

Diplomarbeit

# **Räumliche Kraftregelung an einem Hexapod auf Basis strukturintegrierter Kraftsensorik**

Frederik Schlüter

Geboren am: 29. Juni 1995 in Troisdorf  
Matrikelnummer: 4544931  
Immatrikulationsjahr: 2015

zur Erlangung des akademischen Grades

**Diplom-Ingenieur (Dipl.-Ing.)**

Betreuer

Dr.-Ing. Bernd Kauschinger  
Dipl.-Ing. Christian Friedrich  
Dipl.-Ing. Olaf Holowenko

Betreuer Hochschullehrer

Prof. Dr.-Ing. Steffen Ihlenfeldt

Eingereicht am: 16. März 2021

## Diplomarbeit Nr.: 1518

für Schlüter, Frederik

Matrikelnummer: 4544931

Studiengang: Maschinenbau | Studienrichtung: VTMB

**Thema: Räumliche Kraftregelung an einem Hexapod auf Basis strukturintegrierter Kraftsensorik**

Topic: Spatial force control of a hexapod machine tool based on structure-integrated force sensors

Im Rahmen vorangegangener Arbeiten wurden eine einfache Kraftregelung am Hexapod auf Basis von Lageregleroffsets über das HL-Interface umgesetzt und die dazu notwendigen Hexapod- und Kraftmessmodelle validiert. Defizite dieser Lösung sind die nur geringe Dynamik aufgrund der erforderlichen Bahnplanung sowie das Fehlen einer Möglichkeit zur synchronen Kraftsollwertvorgabe.

In dieser Arbeit sollen die Kraftregelung durch direkten Eingriff in die Lageregler und die Kraftsollwertvorgabe direkt im NC-Programm realisiert werden. Vorarbeiten dazu wurden im Rahmen eines Beleges anhand einer Zwei-Achs-Kinematik geschaffen. Die Umsetzung am Hexapod beinhaltet die Integration der 6D-Transformationen von Hexapod, Kraftmessmodell und Aufgabenkoordinatensystem, die Integration in den Steuerungskern sowie vor allem die Umsetzung eines praxisrelevanten Beispielprozesses.

Die Schwerpunkte sind im Einzelnen:

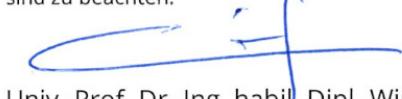
- Implementierung einer hybriden/parallelen räumlichen Kraftregelung für den Hexapod Felix I als C++ Modul in TwinCAT durch Ergänzung der Einzelgelenkregler
- Realisierung von Kraft- und Wegsollwertvorgabe und Zustandssteuerung des Reglers durch G-Code-Variablen und -Befehle
- Entwicklung und Bau eines praxisrelevanten Demonstrationsexperiments zur Validierung der Kraftregelung, bspw. einer Konturverfolgung
- Vergleich verschiedener Varianten strukturintegrierter Kraftsensorik für ihre Eignung zur räumlichen Kraftregelung
- Vergleich der Reglerperformance zur vorhandenen Lösung

Betreuer: Dr.-Ing. Bernd Kauschinger, Dipl.-Ing. Christian Friedrich

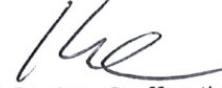
Ausgegeben am: 01.10.2020

Einzureichen am: 28.02.2021

Die Diplomprüfungsordnung der Fakultät Maschinenwesen sowie die von der Professur für Werkzeugmaschinenentwicklung und adaptive Steuerungen erlassenen Hinweise zur Anfertigung von Studien- und Diplomarbeiten sind zu beachten.



Univ.-Prof. Dr.-Ing. habil. Dipl.-Wirt.-Ing. Chokri Cherif  
Studienrichtungsleiter  
Verarbeitungsmaschinen- und Textilmaschinenbau

  
Prof. Dr.-Ing. Steffen Ihlenfeldt  
Betreuer Hochschullehrer

# Kurzfassung

Nur wenige kommerzielle Werkzeugmaschinensteuerungen sehen die Verwendung einer Kraftregelung vor, obwohl daraus im Bereich der Fertigungstechnik zahlreiche Vorteile resultieren. Deswegen verfolgt diese Diplomarbeit das Ziel, auf der Steuerungssoftwarelösung *TwinCAT* die Kraftregelung einer Hexapodwerkzeugmaschine mit strukturintegrierter Kraftmessung zu realisieren.

