprogrammen oftmals zusätzliche Sensorinformationen oder Steuerungsanweisungen von übergeordneten Steuerungssystemen (z. B. Speicherprogrammierbare Steuerungen) berücksichtigt werden müssen, welche den Verlauf der Trajektorie ändern können. Inverse Modelle des Roboters müssen daher in Echtzeit berechenbar sein.

Modelle, die in *Dymola* erstellt wurden, können von der *Modelica* Sprache in C-Code übersetzt werden, welcher mittels eines geeigneten Compilers (z. B. *Microsoft Visual C++ Compiler*) in ein ausführbares Programm kompiliert werden kann. Bei diesem Kompilierungsschritt ist es möglich einen Integrationsalgorithmus in den Code einzufügen, so dass das erstellte Modell über eine *MATLAB-SIMULINK* Schnittstelle direkt als diskreter Block ausgewertet werden kann, ohne einen externen Integrationsalgorithmus zu verwenden. Dieses Verfahren wird mit Inline-Integration bezeichnet (siehe [117]).

Um die Modelle in Echtzeit auswerten zu können, ist es zunächst nötig festzustellen

was die, für das inverse Modell mögliche, maximale Schrittweite Δt_i für einen Integrationsschritt darstellt, ohne dass die Integration numerisch instabil wird.

Wie groß Δt_i gewählt werden kann hängt dabei stark von der Struktur und den Parametern des Modells, sowie von dem verwendeten Integrationsalgorithmus ab.

Innerhalb von *SIMULINK* können über den sog. Multitasking-Modus verschiedene Auswertungstakte für unterschiedliche Modellkomponenten eingestellt werden. Es ist daher möglich, im Rahmen einer Regelung mit zwei Freiheitsgraden (nach Abschnitt 5.1), das inverse Modell als Vorsteuerung in einem langsameren Takt auswerten zu lassen, als den Regler des Roboters. Dies ermöglicht es, die Rechenzeit pro Auswertungstakt zu erhöhen wodurch komplexere Modelle berechenbar sind.

Um die Modelle am Roboter testen zu können wurde *xPC-Target* in Verbindung mit *MATLAB-SIMULINK*, zusammen mit einer *Modelica/Dymola*-Schnittstelle eingesetzt. In der Testumgebung sind zusätzlich die Bahnplanung und die Roboterprogrammauswertung sowie die Benutzeroberfläche auf einen zweiten PC ausgelagert.

Eine weitere Möglichkeit um die Rechenzeit für Modelle zu verbessern ist die Modelle numerisch genau zu analysieren. Hierzu bietet es sich an die Systeme zu linearisieren und den Einfluss der einzelnen Zustände auf das Systemverhalten zu betrachten. Für die Integrationsverfahren sind hierbei besonders Eigenwerte mit sehr großem Betrag für das Systems kritisch, bzw. Pole mit sehr großem Realteil. Diese zwingen den Integrator zu sehr kleinen Berechnungsschritten, um die geforderte Toleranz für die Berechnung einzuhalten. Diese Zustände, bzw. die Komponenten im Modell welche dafür verantwortlich sind, können oft vernachlässigt werden, ohne das Ein-/Ausgangsverhalten des Systems stark zu beeinflussen. Für inverse Modelle des Roboters sind dies etwa extrem hohe Steifigkeiten (im Vergleich zu anderen Steifigkeiten im Modell), welche für eine Auswertung in Echtzeit vernachlässigt werden können.

Kritisch für eine Auswertung in Echtzeit sind auch iterative Algorithmen. Diese müssen so beschränkt werden, dass immer eine maximale Anzahl von Iterationen - zusätzlich zu etwaigen Abbruchbedingungen bezügliche einer festen Toleranz - vorgeben werden, welche mit der verfügbaren Rechenzeit bearbeitet werden können.

6.2 Experimentelle Untersuchungen am Roboter KR16

Ziel der Messungen ist es, die erstellten approximativen strukturelastischen inverse Modelle (im weiteren abgekürzt mit ASIM-KR16 für das inverse Model des KR16 welches analog zu dem in Abschnitt 3.8.3 aufgebaut ist) im Rahmen einer Vorsteuerung $\mathbf{\Sigma}_{ff}$ am realen Roboter zu testen und Vergleiche zu einer Vorsteuerung zu erzielen, welche keine Strukturelastizität, sondern nur (nichtlineare) Getriebeelastizitäten und Reibung in Kombination mit einer Starrkörpermechanik berücksichtigen (RLFJ[1]-Vorsteuerungen). Zusätzlich wurde ein Vergleich zu einer reinen Starrkörpervorsteuerung $\mathbf{\Sigma}_{ff}^{(r)}$ durch-

[1]RLFJ - Rigid Link Flexible Joint. Englische Abkürzung für eine Modellvorstellung mit Getriebeelastizitäten aber ohne Strukturelastizitäten in Kombination mit einem Starrkörpermodell der mechanischen Kinematik

geführt, um den Einfluss von Getriebe und Strukturelastizität zu verdeutlichen.

Als Regler wurde jeweils der Standard-Regler eingesetzt, dessen Parameter nicht verändert wurden und für die Sollwerte der Vorsteuerung eine identische[2] Sollbahn aus der Bahnplanung. Zusätzlich wurde das Fahrverhalten anhand von einer kartesischen Trajektorie untersucht.

6.2.1 Kriterium

Für die Messungen wurde ein 3D-Beschleunigungssensor (Messungen in allen drei Raumrichtungen) am Handwurzelpunkt des KR16 befestigt. Das Kriterium C ist die Summe über den absoluten Betrag der mittelwertfreien Beschleunigungsmesswerte für $T = 0.9$ Sekunden nachdem die abtriebsseitige Sollgeschwindigkeit $v_{a,soll}$ des Roboters zum Zeitpunkt t_0 Null wird (Gleichung (6.1)).

$$C = \sum_{k=1}^{3} \int_{t=t_0}^{t_0+T} |a_k(t) - \bar{a}_k| \, dt \tag{6.1}$$

Es wurde für die Bewertung des Positionierungsverhaltens nur eine sehr kurze Zeit T gewählt, um Messrauschen nicht unnötig mit zu integrieren. Der Großteil der Schwingung ist dann auch abgeklungen.

Das Beschleunigungssignal ist sehr empfindlich und selbst schwache Schwingungen können noch gut gemessen werden. Messungen können zudem einfach im kompletten Arbeitsraum und für beliebige Stellungen durchgeführt werden, so lange die Schwerkraft kompensiert wird, was hierbei durch das Abziehen des Mittelwerts $\bar{a}_k$ geschieht. Für sehr kleine T kann die Zeit für die Mittelwertbildung erhöht werden, um den statischen Mittelwert der Beschleunigung (der durch die Gravitation bestimmt wird) zu ermitteln.

6.2.2 Messungen

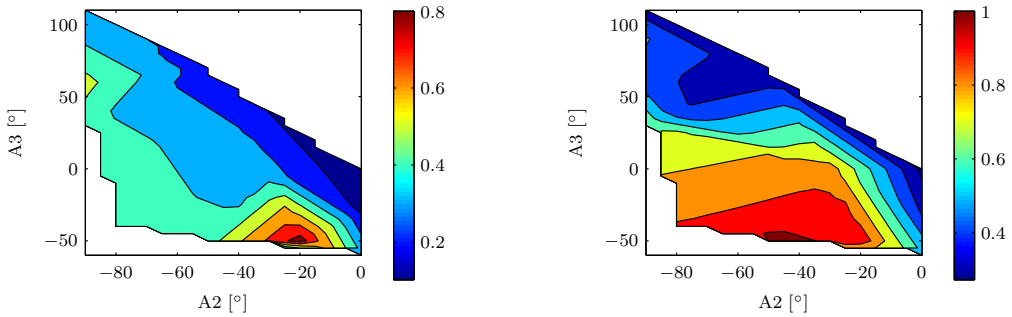
Im Folgenden werden unterschiedliche Ergebnisse (Abbildung 6.2, 6.1 und 6.3) zu den Messungen gezeigt. Die Messungen befinden sich im üblichen Arbeitsbereich des Roboters und wurden zwischen den Stützstellen interpoliert. Messpunkte sind in den 3D-Darstellungen durch Kreise gekennzeichnet. Aufgetragen ist jeweils das Kriterium über den Startwinkeln der Achsen 2 und 3. Die Achsen 4 bis 6 wurden nicht bewegt und stehen jeweils in ihrer 0° Position.

Für die Messungen wurden simultane Bewegungen der drei Grundachsen (PTP-Dreiachsbewegungen) gewählt, wobei die Solltrajektorie direkt aus der Bahnplanung stammt. Simultane Bewegungen der Grundachsen stellen eine starke Anregung der Mechanik dar und sind damit geeignet, um das Positionierverhalten zu überprüfen. Ist es, aufgrund der Startstellung, nicht möglich Bewegungen in positive Rotationsrichtung

[2] Die Vorfilterkonfigurationen von $\mathbf{\Sigma}^*$ wurden entsprechend angepasst, um die für das inverse Modell zusätzlich benötigten Filter zur Differentiation zu kompensieren.

auszuführen, wurden entsprechende Dreiachsbewegungen mit geändertem Vorzeichen durchgeführt, um eine Kollision zu vermeiden.

Die Messungen wurden dabei jeweils drei mal wiederholt und ein Mittelwert gebildet, wobei die Abweichungen des Kriteriums der einzelnen Messungen vom Mittelwert der Messungen stets unter 5 % lag.



(a) Relativer Vergleich (ASIM-KR16 zu RLFJ-Vorsteuerung) der Kriterien für eine simultane 5° Bewegung der Grundachsen.

(b) Relativer Vergleich (ASIM-KR16 zu RLFJ-Vorsteuerung) der Kriterien für eine simultane 20° Bewegung der Grundachsen.

Abbildung 6.1: KR16: Relatives Kriterium $C_R = \left(\frac{C_{ASIM}}{C_{RLFJ}}\right)$ für das approx. strukturelastisches inverses Modell (ASIM-KR16) bezüglich RLFJ-Vorsteuerung für simultane 5° und 20° Bewegungen der Grundachsen. Der verwendete Regler ist jeweils identisch (Regelungsstruktur mit zwei Freiheitsgraden).

Wie man den Messungen nach Abb. 6.1 entnehmen kann, ist der Einfluss der Strukturseifigkeit beim KR16 stark stellungsabhängig. Vor allem in der ausgestreckten Lage (A2 bis A6 jeweils bei 0°) des Roboters führt eine Berücksichtigung der Strukturelastizitäten zu einer deutlichen Verbesserung im Ausschwingverhalten beim Positionieren im Vergleich zu einer Vorsteuerung in welcher lediglich die Getriebeelastizität berücksichtigt wird. Es existieren jedoch auch Stellungen, in welchen die Verbesserung weniger deutlich ist. Zudem regen die kürzeren 5° Dreichachsbewegungen die Mechanik stärker an, als die 20° Bewegungen, weshalb hier die Unterschiede geringer Ausfallen. Jedoch sind auch für 20° Bewegungen für einige Stellungen deutliche Verbesserungen sichtbar. Sieht man sich das absolute Kriterium für die verschiedenen Vorsteuerungen nach Abb. 6.3 an erkennt man, dass das Kriterium für die ASIM-KR16 Vorsteuerung deutlich weniger variiert (daher der Kriteriumsverlauf in der 3D-Darstellung deutlich flacher verläuft) als für die Vorsteuerungen, in welcher nur die Getriebeelastizität (RLFJ-Vorsteuerung) bzw. gar keine Elastizität (Starrkörpervorsteuerung) berücksichtigt wird.

Zusätzlich zu den Messungen über den Arbeitsraum mit Hilfe der Beschleunigungs- und Drehratensensoren wurden auch Messungen mit einer 6D-Kamera[3] durchgeführt.