Dafür wird die Infrastruktur *TwinCATs* zunächst für die Kraftregelung nutzbar gemacht. Das Standard-CNC-Modul wird zur Interpolation und Überwachung von Kraft- und Positions倅erten verwendet. Die Implementierung der Kraftregelung erfolgt in C++ unter Anwendung eines modularen Softwaredesigns. Um eine sichere und flexible Entwicklung der Regelung zu gewährleisten, erfolgt parallel dazu die Implementierung eines virtuellen Prozess- und Steuerungsmodells, das konsequentes Unittesting erlaubt. Die kartesische Kraftregelung ist lagebasiert und verfolgt einen parallel/hybriden Ansatz. Die Verwendung einer Zustandsmaschine sichtert eine transparente und sichere Bedienung. Nach Abschluss der Softwareumsetzung wird die Inbetriebnahme zuerst im virtuellen Modell, dann an der Maschine vorgenommen. Zur Validierung der Leistungsfähigkeit schließt sich die Planung und Umsetzung verschiedener Experimente zur Konturverfolgung mit steigender Komplexität an, anhand derer praxisrelevante Aspekte der Regelung evaluiert werden.

Es kann dabei gezeigt werden, dass die Kraftregelung in *TwinCAT* funktioniert. Durch die Aufgabendefinition im G-Code und die Verwendung der CNC können Interpolation der Kraft- und Positions倅erte synchron stattfinden. Die Dynamik der Regelung erhöht sich durch die gewählte Regelstrategie im Vergleich zur Vorgängerarbeit um das 30-fache. Die Experimente validieren zudem die Tauglichkeit der strukturintegrierten Kraftmessung zur Kraftregelung. Es zeigt sich jedoch, dass das System unter Kraftregelung schwingungsanfällig ist.

## Abstract

Only few commercial machine tool control system softwares provide the usage of force control despite the benefits resulting from its usage, especially in production technology. This diploma thesis therefore aims to implement a force control based on structure integrated measurement for a hexapod machine tool using the control software system *TwinCAT*.

Initially, steps are undertaken to harness the infrastructure of *TwinCAT* for force control use. The default CNC module is utilized for interpolation and monitoring of force and position data. The force controller is written in C++, following a modular design concept. To ensure a safe and flexible development of the force controller, a virtual process and control system model is implemented parallelly to allow for consistent unit testing. The cartesian position based force controller is adopting a parallel/hybrid approach. The application of a state machine ensures

---

a transparent and safe operation. Once finalized, the software is put into use virtually against the simulation before getting deployed on the machine. For validating the force controller performance, a set of experiments with increasing complexity is designed and conducted. They serve as a demonstration of praxis relevant aspects of machining processes for evaluating the force control developed.

This thesis shows that an implementation of a force controller inside *TwinCAT* is possible. Enabling the task definition via G-Code and the usage of a CNC allows for interpolating force and position data synchronously. The dynamic can be increased by a factor of 30 compared to the preceding solution. Furthermore, the experiments underline the aptitude of structure integrated force measurement for force control purposes. However it became apparent that the hexapod tends to be susceptible to vibrations.



# Inhaltsverzeichnis

<b>1. Einleitung</b>	<b>1</b>
1.1. Motivation . . . . .	1
1.2. Aufgabenstellung . . . . .	3
<b>2. Stand der Technik</b>	<b>4</b>
2.1. Kinematik von Werkzeugmaschinen . . . . .	4
2.1.1. Position und Orientierung im Raum . . . . .	4
2.1.2. Parallelkinematische Werkzeugmaschinen . . . . .	7
2.2. Steuerung und Regelung von WZM . . . . .	10
2.2.1. CNC . . . . .	10
2.2.2. SPS . . . . .	11
2.2.3. Kraftregelung . . . . .	11
2.3. Beschreibung des Versuchsträgers . . . . .	19
2.3.1. Hardware . . . . .	19
2.3.2. Software . . . . .	22
<b>3. Anforderungen</b>	<b>27</b>
3.1. Ziele . . . . .	27
3.2. Allgemeine Beschreibung . . . . .	28
3.3. Spezifische Anforderungen . . . . .	29
<b>4. Bedienkonzept</b>	<b>32</b>
4.1. Sollwertvorgabe . . . . .	32
4.2. Benutzerdefinierte Koordinatensysteme . . . . .	33
<b>5. Steuerungssoftware</b>	<b>35</b>
5.1. Anpassung der CNC . . . . .	37
5.2. C++-Transformationsmodul TcNcTrafoExtCnc . . . . .	38
5.3. SPS . . . . .	39
5.4. C++-Modul ForceManager . . . . .	41
5.4.1. C++-Submodul MCommandDispatcher . . . . .	45

5.4.2. C++-Submodul StateMaschine . . . . .	45
5.4.3. C++-Submodul DriveHandler . . . . .	50
5.4.4. C++-Submodul ForceController . . . . .	53
5.5. Fazit . . . . .	56
<b>6. Inbetriebnahme</b>	<b>57</b>
6.1. Virtuelle Inbetriebnahme . . . . .	57
6.2. Reale Inbetriebnahme . . . . .	62
<b>7. Experimentelle Validierung</b>	<b>69</b>
7.1. Kraftregelung im kartesischen Raum . . . . .	69
7.2. Vergleich verschiedener Sensoren . . . . .	73
7.3. Kraftregelung in variablen Koordinatensystemen . . . . .	74
7.3.1. Kraftregelung in Radialrichtung . . . . .	75
7.3.2. Kraftregelung in Normalenrichtung . . . . .	79
<b>8. Zusammenfassung</b>	<b>82</b>
<b>Anhang</b>	
A. Standardwerte der Kanalparameter	85
<b>Literaturverzeichnis</b>	<b>88</b>
Formelzeichen und Abkürzungen . . . . .	92
Abbildungsverzeichnis . . . . .	97
Tabellenverzeichnis . . . . .	98

# 1. Einleitung

## 1.1. Motivation

Die Automatisierung in der Fertigungstechnik ist ein wichtiger Bestandteil der industriellen Produktion. Dabei spielt die Regelung von Werkzeugmaschinen eine wesentliche Rolle. Stand der Technik ist, dass die Regelung im Normalfall ausschließlich die Position des Werkzeugs berücksichtigt. Die wenigsten Maschinen betrachten auftretende Prozesskräfte, obwohl diese in der Fertigungstechnik neben Position und kinematischen Größen die wichtigsten Einflüsse für viele Fertigungsprozesse darstellen. Die Prozesskräfte werden momentan in der Praxis meist nicht erfasst [1] und noch seltener geregelt. Wenn die relevanten Kräfte richtig eingestellt sind, laufen die Prozesse in der Regel stabiler, die Werkzeuge verschleißt nicht so schnell und die Werkstücke weisen eine höhere Qualität auf [2]. Durch die Verwendung einer Kraftregelung lassen sich diese Kräfte wesentlich leichter einstellen. Außerdem kann eine mit Kraftsensoren ausgestattete Maschine bei unerwarteten Kollisionen einen Nothalt auslösen und somit die Mensch-Maschine-Interaktion um einiges sicherer machen. Zusätzlich zu den erwähnten Verbesserungen ist es mit einer Kraftregelung möglich, neue Funktionen umzusetzen. Ein Beispiel dafür ist das Turbinenschaufelschleifen, ein zurzeit manuell durchgeführter Prozess, der mit einer Kraftregelung automatisiert werden kann [3]. Obwohl dieses Regelungsthema die Forschung schon seit mehreren Jahrzehnten beschäftigt, konnte sich die Kraftregelung trotz all dieser Vorteile bisher nicht großflächig durchsetzen. Das hängt wesentlich damit zusammen, dass

1. die empfindliche Kraftsensorik teuer ist, den Arbeitsraum verkleinert und die Steifigkeit der Maschine herabsetzt,
2. die Funktionalität zur Kraftregelung in den meisten kommerziellen Steuerungssystemanwendungen nicht vorgesehen ist [4] und
3. Kraftregelungen in der Aufgabendefinition wesentlich komplexer sind als Positionsregelungen.

In den vergangenen Jahren hat sich das Institut für Mechatronischen Maschinenbau (IMD) der TU Dresden intensiv mit diesen Problematiken auseinandergesetzt.