Abbildung 6.4 zeigt die mit der Kamera aufgezeichnete Trajektorie des TCPs des Roboters KR16 für eine simultane 5° Bewegung der Grundachsen aus der Strecklage. In

[3]6D-Kamera der K-Serie von Metris. Über drei CCD-Kameras kann die Position und Orientierung von Infrarot-Dioden, welche am Roboter angebracht sind, gemessen werden. Die Einzelpunkt-Genauigkeit wird vom Hersteller auf 60μm (2σ) angegeben.

dieser Stellung sind die Anregungen auf die Struktur und Getriebe des Roboters besonders deutlich zu erkennen. Die gleiche Dreiachsbewegung wurde mit drei verschiedenen Vorsteuerungen (bei identischem Regler) durchgeführt. Leichte Unterschiede im Bahnverlauf stammen von der Vorfilterkonfigurationen $\boldsymbol{\Sigma}^*$ welche für die Vorsteuerung $\boldsymbol{\Sigma}_{ff}^{(ASIM)}$ angepasst werden muss, um auf die etwa gleiche (bzw. leicht kürzere) Zeit für den Bahnverlauf zu gelangen, da die Vorsteuerung bereits Filter zu Differenzierung der Bahn enthält.

- Var. 1.: Einfache Starrkörpervorsteuerung $\boldsymbol{\Sigma}_{ff}^{(r)}$. Keine Elastizität wird in der Vorsteuerung berücksichtigt.
- Var. 2.: Vorsteuerung $\boldsymbol{\Sigma}_{ff}^{(RLFJ)}$. Elastizität der Getriebe (nichtlinear) und Reibung wird in der Vorsteuerung berücksichtigt. Die Struktur des Roboters wird als starr angenommen (RLFJ-Modell).
- Var. 3.: Vorsteuerung $\boldsymbol{\Sigma}_{ff}^{(ASIM)}$. Elastizität der Getriebe (nichtlinear) und der Struktur sowie Reibung wird in der Vorsteuerung berücksichtigt.

Neben der PTP-Bewegung der Grundachsen wurde auch eine kartesische Trajektorie mit der Kamera aufgezeichnet. Die Messung zeigt Abb. 6.5. Kartesische Trajektorien lassen sich allerdings nur sehr schwer vergleichen, da die Vorfilterkonfigurationen $\boldsymbol{\Sigma}^*$, welche auf Achsebene wirken, die (kartesischen-) Bahnen leicht verzerren. Ausgangspunkt ist daher nicht die exakte kartesische (rechteckige) Bahn, sondern die mit $\boldsymbol{\Sigma}^*$ auf Achsebene gefilterte Bahn, welche durch die Filterung auch differenzierbar ist. In Tabelle 6.1 sind die resultierenden Abweichungen von der (mit entsprechendem $\boldsymbol{\Sigma}^*$ gefilterten) kartesischen Solltrajektorie $\boldsymbol{r}_{t,s}^{(I)}$ bezüglich des inertialen Koordinatensystems aufgetragen. Die Solltrajektorie wurde mit Hilfe der gefilterten Solldaten $\boldsymbol{\varphi}_{a,r}$ und einem kartesischen (starren) Kinematikmodell ${}_0^L\boldsymbol{T}^{(r)}$ des Roboters berechnet. In Tabelle 6.1 ist sowohl der über die Messdauer t_e integrierte Fehler der Abweichung als auch der maximale Fehler aufgetragen.

Messung	$\boldsymbol{\Sigma}_{ff}^{(r)}$	$\boldsymbol{\Sigma}_{ff}^{(RLFJ)}$	$\boldsymbol{\Sigma}_{ff}^{(ASIM)}$
$\int_0^{t_e} \|\boldsymbol{r}_t^{(I)} - \boldsymbol{r}_{t,s}^{(I)}\| dt$ (norm.)	1	0,5744	0,2464
$\max\left(\|\boldsymbol{r}_t^{(I)} - \boldsymbol{r}_{t,s}^{(I)}\|\right)$ (norm.)	1	0,5831	0,2528

Tabelle 6.1: Abweichungen von der Sollbahn nach Abb. 6.5 für unterschiedliche Vorsteuerungen. Die Werte sind normiert auf die Abweichungen von $\boldsymbol{\Sigma}_{ff}^{(r)}$.

6.3 Experimentelle Untersuchungen am Roboter KR210

Analog zum Roboter KR16 wurden auch Messungen für den KR210 mit einer 210 kg Hantel durchgeführt. Hierbei wurde das approx. inverse Modell nach Abschnitt 3.8.3 eingesetzt, das im Folgenden mit ASIM-KR210 abgekürzt wird.

6.3.1 Kriterium

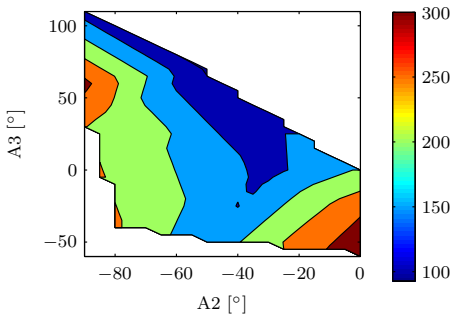
Das Kriterium wurde identisch wie beim KR16 nach Gl. (6.1) gebildet. Der Beschleunigungssensor befand sich auch hier am Handwurzelpunkt des Roboters.

6.3.2 Messungen

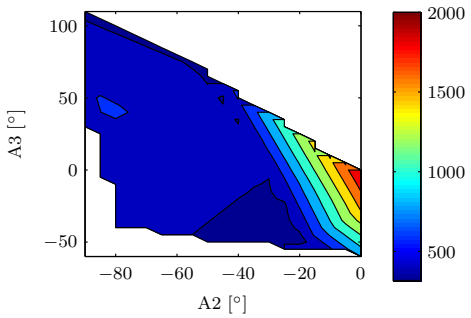
Analog[4] zum KR16 werden im Folgenden Ergebnisse der Messungen gezeigt (Abbildung 6.7, 6.6 und 6.8). Es wurden jeweils simultane 5° und 20° Bewegungen der Grundachsen (A1 bis A3) gefahren. Die Messungen befinden sich im üblichen Arbeitsbereich des Roboters und wurden zwischen den Stützstellen interpoliert. Messpunkte sind in den 3D-Darstellungen durch Kreise gekennzeichnet. Aufgetragen ist jeweils das Kriterium über den Startwinkeln der Achsen 2 und 3.

Wie man den Abbildungen entnehmen kann, ergibt sich ein ähnliches Bild wie beim KR16. Der Einfluss der Struktursteifigkeit ist auch beim KR210 stark stellungsabhängig. Eine Berücksichtigung Strukturelastizität führt auch hier bei vielen Stellungen zu deutlichen Verbesserungen im Positionierverhalten.

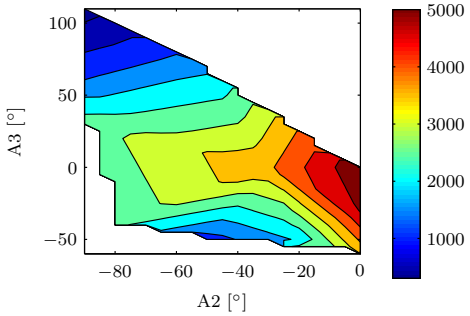
[4]Die reine Starrkörpervorsteuerung wurde beim KR210 nicht aufgeführt, da diese zu extremen Schwingungen am Roboter führt welche den Roboter beschädigen könnten.



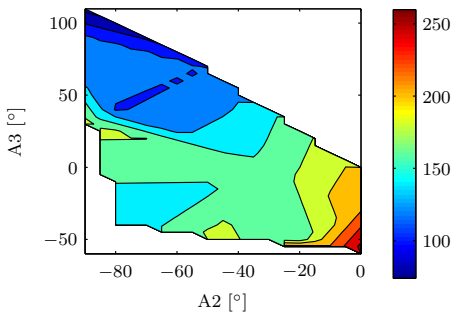
(a) Kriterien für simultane 5° Bewegung der Grundachsen mit ASIM-KR16.



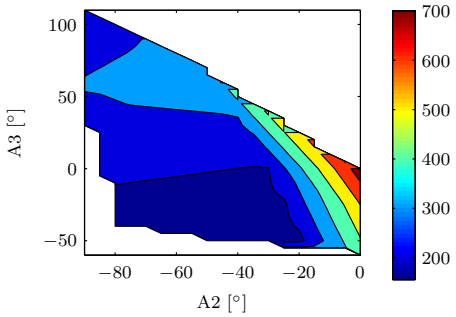
(b) Kriterien für simultane 5° Bewegung der Grundachsen mit RLFJ-Vorsteuerung.



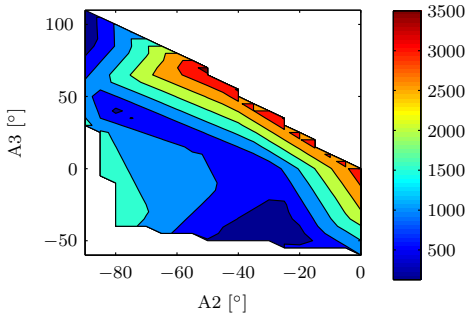
(c) Kriterien für simultane 5° Bewegung der Grundachsen mit Starrkörpervorsteuerung.



(d) Kriterien für simultane 20° Bewegung der Grundachsen mit ASIM-KR16.

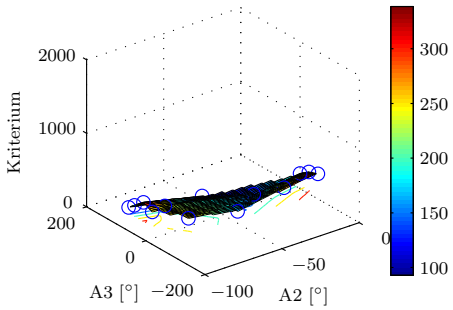


(e) Kriterien für simultane 20° Bewegung der Grundachsen mit RLFJ-Vorsteuerung.

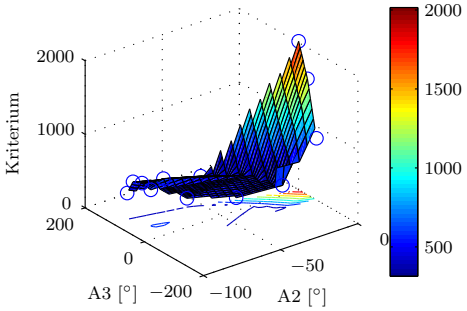


(f) Kriterien für simultane 20° Bewegung der Grundachsen mit Starrkörpervorsteuerung.

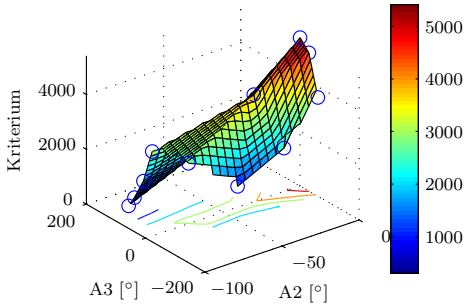
Abbildung 6.2: KR16: Kriterien für approx. strukturelastisches inverses Modell (ASIM-KR16), RLFJ-Vorsteuerung und Starrkörpervorsteuerung für simultane 5° und 20° Bewegungen der Grundachsen (Kontur-Darstellung).



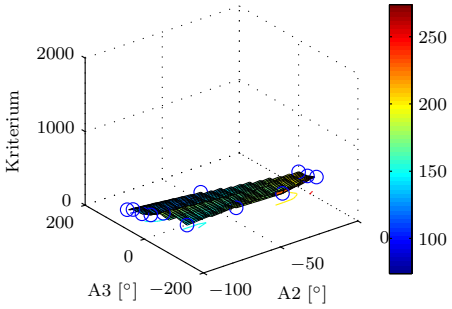
(a) Kriterien für simultane 5° Bewegung der Grundachsen mit ASIM-KR16.