Zur Kraftmessung sind 6D-Kraftmessdosen weit verbreitet, die am Endeffektor angebracht sind. Um die damit verbundenen und oben geschilderten Nachteile zu entkräften, wurden im DFG-finanzierten Projekt "Grundlagen zur strukturintegrierten Messung und steuerungsintegrierten Verarbeitung räumlicher Kräfte und Momente in Fertigungseinrichtungen" an verschiedenen Stellen im Versuchshexapod *Felix* Kraftmesssensoren in die kinematischen Ketten und die Plattform integriert [5]. Mit einem kinetischen Modell der Maschine können auf diese Weise am Werkzeug angreifende Kräfte mit zur 6D-Kraftmessdose vergleichbarer Genauigkeit gemessen werden [6].

Um die Kraftregelung in einer kommerziellen Steuerung verfügbar zu machen, ist in den letzten Jahren an der TU Dresden viel mit offenen Steuerungssystemen experimentiert worden. Dadurch soll dem Benutzer, trotz der neuen Eigenschaften, die gewohnte Steuerungsumgebung erhalten bleiben. So kann die CNC zur gleichzeitigen Kraftwertinterpolation und Bahnplanung eingesetzt werden. Außerdem gewährleisten Steuerungssysteme die Einhaltung der Echtzeitbedingung. Dabei hat sich für die Forschung an Kraftregelungen das kommerzielle Softwaresystem *TwinCAT* von Beckhoff bewährt. Das Aufschalten der Stellgrößen der positionsbasierten Kraftregelung auf die kommandierte Sollposition stellt den anspruchsvollsten Aspekt dar. Momentan werden für Kraftregelungsoffset- und Sollposition getrennte Bahnen geplant und über eine gesonderte SPS aufgeschalten [7]. Bei diesem Verfahren ergeben sich lange Totzeiten.

Diese Arbeit verfolgt das Ziel, die Totzeiten des Status Quos mit einem alternativen Regelkonzept zu verkürzen. Durch die verkürzte Totzeit wird eine verbesserte Leistungsfähigkeit der Kraftregelung erwartet. Außerdem ermöglicht der neue Ansatz eine gleichzeitige Vorgabe und Interpolation von Kraft- und Positionssollwerten. Zudem soll die strukturintegrierte Kraftmessung für die Verwendung in der Kraftregelung evaluiert werden. Damit kann diese Arbeit einen Beitrag zu den beiden Punkten Steuerungsintegration und Kraftsensorik leisten.

## 1.2. Aufgabenstellung

Ziel dieser Diplomarbeit ist es, bestehende Kraftregler und Kraftmessmethoden mit einem alternativen Regelkonzept in die Echtzeitsteuerung *TwinCAT* von Beckhoff zu integrieren. Dadurch soll eine räumliche Kraftregelung am Hexapod *Felix* I der TU Dresden ermöglicht werden. Zu Beginn der Arbeit findet die Positionsregelung des Hexapods im Gelenkraum statt. Weil die Kraftregelung im kartesischen Raum stattfinden soll und die Vorwärtstransformation für eine Hexapodkinematik rechenaufwändig ist, bietet es sich an, die Positionsregelung zuerst in den kartesischen Raum zu verlegen.

Der in dieser Arbeit entwickelte Kraftregler generiert Lageinkremente, welche auf die Sollposition aufgeschaltet werden sollen. Dafür wird der Informationsfluss zwischen CNC und Antrieben über ein C++-Modul umgeleitet, das nach dem Prinzip *Man-in-the-middle* die Sollwerte manipuliert, ohne das die CNC davon Kenntnis erlangt. Um beim Ein- und Ausschalten Positionssprünge zu vermeiden, ist es unabdingbar, einen Mechanismus zu entwerfen, der das Lageinkrement verwaltet. Dafür wird eine Zustandsmaschine entwickelt, die zudem eine komfortable und sichere Interaktion mit dem Benutzer ermöglicht und dem Benutzer den Status der Regelung nachvollziehbar darstellt. Sowohl die Zustandsmaschine, als auch die Kraftsollwertvorgabe soll per G-Code vom Benutzer gesteuert werden können. Nach Implementierung des Kraftreglers demonstrieren einige dafür konzipierte Experimente die Leistungsfähigkeit der neuen Kraftregelung. Zum Abschluss werden die verschiedenen zur Verfügung stehenden Kraftmessmethoden in einem Demonstrationsprozess miteinander verglichen. Im Detail besteht die Vorgehensweise darin