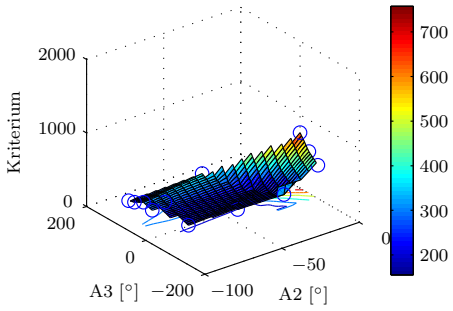
(b) Kriterien für simultane 5° Bewegung der Grundachsen mit RLFJ-Vorsteuerung.



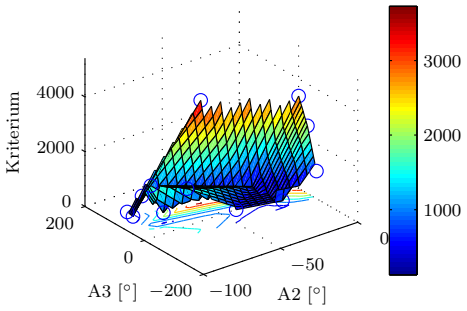
(c) Kriterien für simultane 5° Bewegung der Grundachsen mit Starrkörpervorsteuerung.



(d) Kriterien für simultane 20° Bewegung der Grundachsen mit ASIM-KR16.

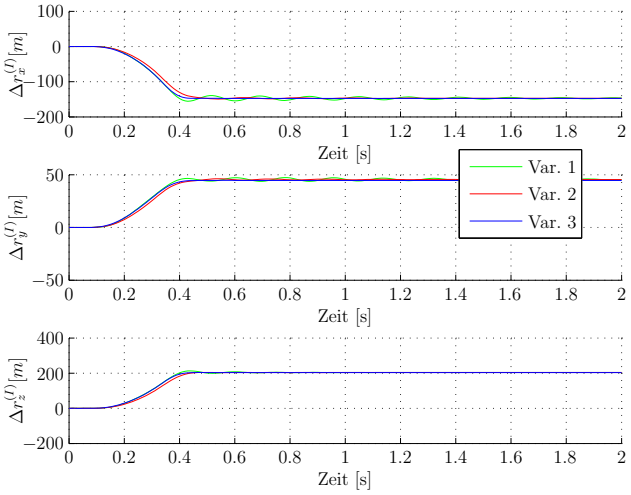


(e) Kriterien für simultane 20° Bewegung der Grundachsen mit RLFJ-Vorsteuerung.

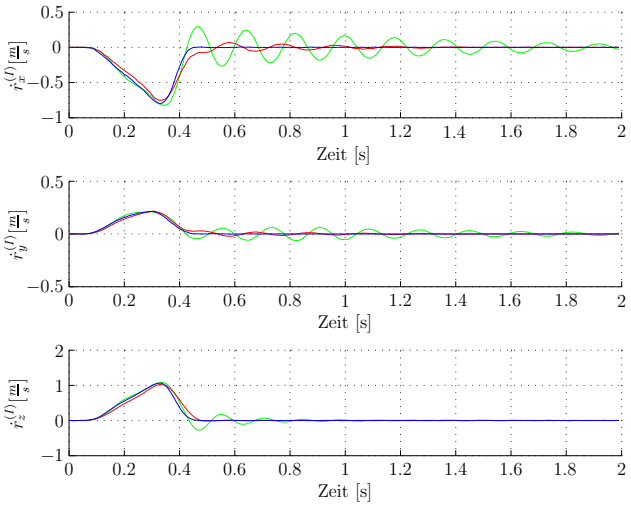


(f) Kriterien für simultane 20° Bewegung der Grundachsen mit Starrkörpervorsteuerung.

Abbildung 6.3: KR16: Kriterien für approx. strukturelastisches inverses Modell (ASIM-KR16), RLFJ-Vorsteuerung und Starrkörpervorsteuerung für simultane 5° und 20° Bewegungen der Grundachsen (Oberflächen-Darstellung).

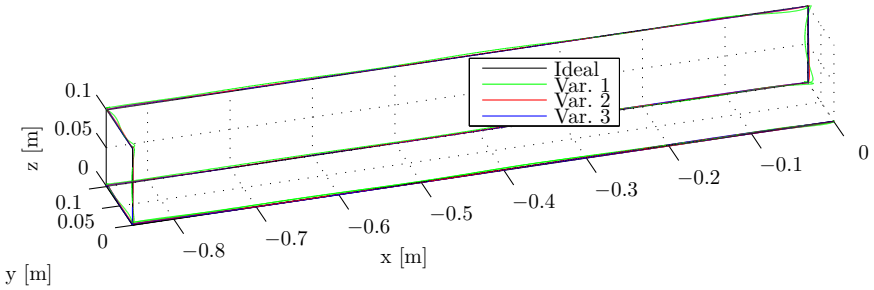


(a) Position des TCP bei einer simultane 5° Bewegung der Grundachsen aus der Strecklage.

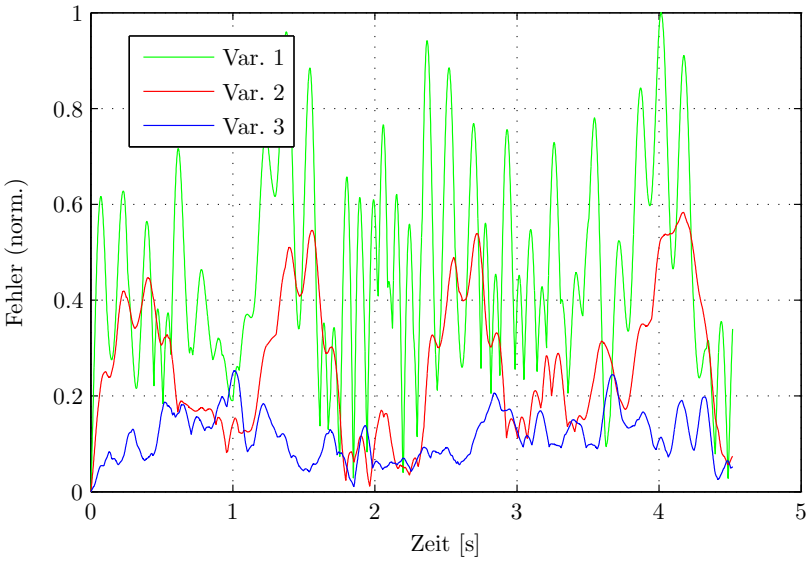


(b) Geschwindigkeit des TCP bei einer simultane 5° Bewegung der Grundachsen aus der Strecklage.

Abbildung 6.4: KR16: Relative Positionsänderung $\Delta \boldsymbol{r}_t^{(I)}$ und Geschwindigkeit $\dot{\boldsymbol{r}}_t{}^{(I)}$ des TCPs bei einer simultane 5° Bewegung der Grundachsen aus der Strecklage. Gemessen mit 6D-Kamera, bezüglich des Inertialsystems I des Roboters mit jeweils unterschiedlicher Vorsteuerung $\boldsymbol{\Sigma}_{ff}^{(r)}$ (Var. 1), $\boldsymbol{\Sigma}_{ff}^{(RLFJ)}$ (Var. 2) und $\boldsymbol{\Sigma}_{ff}^{(ASIM)}$ (Var. 3).

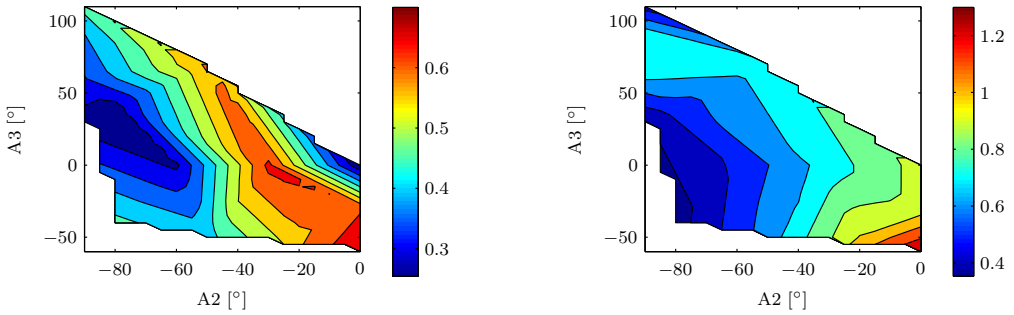


(a) Verlauf der, mit der 6D-Kamera aufgezeichneten, Bahn des TCPs des Roboters.



(b) Abweichung (normiert) der Bahn von der Sollbahn über der Zeit.

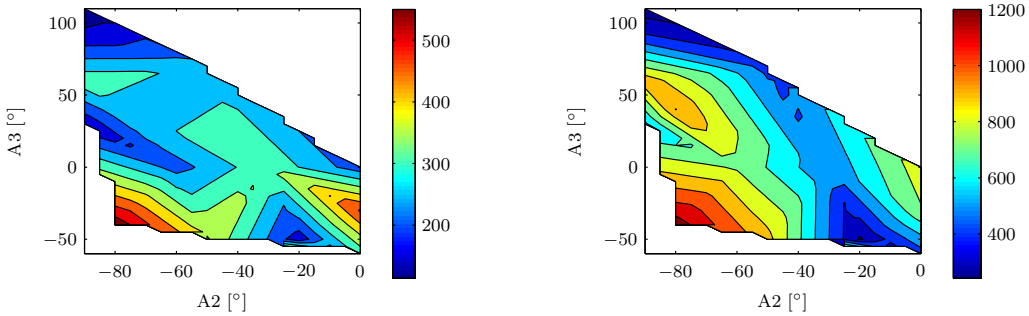
Abbildung 6.5: KR16: Relative Positionsänderung $\Delta \boldsymbol{r}_t^{(I)}$ des TCPs bei einer kartesischen, linearen Bewegung. Gemessen mit 6D-Kamera, bezüglich des Inertialsystems I des Roboters mit jeweils unterschiedlicher Vorsteuerung $\boldsymbol{\Sigma}_{ff}^{(r)}$ (Var.1), $\boldsymbol{\Sigma}_{ff}^{(RLFJ)}$ (Var.2) und $\boldsymbol{\Sigma}_{ff}^{(ASIM)}$ (Var.3). Die Abweichungen von der Sollbahn in Abb. 6.5(b) sind normiert auf den maximalen Fehler, welcher mit der Vorsteuerung $\boldsymbol{\Sigma}_{ff}^{(r)}$ entsteht. Der Regler ist jeweils identisch.



(a) Relativer Vergleich (ASIM-KR210 zu RLFJ-Vorsteuerung) der Kriterien für eine simultane 5° Bewegung der Grundachsen.

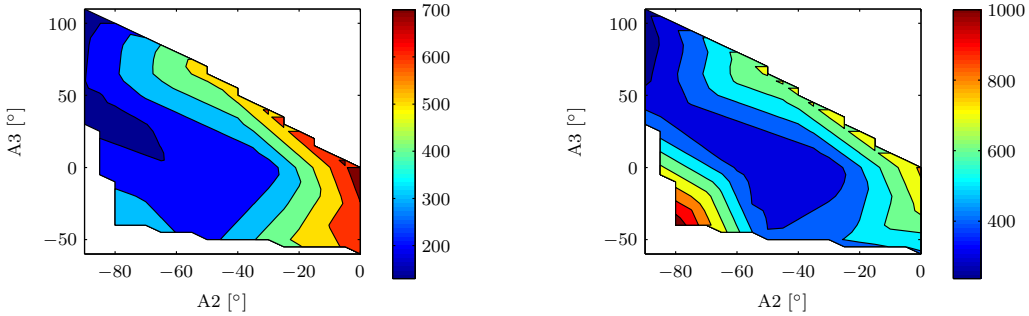
(b) Relativer Vergleich (ASIM-KR210 zu RLFJ-Vorsteuerung) der Kriterien für eine simultane 20° Bewegung der Grundachsen.