- eine geeignete Struktur zum Auslesen und Verarbeiten der Benutzereingaben, vor allem der Kraftwerte aus dem G-Code anzulegen,
- die Zustände der Kraftregelung durch eine geeignete Zustandsmaschine über den G-Code steuerbar zu machen,
- Kraftsoll- und Istwerte in einen geeigneten Aufgabenraum umzurechnen und
- die Wegoffsets aus der Kraftregelung durch einen geeigneten Mechanismus auf die Werte der Positionsregelung aufzuschalten

Die Programmierung findet mit der Hochsprache C++ statt. Das Softwareprojekt soll als *TwinCAT*-Modul ausgeführt werden.

# 2. Stand der Technik

## Überblick

Dieses Kapitel gliedert sich in zwei Teile. Im ersten Abschnitt werden allgemeine, zum Verständnis der Thematik nötige Grundlagen erläutert. Das umfasst sowohl die Theorie, als auch die Beschreibung der Technologien, die in der Werkzeugmaschinensteuerung Anwendung finden. Der zweite Abschnitt enthält eine spezifische Beschreibung des im Rahmen dieser Arbeit genutzten Versuchsträgers *Felix I*. Dabei werden der genaue Aufbau, sowie einzelne Bestandteile erläutert. Es folgt eine Beschreibung der Ausgangssituation und ein Überblick über die Vorgängerarbeiten am Hexapod *Felix I* der TU Dresden. Dem schließt sich eine Betrachtung weiterer Arbeiten anderer Forschungseinrichtungen zur vorliegenden Problematik an.

## 2.1. Kinematik von Werkzeugmaschinen

### 2.1.1. Position und Orientierung im Raum

Grundlegende Voraussetzung für die Positionsregelung bzw. für die lagebasierte Kraftregelung ist die Beschreibung der Pose des charakteristischen Werkzeugpunktes, auch TCP (Tool-Center-Point) genannt. Unter Pose versteht man nach DIN EN ISO 8373 die Kombination aus Position und Orientierung im Raum. Im Folgenden wird erläutert, wie die Pose eines allgemeinen Objektes, im Speziellen hier des TCPs, mathematisch beschrieben werden kann. Zur Beschreibung eines beliebigen Objektes bietet es sich an, ein Koordinatensystem zu definieren, dessen Ursprung ein charakteristischer Punkt des Objektes ist. Zu dem beschriebenen Punkt wird ein fixer Bezug benötigt, der ebenfalls mit einem Koordinatensystem versehen ist. Im Rahmen dieser Arbeit finden ausschließlich kartesische Koordinatensysteme Verwendung. Das bedeutet, dass alle Koordinatensysteme außer des Ursprungs noch über drei orthogonale Achsen verfügen. Die Pose beschreibt dabei immer die Differenz des betrachteten Koordinatensystems  $\{M\}$  zum Bezugskoordinatensystem  $\{B\}$ . Im dreidimensionalen sind sechs Parameter notwendig, um die Pose darzustellen. Als Position bezeichnet man die Koordinaten des Ursprungs  $O$  von  $\{M\}$  im Koordinatensystem  $\{B\}$ . Die dreidimensionale Position lässt sich

folgendermaßen notieren:

$${}^B P_M = \begin{pmatrix} {}^B O_x \\ {}^B O_y \\ {}^B O_z \end{pmatrix} \quad (2.1)$$

Die Position bezeichnet gleichermaßen die Lage des Ursprungs im Bezugssystem und die notwendige Verschiebung der Basis, um zum betrachteten Punkt zu gelangen. Als Verschiebung wird eine Differenz zweier Koordinatensysteme bezeichnet, bei welcher alle Achsen der jeweiligen Systeme parallel zueinander bleiben [8].

Die Orientierung beschreibt, wie die Koordinatenachsen des betrachteten Systems gegenüber der Basis verdreht sind. Die Rotation ist das Pendant der Verschiebung und bezeichnet eine Differenz zwischen zwei Koordinatensystemen, bei welcher der Ursprung am gleichen Ort bleibt [8]. Während die Verschiebungen in den einzelnen Koordinatenrichtungen kommutativ sind, spielt die Reihenfolge der Rotationen um die verschiedenen Achsen eine Rolle. Im Zuge dessen hat sich eine Vielzahl an Darstellungsformen der Orientierung herausgebildet. Im Folgenden findet eine kurze Beschreibung der in dieser Arbeit verwendeten Methoden statt.

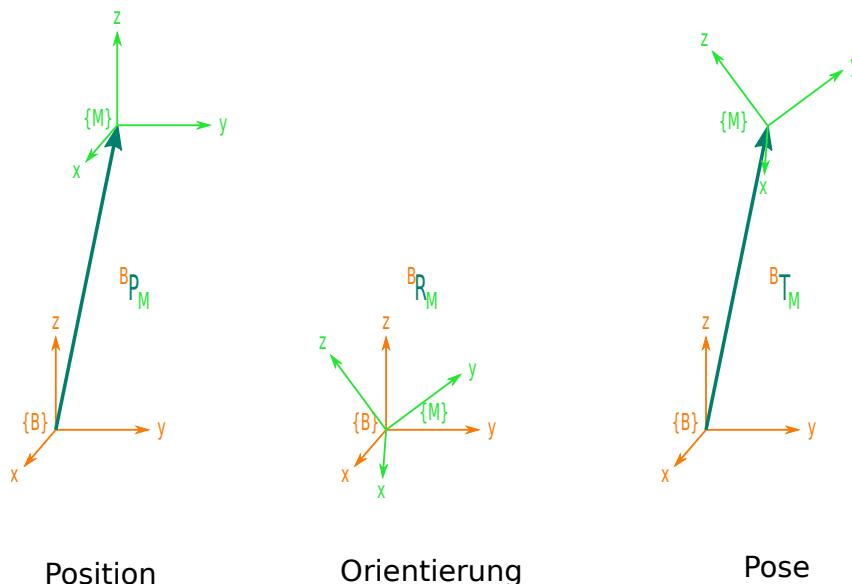


Abbildung 2.1.: Veranschaulichung 3D-Pose

Um eine Orientierung im dreidimensionalen kartesischen Raum zu beschreiben, benötigt man mindestens drei Parameter. Eulerwinkel drücken die Orientierung eines Körpers als drei Rotation um verschieden definierte Achsen aus. In der Literatur variieren die Bezeichnungen von Variante zu Variante. Wird um die nicht mitbewegten Bezugssystemachsen rotiert, wird das Verfahren in dieser Arbeit als *globale Eulerwinkel* bezeichnet. Es existieren mehrere gültige Varianten. Hier wird zuerst um die x-, dann die y- und schließlich um die z-Achse des Bezugssystems  $\{B\}$  rotiert. Erfolgt die Drehung konsekutiv, das heißt um bereits gedrehte Achsen des Systems  $\{M\}$ , spricht man von *lokalen Eulerwinkel* [9]. Eine dritte Variante, eine Abände-

rung der mitgedrehten Z-Y-Z-Rotation, ist in der Literatur als *modifizierte Eulerwinkel* zu finden [10]. Die Rotationen finden erst um die alte z-Achse statt, dann in der neuen y-z-Ebene und schließlich um die neue z-Achse. Ein Vorteil ist, dass die Winkel die Orientierung im Raum sehr anschaulich darstellen (siehe Abbildung 2.2). Die beiden letztgenannten, konsekutiven Verfahren bergen jedoch die Gefahr einer *kardanischen Blockade* (engl.: gimbal lock): Falls in der y-z-Ebene keine Rotation stattfindet, kommt es zum Verlust eines Freiheitsgrades.

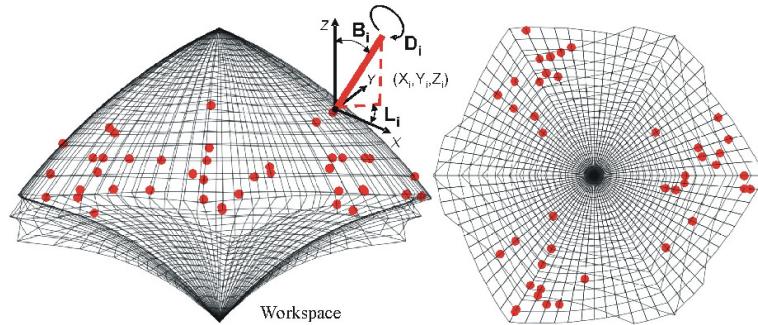


Abbildung 2.2.: modifizierte Eulerwinkel B,L & D [10].