Abbildung 6.6: KR210: Relatives Kriterium $C_R = \left(\frac{C_{ASIM}}{C_{RLFJ}}\right)$ für approx. strukturelastisches inverses Modell (ASIM-KR210) bezüglich RLFJ-Vorsteuerung für simultane 5° und 20° Bewegungen der Grundachsen. Der verwendete Regler ist jeweils identisch (Regelungsstruktur mit zwei Freiheitsgraden).



(a) Kriterien für simultane 5° Bewegung der Grundachsen mit ASIM-KR210.

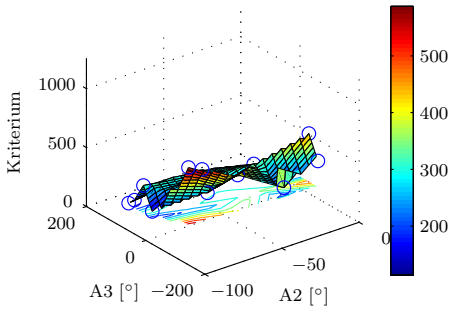
(b) Kriterien für simultane 5° Bewegung der Grundachsen mit RLFJ-Vorsteuerung.



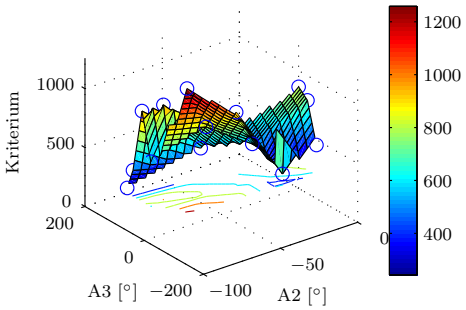
(c) Kriterien für simultane 20° Bewegung der Grundachsen mit ASIM-KR210.

(d) Kriterien für simultane 20° Bewegung der Grundachsen mit RLFJ-Vorsteuerung.

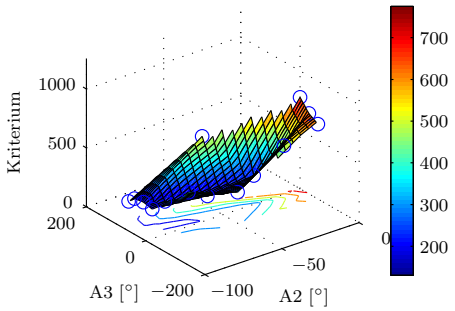
Abbildung 6.7: KR210: Kriterien für approx. strukturelastisches inverses Modell (ASIM-KR210) und RLFJ-Vorsteuerung für simultane 5° und 20° Bewegungen der Grundachsen (Kontur-Darstellung).



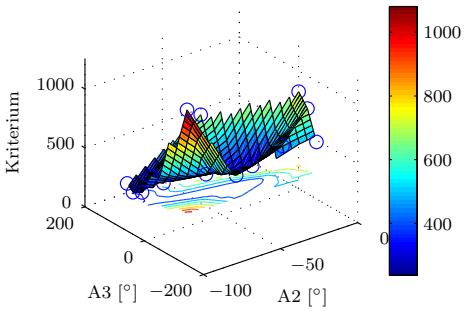
(a) Kriterien für simultane 5° Bewegung der Grundachsen mit ASIM-KR210.



(b) Kriterien für simultane 5° Bewegung der Grundachsen mit RLFJ-Vorsteuerung.



(c) Kriterien für simultane 20° Bewegung der Grundachsen mit ASIM-KR210.



(d) Kriterien für simultane 20° Bewegung der Grundachsen mit RLFJ-Vorsteuerung.

Abbildung 6.8: KR210: Kriterien für approx. strukturelastisches inverses Modell (ASIM-KR210)und RLFJ-Vorsteuerung für simultane 5° und 20° Bewegungen der Grundachsen (Oberflächen-Darstellung).

Kapitel 7

Zusammenfassung und Ausblick

In dieser Arbeit „Modellierung und Steuerung von strukturelastischen Robotern“ werden neuartige Konzepte zur Modellierung, Steuerung und Identifikation sowie Trajektorienoptimierung von strukturelastischen Robotern vorgestellt und experimentell unter realistischen Bedingungen validiert.

Motiviert ist dieser Ansatz dadurch, dass es notwendig ist, neben der Elastizität der Getriebe auch die Elastizität der Struktur und der Gelenklagerungen zu berücksichtigen, um die Leistungsfähigkeit von Robotern gegenüber dem aktuellen Stand der Technik entscheidend zu verbessern.

Im Gegensatz zu vielen anderen Arbeiten auf diesem Gebiet, wird dieses Ziel hier ohne zusätzliche Sensorik erreicht, d.h. zur Regelung werden nur Winkel-Resolver an den Achsen der Motoren verwendet.

Grundlegend dabei war der Aufbau einer neuen Roboter-Modell-Bibliothek. Diese besteht aus modular aufgebauten nichtlinearen Modellkomponenten, welche in der Modellierungssprache *Modelica* erstellt wurden. Diese beinhaltet sowohl komplexe Module für hochgenaue Referenzmodelle, als auch stetig differenzierbare Approximation für invertierbare Synthesemodelle.

Die elastischen Strukturteile des Roboters können mit der Modell-Bibliothek in unterschiedlichen Detaillierungsgraden modelliert werden. Hierzu zählen modal reduzierte FEM-Modelle, welche aus CAD-Daten der Bauteile oder als Balkenmodelle modelliert werden können. Zudem wurden auch echtzeitfähige Balkenersatzmodelle erstellt, welche als Mehrkörpersysteme mit zusätzlichen Gelenken gestützt von Feder-Dämpferelementen modelliert werden. Ergänzt wird die Bibliothek durch umfangreiche Antriebsstrangskomponenten. Hierzu zählen neu entwickelte invertierbare Approximation für Getriebe-Hysterese sowie vom Kraftfluss abhängiger Wirkungsgrad und nichtlineare Getriebesteifigkeiten.

Über ein neuartiges, mehrstufiges Identifikations-Verfahren kann eine hohe Parameteranzahl mit möglichst geringem Rechen- und Messaufwand abgedeckt werden. Als Ausgangspunkt für die Parameter müssen nur ungenaue Startwerte und Parameterbereiche zur Verfügung stehen. Mit diesem Identifikations-Verfahren kann eine sehr gute Übereinstimmung der Modelle im gesamten Arbeitsbereich des Roboters, mit Messdaten auf der Basis von globalen Optimierungsalgorithmen erreicht werden. Für

das mehrstufige Optimierungskonzept werden die Parameter in unterschiedliche Kategorien geteilt. Diese werden in mehreren Schritten identifiziert.

Zusätzlich erfolgt die Optimierung in Stufen, in denen die Parameter immer genauer bestimmt werden. Dazu dienen unterschiedliche Modell- und Optimierungs-Konfigurationen mit ansteigender Komplexität. Diese unterscheiden sich in Aufbau und Genauigkeit und damit auch in der benötigten Rechenzeit entscheidend.

Das Verfahren ist so aufgebaut, dass es parallelisierbar auf einem Rechen-Cluster berechnet werden kann, wodurch eine Parameteridentifikation Offline anhand von Messdaten automatisierbar ablaufen kann.

Da die exakte Invertierung eines strukturelastischen Robotermodells zu einer instabilen Lösung führt wurden neue Approximationsverfahren zur Invertierung entwickelt. Die Invertierung ist notwendig, um aus den erstellten Modellen eine modellbasierte Steuerung, als Teil einer Regelung mit zwei Freiheitsgraden, abzuleiten.

Analysiert man die systemdynamischen Eigenschaften der nichtlinearen Modelle, so zeigt sich, dass eine elastische Modellierung der Struktur des Roboters dazu führen kann, dass eine instabile Nulldynamik auftritt, und hierdurch keine exakte Invertierung der Modelle möglich ist, wenn die Position und Orientierung der Werkzeugspitze des Roboters vorgegeben werden soll.

Eine Variante, auf Basis von Balkenersatzmodellen, löst das Invertierungsproblem über eine neuartige quasi-statische Kompensation, welche über ein paralleles Modell berechnet wird. Das Verfahren kann auf ein Robotermodell mit 6 Achsen und zusätzlichen elastischen Freiheitsgraden in Echtzeit (auf aktuellen Standard-Prozessoren) angewandt werden. Die Verformungen werden approximativ über das Lösen von DGL-Systemen erzielt, so dass auch im dynamischen Fall eine gute Approximation für die Verformungen der Struktur berechnet werden kann, in welcher auch die Dämpfung im System berücksichtigt wird.

Die inverse Kinematik wird in diesem Ansatz über einen robusten Lösungsansatz mit Hilfe eines Dämpfungsterms berechnet, so dass auch Lösungen in Singularitäten oder – durch die Verformungen des Roboters – nicht mehr erreichbare Wunschpositionen und Orientierungen der Werkzeugspitze erzielt werden können. Die Invertierung erfolgt nach der Kompensation der Verformungen über Indexreduktion auf Basis von DAE-Modellen, wodurch keine Linearisierung notwendig ist und Nichtlinearitäten (in stetig differenzierbarer Approximation) komplett berücksichtigt werden können.

Auch für aus FEM-Komponenten aufgebaute strukturelastische Robotermodelle wurde eine neue approximative Invertierungsmethode entwickelt. Diese Methode basiert auf einer strukturellen Entkopplung der für die Strukturelastizität verantwortlichen Terme und der Starrköperanteilen in den Bewegungsgleichungen für modal reduzierte FEM-Modelle und erlaubt eine approximative Invertierung von Robotermodellen, die diese Komponenten beinhalten.

Zusätzlich wurde untersucht wie es über eine Invertierung bezüglich neu definierter oder virtueller Ausgänge des zu invertierenden Systems möglich ist die interne Dynamik (bzw. Nulldynamik) des Systems so zu ändern, dass über differential-geometrische DAE-Invertierungsmethoden eine Näherungslösung für das System gefunden werden kann. Hierzu wurden bekannte Methoden, um neuartige Konzepte erweitert.

Auf Basis der inversen Modelle wurde zudem ein neues Regelungskonzept vorgestellt, bei welchem Fehler, welche durch die Approximationen in den inversen Modellen entstehen können, durch eine zusätzliche modellbasierte Regelung innerhalb der Vorsteuerung, auf Basis des nicht-approximierten Robotermodells, kompensiert werden können.

Dieses Verfahren ist allgemein auf Systeme anwendbar, bei denen nur eine approximative Inverse des Systems berechenbar ist.

Weiterhin wurde eine neue Methode für eine zeitoptimale (bzw. energieoptimale) Trajektorienoptimierung eines strukturelastischen Roboters entwickelt. Durch die Verwendung eines approximativen inversen, strukturelastischen Robotermodells wird die Optimierung um etwa ein bis zwei Größenordnungen schneller und sehr viel robuster, da die Lösungsmenge für den Optimierer stark reduziert wird. Mit dieser neuen Methode wurden Trajektorien-Optimierungen durchgeführt, welche die komplette nichtlineare Dynamik und Elastizität der Antriebsstränge und Strukturteile berücksichtigen, um zeitoptimale Roboter-Trajektorien - unter Einhaltung von Stellgrößenbeschränkungen und Belastungsgrenzen - zu berechnen. Die berechneten Trajektorien reduzieren die Verfahrzeit eines Roboters typischerweise um 10 bis 20 Prozent: Verglichen mit den zeitoptimalen Trajektorien eines Starrkörperroboters, die bei einem strukturelastischen Roboter verwendet werden.