Eine weitere Beschreibungsmöglichkeit liegt in der Rotationsmatrix (Gl. 2.2). Das Skalarprodukt der die beiden Koordinatensysteme festlegenden Einheitsvektoren ist eindeutig definiert, es kann keine kardanische Blockade auftreten, dafür sind sechs der neun Einträge redundant.

$${}^B R_M = \begin{pmatrix} \hat{x}_B \cdot \hat{x}_M & \hat{y}_B \cdot \hat{x}_M & \hat{z}_B \cdot \hat{x}_M \\ \hat{x}_B \cdot \hat{y}_M & \hat{y}_B \cdot \hat{y}_M & \hat{z}_B \cdot \hat{y}_M \\ \hat{x}_B \cdot \hat{z}_M & \hat{y}_B \cdot \hat{z}_M & \hat{z}_B \cdot \hat{z}_M \end{pmatrix} \quad (2.2)$$

Die Rotationsmatrix  ${}^B R_M$  gibt die Basisvektoren des Koordinatensystems  $\{M\}$  in Basiskoordinaten des Systems  $\{B\}$  an. Die interne Berechnung von Posen findet in der Regel mit Rotationsmatrizen statt, während sich die Eulerwinkel bei der Benutzerinteraktion etabliert haben.

In der Realität ist es oft nötig, mehrere Umrechnungen hintereinander durchzuführen, um ein Objekt, das in einem verketteten Koordinatensystem definiert ist, im Basiskoordinatensystem auszudrücken. Dafür hat sich eine kompakte Darstellungsform entwickelt, die homogene Transformation. Damit kann eine kombinierte Verschiebung und Rotation als einfache Vektor-Matrix-Multiplikation ausgedrückt werden. Analog zu [9] ist die homogene Transformation, die ein in  $\{M\}$  beschriebenes Objekt  $P$  in  $\{B\}$  ausdrückt in Gl. 2.4 definiert:

$${}^B p = {}^B R_M \cdot {}^A p + {}^B o_M \quad (2.3)$$

$$\begin{pmatrix} {}^B p \\ 1 \end{pmatrix} = \underbrace{\left( \begin{array}{c|c} {}^B R_M & {}^B o_M \\ \hline 0^T & 1 \end{array} \right)}_{{}^B T_M} \cdot \begin{pmatrix} {}^A p \\ 1 \end{pmatrix} \quad (2.4)$$

Es lassen sich beliebig viele Transformationen aneinanderreihen, womit Posen einfach in beliebige Koordinatensysteme umgerechnet werden können:

$${}^0\mathbf{P}_n = {}^0\mathbf{T}_1 \cdot {}^1\mathbf{T}_2 \cdots {}^{n-1}\mathbf{T}_n \cdot {}^n\mathbf{P} \quad (2.5)$$

Die homogene Transformation ist immer eindeutig [11].

### 2.1.2. Parallelkinematische Werkzeugmaschinen

#### Einordnung

Die Aufgabe einer Werkzeugmaschine besteht darin, den TCP einer Vorgabe folgend präzise zu führen und das Werkstück durch den Kontakt mit dem ausgewählten Werkzeug wiederholgenau zu bearbeiten. Deswegen ist es wichtig, dass der TCP nicht nur die geforderte Zielposition erreicht, sondern dass die Zielposition auf einer vorgegebenen Bahn exakt angefahren wird. Durch die deswegen benötigte Bahnplanung und Positionierungsgenauigkeit grenzen sich Werkzeugmaschinen stark von der Robotik ab, wo eine einfache Punkt-zu-Punktbewegung in den meisten Fällen ausreicht. Die sich daraus ableitenden Anforderungen an die Steuerung werden in Abschnitt 2.2.1 geschildert. Der mechanisch belastete Teil einer Werkzeugmaschine muss für eine hohe Präzision möglichst steif sein. Das wird in konventionellen Werkzeugmaschinen durch massive Bauteile realisiert. Das Kollidiert mit der Forderung, bei geringer Aufstellfläche einen möglichst großen Arbeitsraum zur Verfügung zu stellen. Seit Jahren wird deswegen an alternativen Maschinenprinzipien geforscht. Traditionelle Werkzeugmaschinen sind seriell aufgebaut. Das bedeutet, dass die kinematische Kette von Gestell zu Endeffektor offen ist. Ein Beispiel für solche Maschinen sind die Konsolfräsmaschinen, bei denen die Antriebe so angeordnet sind, dass die Bewegung in Richtung jedes Freiheitsgrades von den anderen Richtungen entkoppelt ist (siehe Abb. 2.3).