Durch die Verwendung der inversen Modelle in der Optimierung entfallen rechentechnisch aufwendige Verfahren wie das Mehrfachschiessen, um etwa Pfadbeschränkungen zu berücksichtigen, da diese bereits durch das inverse Modell erfüllt werden. Das Trajektorien-Optimierungsproblem kann mit diesem Verfahren auf ein Parameter-Optimierungsproblem überführt werden.

Die in der Arbeit vorgestellten approximativen inversen Modelle wurden zudem experimentell untersucht und verifiziert. Hierzu wurden sie zur Vorsteuerung von zwei unterschiedlichen Robotern (mit 16 und 210 kg Traglast) getestet. Gemessen wurde eine deutliche Verbesserung im Ausschwing- und Fahrverhalten der Roboter gegenüber Vorsteuerungen, welche nur Elastizitäten in den Antriebssträngen (und nicht in der Struktur des Roboters) berücksichtigen. Die erstellten Vorsteuerungen sind auf aktuellen Standard-Prozessoren echtzeitfähig und müssen nicht vorab offline berechnet werden.

In naher Zukunft sollte es mit der steigenden Rechenleistung zukünftiger Steuerungen möglich sein, auch die in dieser Arbeit vorgestellten rechentechnisch aufwendigeren Verfahren in Echtzeit zu lösen, um sie damit industriell einsetzbar zu machen. Dadurch wäre eine zusätzliche nicht unerhebliche Leistungssteigerung der Roboter zu erwarten.

Anhang A

Anhang

A.1 Das KUKA Roboter-System

Die von KUKA[1] produzierten Roboter sollen in diesem Abschnitt kurz vorgestellt werden. Die Standard-Industrieroboter von KUKA besitzen 6 Achsen. Diese sind so angeordnet, dass sich sechs Freiheitsgrade im Raum ergeben. Sie sind eingeteilt nach ihrer maximalen Traglast, die sie bei voller Dynamik bewegen können. Die Palette reicht (momentan) vom KR3 (für 3 kg Traglast) bis zum KR1000 (1000 kg Traglast). Abb. A.1 zeigt einige der aktuellen Modelle. Der Bediener steuert das Robotersystem über ein Handbediengerät, KCP (KUKA Control Panel) genannt (siehe Abb.A.2). Über das KCP kann der Anwender den Roboter von Hand steuern, ihn programmieren und sich über den aktuellen Status des Roboters informieren. Zum Verfahren des Roboters stehen dem Benutzer am KCP eine Maus mit sechs Freiheitsgraden, Spacemaus oder 6D-Maus genannt, sowie Verfahrtasten zur Verfügung. Roboter-Programme werden in der Programmiersprache KRL[2] geschrieben. Dies ist direkt am KCP oder mit einem externen Editor möglich. Die KRL ist ähnlich wie die Programmiersprache PASCAL aufgebaut. Durch zusätzliche Interrupt-Funktionen und das Setzten von externen Ein- und Ausgängen können jedoch auch relativ komplexe Programme geschrieben werden. über die Ein- und Ausgänge kann mit einer übergeordneten (SPS-)Steuerung kommuniziert werden.

Anzufahrende Punkte können entweder numerisch eingegeben werden oder der Roboter wird über die Verfahrtasten oder die Spacemaus in die entsprechende Position gefahren. Durch einen Tastendruck kann der aktuelle Punkt (des TCPs[3]) in das Programm aufgenommen werden. Dieser Vorgang wird mit „Teachen“ bezeichnet. Im Roboterprogramm können solche Punkte auf bestimmten Bahnen miteinander verbunden werden, wobei die Bahnen nicht direkt durch diese Punkte führen müssen. Dabei werden zwei grundsätzliche Bewegungsarten unterschieden. Die PTP-Bewegung (point to point) sowie die CP-Bewegung (continuos path, lineare kartesische Bewegung). Bei der PTP-Bewegung versucht der Roboter möglichst schnell den entsprechenden Punkt zu erreichen. Dabei drehen sich die Achsen des Roboters gleichzeitig, wobei die Achse,

[1] Keller und Knappich, Augsburg, KUKA Roboter GmbH
[2] KUKA Robot Language
[3] Tool Center Point, Werkzeugspitze

(a) Niedrige Traglasten : KR16

(b) Mittlere Traglast : KR30

(c) Hohe Traglast : KR210

(d) Schwerlast : KR1000-Titan

Abbildung A.1: KUKA Roboter für unterschiedliche Traglasten (Abbildung: KUKA).

Abbildung A.2: Das KUKA Control Panel (Abbildung: KUKA).

welche die größte Verdrehung zu bewerkstelligen hat, die Zeit bestimmt, welche den einzelnen Achsen für ihre Verdrehung zur Verfügung steht. Dies ist die schnellste Bewegungsart, um den Roboter von Punkt A nach Punkt B zu bewegen. Soll die Werkzeugorientierung während einer Bewegung konstant gehalten werden oder mit einer definierten Geschwindigkeit verfahren werden, werden CP-Bewegungen verwendet. Bei diesen verfährt der Roboter auf einer Geraden im Raum zwischen zwei Punkten. Falls die zwei verwendeten Punkte unterschiedliche Orientierungen aufweisen, wird zwischen diesen linear interpoliert. Eine Variante der CP-Bewegung ist die CIRC-Bewegung[4], bei der eine Kreisbahn aus drei Punkten interpoliert wird. Für weitere Details sei auf [118] verwiesen.

A.2 Bezeichnungen

Die wichtigsten Abkürzungen und Formelzeichen dieser Arbeit sind im Folgenden zusammengestellt.

A.2.1 Abkürzungen

Abkürzung	Beschreibung
DLR	Deutsches Zentrum für Luft- und Raumfahrt (e.V.)
TCP	Werkzeugspitze, bzw. Werkzeug-Vermessungspunkt (tool center point)
HWP	Handwurzelpunkt. Schnittpunkt der Achsen vier bis sechs.
MOPS	Multi-Objective Parameter Synthesis
SISO	Single-Input & Single-Output
MIMO	Multiple-Input & Multiple-Output
PTP	Punkt-zu-Punkt (point to point)
CP	Lineare (kartesische) Bewegung (continuos path)
HIL	Hardware in the Loop

[4]Circular, kreisförmig

FEM	Finite-Elemente-Methode
CAD	Rechnerunterstützte Konstruktion (computer aided design)
KUKA	KUKA Roboter GmbH (Keller und Knappich)
KR16	Industrieroboter KUKA KR16 (Nutzlast: 16 kg)
KR210	Industrieroboter KUKA KR210 (Nutzlast: 210 kg)
DAE	Differential-algebraische Gleichung (differential algebraic equation)
DOB	Disturbance Observer
IDOB	Inverse Disturbance Observer
RLFJ	Roboter(-Modellvorstellung) mit starrer Struktur und elastischen Antriebssträngen (rigid link flexible joint robot manipulator)

A.2.2 Typografische Kennzeichnungen

Art	**Beschreibung**	**Beispiel**
Vektor	fettgedruckter Kleinbuchstabe	$\boldsymbol{x}$
Matrix	fettgedruckter Großbuchstabe	$\boldsymbol{M}$
Geschätzte Größe	gekennzeichnet mit $\hat{(.)}$	$\hat{a}$
Transponierter Vektor bzw. Matrix	gekennzeichnet mit $(.)^T$	$\boldsymbol{a}^T, \boldsymbol{A}^T$
1. & 2. Zeitableitung	gekennzeichnet mit Punkten	$\dot{a}, \ddot{a}$
n. Zeitableitung	gekennzeichnet mit $(.)^{(n)}$	$a^{(3)}$
Motorseitige Größe	gekennzeichnet mit $(.)_m$	φ_m
Abtriebsseitige Größe	gekennzeichnet mit $(.)_a$	φ_a
Starrkörper Größe	gekennzeichnet mit $(.)_r$ (rigid)	$\varphi_{a,r}$
Strukturelastische Größe	gekennzeichnet mit $(.)_e$	$\hat{\varphi}_{e,j}$
Inverse, Umkehrfunktion	gekennzeichnet mit $(.)^{-1}$	$\boldsymbol{f}^{-1}$
Pseudoinverse	gekennzeichnet mit $(.)^+$	$\boldsymbol{J}_f^+$

A.2.3 Formelzeichen

Symbol	**Beschreibung**
c	Federsteifigkeit
d	Dämpfung
η	Wirkungsgrad
$\boldsymbol{\tau}_m$	Motormoment-Vektor
$\boldsymbol{I}_m$	Motorstrom-Vektor
T_m	Motor-Zeitkonstante
τ_r	Reibmoment
τ_h	Haftreibung
τ_v	Viskose Reibung (Gleitreibung)
s	Steilheit der Exponentialfunktion ($s \gg 1$)
A, B bzw. a, b	Flansch-Beschreibung
i	Getriebeübersetzung, Index-Variable
ω	Winkelgeschwindigkeit

ξ	Approx. Kraftflussrichtung
T_m	Motor-Zeitkonstante
$\mathfrak{P}_{r_h}$	Play-Operator
$\mathfrak{H}$	Prandtl-Ishlinskii-Hystereseoperator
$G(s)$	Übertragungsfunktion mit Laplace-Variablen s
φ	Winkel
$\boldsymbol{\varphi}_m$	Motorseitiger Winkelvektor
$\boldsymbol{\varphi}_a$	Abtriebsseitiger Winkelvektor
$\boldsymbol{r}$	Ortsvektor (kann auch Winkel beinhalten)
$\boldsymbol{r}_t$	Translatorischer Anteil des Ortsvektors
$\boldsymbol{v}$	Geschwindigkeitsvektor
$\boldsymbol{a}$	Beschleunigungsvektor
$\boldsymbol{\Phi}$	Modalformen
$\boldsymbol{q}$	Modale Amplituden, Winkelvektor
m, ρ	Masse, Dichte
$\boldsymbol{M}$	Massenmatrix
$\boldsymbol{\Theta}, \Theta$ bzw. J	Trägheitstensor, Trägheit
l, A	Länge, Fläche
E, G	Elastizitätsmodul, Schubmodul
I	Flächenträgheitsmoment
δ	Dämpfung von Moden
${}^{l}_{k}\boldsymbol{T}$	Transformationsmatrix von Frame l nach Frame k
$\boldsymbol{R}$	Rotationsmatrix
$\boldsymbol{f}$	Vektorfunktion
$\boldsymbol{x}_0$	Anfangszustand
$\boldsymbol{x}$	Zustandsvektor
$\boldsymbol{u}$	Eingangsvektor
$\boldsymbol{y}$	Ausgangsvektor
$\boldsymbol{p}$	Parametervektor
$\mathfrak{G}$	Gütefunktion, Kriterium
$\boldsymbol{\Sigma}^*$	Vorfilter
$\boldsymbol{\Sigma}$, $\boldsymbol{\Sigma}^m$	Strecke, Streckenmodell
$\boldsymbol{\Sigma}_{ff}$, $\boldsymbol{\Sigma}_{ff}^{(r)}$	Vorsteuerung, Starrkörpervorsteuerung (feed-forward)
$\boldsymbol{\Sigma}_{fb}$	Regler (feed-back)
$(\boldsymbol{\Sigma}^m)^{-1}$	(Approximatives) strukturelastisches inverses Roboter Modell
$\boldsymbol{\Sigma}_{ff}^{(ASIM)}$	Approximatives strukturelastisches inverses Robotermodell als Vorsteuerung
$A1, A2, A3, \ldots$	Achsen des Roboters. Achse ein bis drei sind Grundachsen, Achsen vier bis sechs Handachsen
t	Zeit
L	Lie-Ableitung
f_c	Eckfrequenz, Cut-Off-Frequenz
$\boldsymbol{g}$	Gravitationsvektor,
$\mathfrak{S}$	Nichtlineares System
$\boldsymbol{J}_f, \boldsymbol{J}_{f,t}$	Jacobimatrix, translatorische Jacobimatrix
$\mathfrak{T}$	Trajektorie
$\boldsymbol{e}$	Koordinatensystem mit Basisvektoren $\{\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z\}$

$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$	Matrizen eines linearen Systems
$\boldsymbol{F}(.)$	DAE-System

A.3 Lie-Ableitung

Die Lie-Ableitung $L_f\beta(\boldsymbol{x})$ stellt die Ableitung einer skalarwertigen Funktion $\beta(\boldsymbol{x})$ eines Vektors $\boldsymbol{x}$ entlang der n-vektorwertigen Funktion $\boldsymbol{f}(\boldsymbol{x})$ dar (A.1). $L_gL_f\beta(\boldsymbol{x})$ ist die wiederholte Ableitung entlang $\boldsymbol{f}(\boldsymbol{x})$ und dann entlang $\boldsymbol{g}(\boldsymbol{x})$.

$$\beta(\boldsymbol{x}) = \beta(x_1, x_2, \ldots, x_n) \tag{A.1a}$$

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{pmatrix} f_1(x_1, x_2, \ldots, x_n) \\ \vdots \\ f_n(x_1, x_2, \ldots, x_n) \end{pmatrix} \tag{A.1b}$$

$$L_f\beta(\boldsymbol{x}) = \sum_{i=1}^{n} \frac{\partial\beta(\boldsymbol{x})}{\partial x_i} f_i(x_1, x_2, \ldots, x_n) \tag{A.1c}$$

$$L_gL_f\beta(\boldsymbol{x}) = \sum_{i=1}^{n} \left(\frac{\partial}{\partial x_i} \left(\sum_{i=1}^{n} \frac{\partial\beta(\boldsymbol{x})}{\partial x_i} f_i(\boldsymbol{x}) \right) \right) g_i(\boldsymbol{x}) \tag{A.1d}$$

A.4 Kreuzprodukt als Matrixmultiplikation

Der $\tilde{(\cdot)}$-Operator steht in dieser Arbeit für eine schiefsymmetrische Matrix zur Realisierung des Kreuzprodukts als Matrixmultiplikation (Gl. (A.2)).

$$\boldsymbol{a} \times \boldsymbol{b} = \tilde{\boldsymbol{a}}\boldsymbol{b} \tag{A.2a}$$

$$\tilde{\boldsymbol{a}} = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix} \tag{A.2b}$$

A.5 Atan2-Funktion

Die atan2-Funktion stellt eine Modifikation des Arkustangens dar. Sie ist nach Gl. (A.3) definiert. Wird die atan2-Funktion mit nur einem Argument angegeben so entspricht dieses y und es wird $x = 1$ gesetzt.

$$\operatorname{atan2}(y, x) = \begin{cases} \arctan(\frac{y}{x}) & \text{für } x > 0 \\ \pi + \arctan(\frac{y}{x}) & \text{für } y \geq 0, x < 0 \\ -\pi + \arctan(\frac{y}{x}) & \text{für } y < 0, x < 0 \\ \frac{\pi}{2} & \text{für } y > 0, x = 0 \\ -\frac{\pi}{2} & \text{für } y < 0, x = 0 \\ 0 & \text{für } y = 0, x = 0 \end{cases} \tag{A.3}$$

A.6 Konvention der Achswinkel

Die Definition der Winkel der einzelnen Achsen zeigt Abbildung A.3. Die Nullstellung ist die ausgestreckte Lage des Roboters.

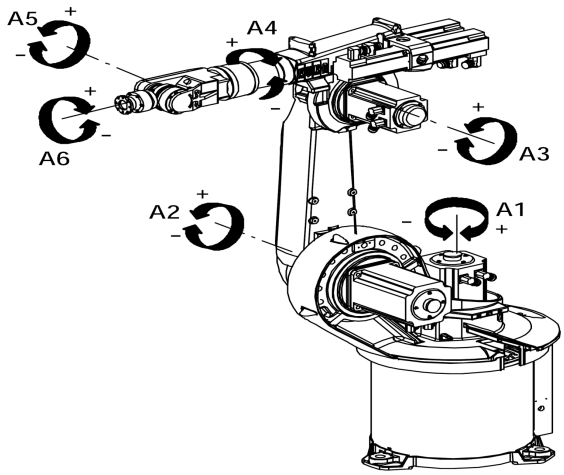


Abbildung A.3: Darstellung der Konvention der Achswinkel. Im Bild befindet sich der Roboter in der sog. Kanonenstellung (Abbildung: KUKA).

A.7 Multi-Objective Parameter Synthesis - MOPS

Das Programm „Multi-Objective Parameter Synthesis“ (kurz *MOPS*) wurde am Institut für Robotik und Mechatronik des DLR entwickelt wurde (siehe [114, 115]). *MOPS* stellt eine Reihe von Optimierungsverfahren zur Verfügung:

- SQP - sequential quadratic programming
- PATTERN - pattern search
- BOUNDS - quasi Newton
- GA - genetic algorithm
- PSO - particle swarm optimization

MOPS basiert auf *MATLAB* und kann sowohl über ein grafisches Interface (siehe Abbildung A.4), das als *MATLAB*-MEX Interface implementiert ist, als auch über *MATLAB*-Scripte, gesteuert werden. Die Software-Struktur von *MOPS* zeigt Abbildung A.6. Über die zentralen Skripte („model-run scripts“) wird der Ablauf der Optimierung gesteuert. Hierin werden die Initialisierung und die Ablaufsteuerung sowie die Visualisierung der Ergebnisse festgelegt. Die Simulationen der Modelle können mit unterschiedlichen Programmen erfolgen, wobei jeweils ein neuer Satz Parametern übergeben wird und Kriterien zurück geliefert werden müssen. Im Folgenden sollen einige Begriff-

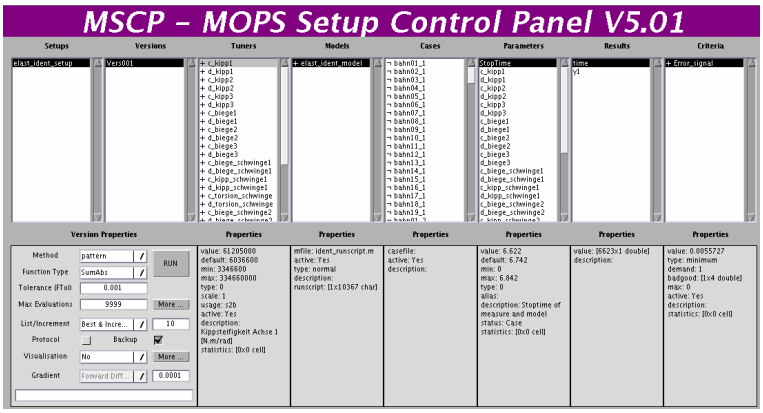


Abbildung A.4: Das *MOPS*-Interface, von welchem aus die Optimierung gesteuert werden kann.

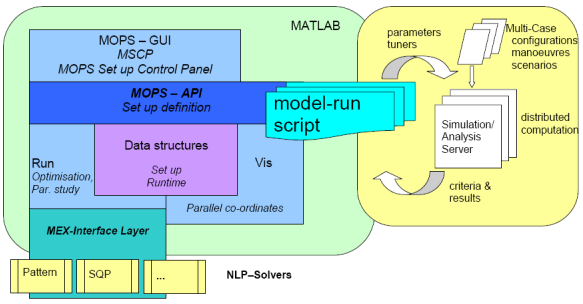


Abbildung A.5: Die Software Struktur von *MOPS* (aus [114]).

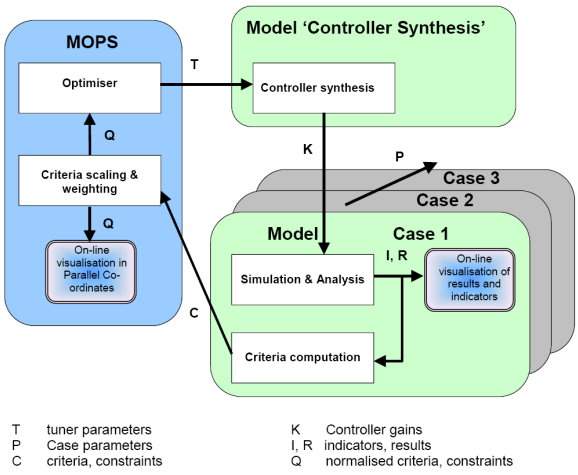


Abbildung A.6: Beispielhafte Darstellung einer robusten Optimierung eines Reglers in *MOPS* mit mehreren *Cases* (aus [114]).

lichkeiten kurz erläutert werden, welche in *MOPS* verwendet wurden und die auch in dieser Arbeit übernommen wurden.

- **Tuner:** Der Begriff Tuner wird in *MOPS* verwendet um eine sprachliche Trennung zwischen Parametern eines Modells, welche in die Optimierung einfließen und konstanten Parameter, zu erhalten. Tuner sind daher Parameter des Modells, welche vom Optimierungsalgorithmus (auch als Optimierer bezeichnet) variiert werden. Sie stellen daher eine Untermenge der Parameter des Modells dar.
- **Case:** Um eine Robustheit bei der Optimierung von Tuner-Variablen (also einem Satz von zu optimierenden Parametern) gegenüber Arbeitspunktänderungen, bzw. unterschiedlichen Anregungen des Modells zu erreichen können in *MOPS* sog. *Cases* definiert werden. Für jeden *Case* wird der gleiche Satz Tuner verwendet, allerdings können die übrigen Parameter, sowie die gewählte Anregung und das Optimierungskriterium für jeden einzelnen *Case* individuell gewählt werden.

A.8 Translation und Orientierung

Da in dieser Arbeit häufig mathematische Transformationen verwendet werden, sollen die wichtigsten kurz vorgestellt werden. Für weitere Informationen sei auf [84] verwiesen. Eine vollständige Darstellung von Translation und Orientierung wird in der Robotik häufig mit dem Begriff „Frame“ bezeichnet.

A.8.1 Rotationsmatrizen

Eine Rotationsmatrix $\mathbf{R}$ ist eine 3x3 Matrix, welche aus den 3 Basisvektoren $\mathbf{e}$ eines kartesischen Koordinatensystems besteht (A.4).

$$\mathbf{R} = \begin{pmatrix} e_{xx} & e_{xy} & e_{xz} \\ e_{yx} & e_{yy} & e_{yz} \\ e_{zx} & e_{zy} & e_{zz} \end{pmatrix} \tag{A.4}$$

Eine Rotations-Matrix ist orthogonal, d.h. es gilt Gleichung (A.5), was rechentechnisch viel vereinfacht.

$$\mathbf{R}^T = \mathbf{R}^{-1} \tag{A.5}$$

Zusätzlich gilt für die Determinante der Rotationsmatrix $det(\mathbf{R}) = 1$. Sollen mehrere Transformationen nacheinander erfolgen, können die einzelnen Rotationsmatrizen miteinander multipliziert werden. Die Reihenfolge darf dabei, wie bei allen Matrix-Multiplikationen, nicht vertauscht werden. Die Rotationsmatrix für eine Drehung *phi* um die x-Achse zeigt Gleichung (A.6), entsprechend (A.7) für die Drehung um die y-Achse und (A.8) für die Drehung um die z-Achse.

$$\mathbf{R}_x(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{pmatrix} \tag{A.6}$$

$$\mathbf{R}_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \tag{A.7}$$

$$\mathbf{R}_z(\psi) = \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{A.8}$$

Werden zwei Rotationsmatrizen miteinander multipliziert, so ist das Ergebnis wieder eine Rotationsmatrix, die alle angegebenen Eigenschaften besitzt.

A.8.2 Euler-Winkel & Roll-Pitch-Yaw

Euler-Winkel erlauben die Darstellung einer Drehung mit nur 3 Werten (ϕ, θ, ψ). Sie haben allerdings den Nachteil, dass es verschiedene Darstellungen mit den Euler Winkeln für die gleiche Rotation gibt. Es gibt verschiedene Definitionen für die Euler-Winkel. Sie unterscheiden sich darin, in welcher Reihenfolge um welche Achse gedreht wird. Die x-Konvention lautet wie folgt: Zuerst eine Drehung um die originale z-Achse um den Winkel ϕ, danach eine Rotation um den Winkel θ um die neue x-Achse und anschließend eine Rotation um den Winkel ψ um die nach den beiden vorherigen Drehungen erhaltene z-Achse. (A.9) zeigt diese Transformation als Rotationsmatrix mit der Abkürzung $\cos\psi = c\phi, \sin\psi = s\phi$ und analog für θ und ψ:

$$\mathbf{R}(\psi, \theta, \phi) = \begin{pmatrix} c\psi c\phi - c\theta s\phi s\psi & c\psi s\phi + c\theta c\phi s\psi & s\psi s\theta \\ -s\psi c\phi - c\theta s\phi c\psi & -s\psi s\phi + c\theta c\phi c\psi & c\psi s\theta \\ s\theta s\phi & -s\theta c\phi & c\theta \end{pmatrix} \tag{A.9}$$

Gebräuchlich ist die abc-Darstellung. Hierbei steht der a-Wert für eine Drehung um die z-Achse, der b-Wert für die Drehung um die y-Achse und der c-Wert die Drehung um die x-Achse. Dies entspricht der Definition der Roll-Pitch-Yaw Werte in der Luft- und Raumfahrt. Im Automobilbau entspricht dies Wanken-Nicken-Gieren. Wobei Roll einer Rotation um die x-Achse, Pitch einer Rotation um die y-Achse und Yaw einer Rotation um die z-Achse entspricht. Hierbei wird nicht um die durch die Rotation erhaltenen neuen Achsen gedreht, sondern um konstante (unveränderliche) Achsen.

A.8.3 Homogene Matrizen

Eine homogene Matrix $\mathbf{T}$ ist eine 4x4 Matrix (A.10), die Translation und Rotation kombiniert. Sie besteht aus einer 3x3 Rotationsmatrix $\boldsymbol{R}$ und einem 3x1 Translationsvektor $\mathbf{r}_t$.

$$\mathbf{T} = \left(\begin{array}{c|c} \mathbf{R} & \mathbf{r}_t \\ \hline \mathbf{0} & 1 \end{array}\right) \tag{A.10}$$

Die kompakte Darstellung erlaubt die Berechnung von Translation und Rotation in einem Rechenschritt. In der Robotik wird diese Kombination als Frame bezeichnet. Die letzte Zeile der Matrix $\mathbf{T}$ ist in der Robotik stets konstant, in der Computergrafik wird sie zur Skalierung und für perspektivische Transformationen genutzt. Die Inversion der homogenen Matrix ist, wegen den Eigenschaften der Rotationsmatrix, relativ einfach

zu berechnen (A.11).

$$\mathbf{T}^{-1} = \left(\begin{array}{c|c} \mathbf{R}^T & -\mathbf{R}^T\mathbf{r}_t \\ \hline \mathbf{0} & 1 \end{array}\right) \tag{A.11}$$

A.8.4 Quaternionen

Ein Quaternion $\mathbf{q}$ ist ein Quadrupel, bestehend aus einem Skalar q_w und einem Vektor $\mathbf{q}_v$. Sie stellen eine Erweiterung der komplexen Zahlen dar. Es ist wie folgt aufgebaut:

$$\begin{aligned} \boldsymbol{q} &= (\mathbf{q}_v, q_w) && \text{(A.12)} \\ &= (q_x, q_y, q_z, q_w) && \text{(A.13)} \\ &= iq_x + jq_y + kq_z + q_w && \text{(A.14)} \\ \boldsymbol{q}^* &= -iq_x - jq_y - kq_z + q_w && \text{(A.15)} \end{aligned}$$

Wobei $q_x, q_y, q_z, q_w \in \mathbb{R}$. Für i, j, k gelten die in (A.16) beschriebenen Zusammenhänge.

$$ij = k, ji = -k, jk = i, kj = -i, ki = j, ik = -j \tag{A.16}$$

Für ein Einheitsquaternion, d.h. falls gilt $\mathbf{q}^* = \mathbf{q}^{-1}$, lässt sich die Drehung um eine Achse $\mathbf{u}_q$ um den Winkel 2ϕ in der Form von Gleichung (A.17) darstellen.

$$\mathbf{q} = (\mathbf{u}_q \sin\phi, \cos\phi) \tag{A.17}$$

Literaturverzeichnis

[1] L. Petzold. A description of DASSL: A differential/algebraic system solver. *IMACS Trans. Scientific Computing Vol. 1, S. 65-68*, 1993.

[2] B. Gebler. Modellbildung, Steuerung und Regelung für elastische Industrieroboter. *VDI Verlag*, 1987.

[3] U. Kleemann. Regelung elastischer Roboter. *VDI Verlag*, 1989.

[4] W. Höbarth, H. Gattringer and H. Bremer. Modeling and Control of an Articulated Robot with Flexible Links/Joints. *International Conference on Motion and Vibration Control*, 2008.

[5] H. Bremer. On the Use of Nonholonomic Variables in Robotics, in: Selected Topics in Structronics and Mechatronic Systems. *World Scientifc*, 2003.

[6] H. Gattringer. Realisierung, Modellbildung und Regelung einer zweibeinigen Laufmaschine. *Dissertation, Universität Linz*, 2006.

[7] R. Mitterhuber. Modellierung und Regelung kooperierender Knickarmroboter mit elastischen Komponenten. *Dissertation, Universität Linz*, 2005.

[8] J. Haug. Zur Modellierung aktiv geregelter elastischer Mehrkörpersysteme. *VDI Verlag*, 1996.

[9] M. Hermle. Hierarchische Regelung globaler Bewegungen elastsicher Roboter. *VDI Verlag*, 2001.

[10] M. Thümmel. Modellbasierte Regelung mit nichtlinearen inversen Systemen und Beobachtern zur Optimierung der Dynamik von Robotern mit elastischen Gelenken. *Dissertation Uni München*, 2006.

[11] S. Moberg. On Modeling and Control of Flexible Manipulators. *Thesis No.1136 : Linköping studies in science and technology*, 2007.

[12] E. Wernholt. Multivariable Frequency-Domain Identification of Industrial Robots. *Dissertation No.1138 : Linköping studies in science and technology*, 2007.

[13] Modelica Association. Modelica - A Unified Object-Oriented Language for Physical Systems Modeling. 2004.

[14] M. Spong. Robot Dynamics and Control. *Wiley*, 1989.

[15] A. Schaefer. Regelung von Robotern mit elastischen Gelenken am Beispiel der DLR-Leichtbauarme. *Dissertation Uni München*, 2002.

[16] Dynasim AB. Dymola, Dynamic Modeling Laboratory. *User's Manual*, 2006.

[17] M. Otter. Objektorientierte Modellierung mechatronischer Systeme am Beispiel geregelter Roboter. *Dissertation, Ruhr-Universität Bochum*, 1995.

[18] F. Jurisch. Nutrastmomente in elektrischen Maschinen: Neue Betrachtungsweise und Maßnahmen zur gezielten Beeinflussung. *Vacuumschmelze GmbH, DM-PM 4*, 2003.

[19] D. Hanselman. Brushless Permanent Magnet Motor Design,Second Edition. *Magna Physics Pub*, 2006.

[20] H. Olsson. Control Systems with Friction. *Dept. of Automatic Control, Lund Inst. of Technology*, 1996.

[21] R. Koeppe und G. Hirzinger. From Human Arms to a New Generation of Manipulators: Control and design principles. *ASME Int. Mechanical Engineering Congress*, 2001.

[22] H. Taghirad,P. Belanger und A. Helmy. An Experimental Study on Harmonic Drives. *Technischer Bericht, McGill University*, 1996.

[23] C.Perlchen, C. Schweiger und M. Otter. Modeling and Simulating the Efficiency of Gearboxes and of Planetary Gearboxes. *Second International Modelica Conference, Proceedings*, 2002.

[24] M. Kurze. Modellbasierte Regelung von Robotern mit elastischen Gelenken ohne abtriebsseitige Sensorik. *Dissertation: Universität München*, 2008.

[25] G. Niemann, H. Winter. Maschinenelemente Band II und III. *Springer*, 1989.

[26] J. Volmer. Leitfaden Getriebetechnik, 3. Auflage. *Vieweg*, 1989.

[27] F. Gandhi, I. Chopra. A time-domain non-linear viscoelastic damper model. *Smart Mater. Struct. 5*, 1996.

[28] K. Kuhen. Inverse Steuerung piezoelektirsicher Aktoren mit Hysterese-, Kriech- und Superpostionsoperatoren. *Dissertation Uni Saarbrücken*, 2001.

[29] D. Pesotski, H. Janocha und P. Pagliarulo. Echtzeitfähiger Hysteresekompensator für Festkörperaktoren. *Automatisierungstechnik*, **56**, 2008.

[30] H. Lee, T. Royston und G. Friedman. Modeling and Compensation of Hysteresis in Piezoceramic Transducers for Vibration Control. *Journal of intelligent material systems and structres, Nr. 11*, 2000.

[31] O. Föllinger. Regelungstechnik. *Huethig Verlag Heidelberg*, 1994.

[32] H. Bremer, F. Pfeiffer. Elastische Mehrkörpersysteme. *Teubner Studienbücher*, 1992.

[33] M. Riemer, J. Wauer, W. Wedig. Mathematische Methoden der Technischen Mechanik. *Skriptum, Universität Karlsruhe*, 2002.

[34] A. Heckmann, M. Otter, S. Dietz, J. López. The DLR Flexible Bodies library to model large motions of beams and of flexible bodies exported from finite element programs. *The Modelica Association*, 2006.

[35] J. Rauh. Ein Beitrag zur Modellierung elastischer Balkensysteme. *VDI Verlag*, 1987.

[36] O. Wallrapp. Standardization of Flexible Body Modeling in Multibody System Codes, Part 1: Definition of Standard Input Data. *Mechanics of Structures and Machines 22(3):283-304*, 1994.

[37] C. Schweiger, M. Otter. Modeling 3D Mechanical Effects of 1D Powertrains. *Proceedings of the third International Modelica Conference*, 2003.

[38] F. Pfeiffer. Einführung in die Dynamik. *Teubner Studienbücher*, 1989.

[39] M. Otter, H. Elmqvist und S. Mattsson. The New Modelica MultiBody Library. *Proceedings of the 3rd International Modelica Conference*, 2003.

[40] L. Brita, I. Mufti. Some Results on an Inverse Problem in Multivariable Systems. *IEEE Transactions on Automation Control 12*, 1967.

[41] J. Buchholz, v. Grünhagen. Inversion Impossible? *IB 111-2003/34, DLR*, 2003.

[42] A. Isidori. Nonlinear Control Systems. *Springer*, 1995.

[43] J. Polzer. Differentialalgebraischer Nulldynamikalgorithmus für Systeme mit rationalen Funktionen. *Forschungsbericht 02/99. MSRT. Universität Duisburg*, 1999.

[44] A. Isidori. Nonlinear Control Systems II. *Springer*, 1999.

[45] M. Otter, H. Elmqvist, S. Mattsson und C. Schlegel. Objektorientierte Modellierung Physikalischer Systeme (Teil 1-9). *Automatisierungstechnik 48, R. Oldenbourg Verlag*, 2000.

[46] M. Thümmel, G. Looye, M. Kurze, M. Otter, J. Bals. Nonlinear Inverse Models for Control. *Proceedings of the 4th International Modelica Conference,Hamburg, March 7-8*, 2005.

[47] A. Pfeifer. Numerische Sensitivitätsanalyse unstetiger multidisziplinärer Modelle mit Anwendungen in der gradientenbasierten Optimierung. *Fortschritt-Berichte VDI, Reihe 20, Nr. 417, VDI-Verlag*, 2008.

[48] K. Brenan, S. Campell, L. Petzold. Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations. *Elsevier Science Publishing*, 1989.

[49] C. Pantelides. The consistent initialization of differential-algebraic systems. *SIAM Journal of Scientific and Statistical Computing*, 1988.

[50] S. Mattsson, G. Söderlind. Index reduction in differential-algebraic equations using dummy derivatives. *SIAM Journal of Scientific and Statistical Computing*, 1993.

[51] H. Elmqvist und M. Otter. Methods for Tearing Systems of Equations in Object-Oriented Modeling. *Proceedings of the European Simulation Multiconference (S. 326-332)*, 1994.

[52] R. Tarjan. Depth First Search and Linear Graph Algorithms. *SIAM Journal of Computing, Nr. 1, S.146-160*, 1972.

[53] A. Preumont. Vibration Control of Active Structures - An Introduction 2nd Edition. *Kluwer Academic journals*, 2002.

[54] M. Bachmayer, J. Rudolph, H. Ulbrich. Acceleration of linearly actuated elastic robots avoiding residual vibrations. *International Conference on Motion and Vibration Control*, 2008.

[55] M. Bachmayer, J. Rudolph, H. Ulbrich. Flatness based feed forward control for a horizontally moving beam with a point mass. *European Conference on Structural Control*, 2008.

[56] C. Kravaris, M. Niemiec, R. Berber und C. Brosilow. Nonlinear Model-based Control of Nominimum-phase Processes. *Nonlinear Model Based Process Control*, 1998.

[57] M. Niemiec, C. Kravaris. Nonlinear model-state feedback control for nonminimum-phase processes. *Automatica 39 (S. 1295 - 1302)*, 2002.

[58] J. Kanter, W. Seider. Real-Time, Nonlinear Control of a Constrained, Nonminimum-Phase Process. *AIChE-Journal Vol. 48, No. 10*, 2002.

[59] F. Pfeiffer, E. Reithmeier. Roboterdynamik. *Teubner*, 1987.

[60] A. Goldenberg, B. Benhabib und R. Fenton. A complete generalized solution to the inverse kinematics of robots. *IEEE Journal of Robots and Automation (Vol. 1, No. 1)*, 1985.

[61] H. Bremer. Dynamik und Regelung mechanischer Systeme. *Teubner Studienbücher*, 1988.

[62] W. Smidt. Verallgemeinerte inverse Kinematik für Anwendungen in der Robotersimulation und der virtuellen Realität. *Diplomarbeit Uni Dortmund*, 1998.

[63] C. Wampler. Manipulator inverse kinematic solutions based on vector formulations and damped least squares methods. *IEEE Transactions on Systems, Man, and Cybernetics, 16*, 1986.

[64] Y. Nakamura und H. Hanafusa. Inverse kinematics solutions with singularity robustness for robot manipulator control. *Journal of Dynamic Systems, Measurement, and Control, 108*, 1986.

[65] G. Golub, C. Van Loan. Numerical Linear Algebra for Applications in Statistics. *Springer-Verlag*, 1998.

[66] G. Golub, C. Van Loan. Matrix Computations, 3rd ed. *Springer-Verlag*, 1996.

[67] J. Nash. Compact Numerical Methods for Computers: Linear Algebra and Function Minimisation, 2nd ed. *Adam Hilger*, 1990.

[68] S. Buss, J. Kim. Selectively Damped Least Squares for Inverse Kinematics. *Online Publication from the Department of Mathematics,University of California, San Diego*, 2004.

[69] M. Kircanski. Symbolic Singular Value Decomposition for Simple Redundant Manipulators and Its Application to Robot Control. *The International Journal of Robotics Research 14*, 1995.

[70] S. Buss. Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods. *Online Publication from the Department of Mathematics,University of California, San Diego*, 2004.

[71] R. Schwertassek, O. Wallrapp. Dynamik flexibler Mehrkoerpersysteme. *Vieweg*, 1999.

[72] SIMPACK. http://www.simpack.com.

[73] A. Emami-Naeini, A. und P. Van Dooren. Computation of Zeros of Linear Multivariable Systems. *Automatica 18, S. 415-430*, 1982.

[74] Herman Jan Van de Straete. Physical Meaning of Zeros and Transmission Zeros from Bond Graph Models. *Massachsetts Institute of Technology*, 1995.

[75] A. Isidori, C. Moog. On the Nonlinear Equivalent of the Notion of Transmission Zeros. *IIASA Workshop on Modelling and Adaptive Control*, 1986.

[76] M. Reiner, A. Heckmann, M. Otter. Inversion based control of non minimum phase flexible bodies. *International Conference on Motion and Vibration Control*, 2008.

[77] R. Höpler, R. Thümmel. Symbolic Computation of the Inverse Dynamics of Elastic Joint Robots. *Proceedings of the 2004 IEEE International Conference on Robotics & Automation*, 2004.

[78] John G., Proakis und Dimitris G. Manolakis Digital Signal Processing Principals, Algorithms and Applications third edition. *Prentice Hall*, 1996.

[79] D. Fannin, W. Tranter und R. Ziemer. Signals & Systems Continuous and Discrete, fourth edition. *Prentice Hall*, 1998.

[80] M. Antonio, A. Pérez. Über die Eigenschaften des Modells eines elastischen Roboters. *Forschungsbericht Nr. 17/95, Meß-, Steuer und Regelungstechnik*, 1995.

[81] A. Lyapunov. Stability of motion. *Academic Press, New-York and London*, 1966.

[82] J.J. Slotine und W. Li. Applied Nonlinear Control. *Prentice Hall, Upper Saddle River, NJ*, 1991.

[83] H. Khalil. Nonlinear Systems. *Macmillian Publishing Company, New York*, 1992.

[84] I. Bronstein, K. Semendjajew, G. Musiol und H. Mühlig. Taschenbuch der Mathematik. *Verlag Harri Deutsch*, 2001.

[85] S. Gerschgorin. Über die Abgrenzung der Eigenwerte einer Matrix. *Izv. Akad. Nauk. UdSSR Otd. Fiz.-Mat. Nauk 7*, 1931.

[86] R. Varga. Gerschgorin and his circles. *Springer-Verlag*, 2004.

[87] E.D. Sontag. Input to state stability: Basic concepts and results. *Nonlinear and Optimal Control Theory,Springer-Verlag*, 2007.

[88] R. Johansson, A. Robertsson, K. Nilsson und M. Verhaegen. State-space system identification of robot manipulator dynamics. *Mechatronics*, 2000.

[89] R. Jansen. Learning an accurate neural model of the dynamics of a typical industrial robot. *International Conference on Artifcial Neural Networks*, 1994.

[90] H. Schuette,W. Moritz, R. Neumann und G. Wittler. Practical realization of Mechatronics in Robotics. *Preprints of the Third Int. Symposium on experimental Robotics*, 1993.

[91] R. Isermann. Parameter-Identifikationsverfahren. *TU Berlin, Automatisierung, Analyse und Synthese dynamischer Systeme, Nr. 42*, 1972.

[92] R. Isermann. Identifikation dynamischer Systeme: Grundlegende Methoden. *Springer-Verlag*, 1992.

[93] R. Isermann. Identifikation dynamischer Systeme: Besondere Methoden, Anwendungen. *Springer-Verlag*, 1992.

[94] P. Eykhoff. System Identification. *John Wiley and Sons*, 1974.

[95] A. Oppenheim und R. Schafer. Discrete-Time Signal Processing. *Prentice-Hall*, 1989.

[96] F. Gustafsson. Determining the initial states in forward-backward filtering. *IEEE Transactions on Signal Processing Volume 44, Issue 4*, 1996.

[97] S. Orfanidis. Optimum Signal Processing. An Introduction. 2nd Edition. *Prentice-Hall*, 1996.

[98] V. Torczon und M. Trosset. From evolutionary operation to parallel direct search: pattern search algorithms for numerical optimization, 1998.

[99] K. Schnepper. Parallel MATLAB Toolbox, Version 1.0. *DLR internal report*, 2005.

[100] L. Schmitt. Gerschgorin and His Circles. *Theoretical Computer Science (259)*, S. 1–61, 2001.

[101] D. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning. *Kluwer Academic Publishers*, 1989.

[102] D. Goldberg. The Design of Innovation: Lessons from and for Competent Genetic Algorithms. *Addison-Wesley*, 2002.

[103] G. Syswerda. Uniform crossover in genetic algorithms. *J. D. Schaffer Proceedings of the Third International Conference on Genetic Algorithms*, 1989.

[104] K. Schittkowski. Nonlinear Programming Codes. *Springer*, 1980.

[105] G. Kreisselmeier. Struktur mit zwei Freiheitsgraden. *Automatisierungstechnik 47*, 1999.

[106] V. Hagenmeyer, M. Zeitz. Flachheitsbasierter Entwurf von linearenund nichtlinearen Vorsteuerungen. *Automatisierungstechnik 52*, 2004.

[107] S. Bhattacharyya, H. Chapellat, und L. Keel. Robust Control The Parametric Approach. *Prentice Hall PTR*, 1995.

[108] N. Bajcinca,T.Bünte. A novel control structure for dynamic inversion and tracking tasks. *Proceedings IFAC*, 2005.

[109] F. Allgöwer, A. Zheng. Nonlinear model predictive control. *Progress in Systems Theory 26*, 2000.

[110] T. Umeno, Y. Hori. Robust speed control of DC servomotors using modern two degrees-of-freedom controller design. *IEE Trans. Ind. Electron. 38*, 1991.

[111] E. Bobrow, S. Dubowsky und J. Gibson. Time-Optimal Control of Robotic Manipulators Along Specified Paths. *International Journal of Robotics Research, Vol. 4, No. 3*, 1985.

[112] F. Pfeiffer, R. Johanni. A Concept for Manipulator Trajectory Planning. *IEEE Journal of Robotics and Automation, Vol. RA-3, No. 2*, 1987.

[113] Optimale Bahnplanung bei Industrierobotern. A Concept for Manipulator Trajectory Planning. *R. Johanni, Fortschritt-Berichte VDI, Reihe 18, Nr. 51, VDI Verlag*, 1988.

[114] H. Joos. MOPS - Multi-Objective Parameter Synthesis . *User Guide - Version 5.01*, 2006.

[115] H. Joos, J. Bals, G. Looye, K. Schnepper, A. Varga. A multi-objective optimisation based software environment for control systems design. *Proc. of 2002 IEEE International Conference on Control Applications and International Symposium on Computer Aided Control Systems Design, CCA/CACSD*, 2002.

[116] J. Maciejowski. Predictive control with constraints. *Prentice-Hall, Pearson Education Limited*, 2002.

[117] H. Elmqvist, M. Otter, F. Cellier. Inline Integration: A New Mixed Symbolic/ Numeric Approach for Solving Differential-Algebraic Equation Systems. *Proceedings of ESM, SCS European Simulation MultiConference*, 1995.

[118] KR C2 / KR C3 Programmierung Experte. *KUKA System Software (KSS), V. 5.2*, 2003.