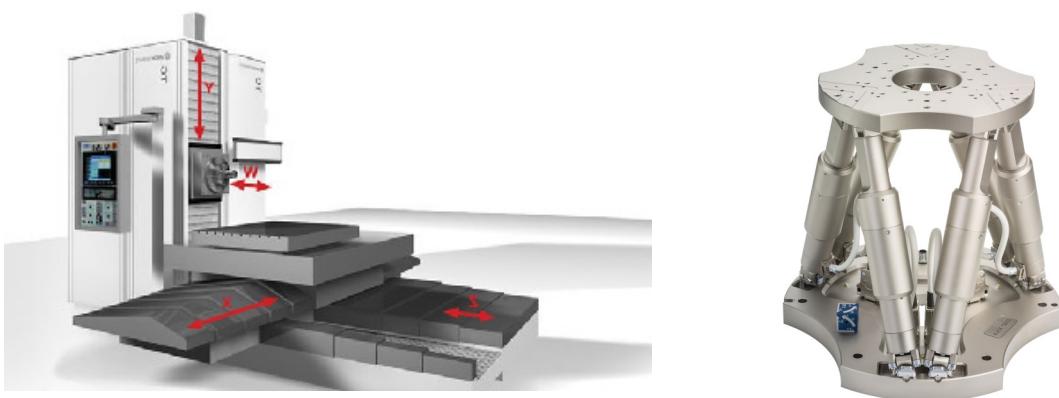


Abbildung 2.3.: Links: Entkoppelte, serielle Kinematik einer konventionellen Werkzeugmaschine, Freiheitsgrade in rot [12], rechts: Parallelkinematik Hexapod [13]

Im Gegensatz dazu stehen parallelkinematische Werkzeugmaschinen. Bei ihnen sind die kinematischen Ketten zwischen Gestell und TCP geschlossen (siehe auch Abb. 2.3). Dadurch ist

die Steifigkeit im Vergleich mit einer seriellen Maschine gleicher Masse höher [14]. Allerdings führt die geschlossene kinematische Kette zu stark miteinander verkoppelten Antrieben. Das bedeutet, dass für die Bewegung des Endeffektors in einer kartesischen Richtung immer mehrere Antriebe synchronisiert aktiv sein müssen. Infolge dessen ist die Regelung für die meisten Vertreter parallelkinematischer Maschinen deutlich komplexer, als bei klassischen Werkzeugmaschinen. Lange Zeit war eine Bahnplanung, ergo eine Verwendung als Werkzeugmaschine deswegen nicht möglich [15]. Die Rechentechnik ist allerdings so weit fortgeschritten, dass die komplexe Steuerung nur den Aufwand erhöht, jedoch kein Ausschlusskriterium mehr darstellt. Auf diese Problematik wird anhand der in dieser Arbeit verwendeten Kinematik in Abschnitt 2.3.1 eingegangen. Die Steifigkeit von parallelkinematischen Werkzeugmaschinen liegt in der Regel zwischen konventionellen Werkzeugmaschinen und Roboterarmen [16], wobei die Aufstellfläche kleiner ist als bei ersteren. Die Dynamik paralleler Kinematiken ist weitaus höher, weil die Antriebe alle im Gestell untergebracht werden können und deshalb nicht mitbewegt werden müssen. Dadurch und durch die generell geringere Masse liegen die Eigenfrequenzen höher, als bei konventionellen Werkzeugmaschinen. Die Ungenauigkeiten der einzelnen Achsen summieren sich im Gegensatz zu seriellen Kinematiken nicht auf, sondern gehen nur zu Anteilen in die Positioniergenauigkeit ein [17]. Durch die Verwendung mehrerer baugleicher kinematischer Teilketten sinkt der Fertigungs- und Wartungsaufwand. Ein Überblick über parallelkinematische Werkzeugmaschinen befindet sich in [18].

### Arbeitsraum und Gelenkraum

Bei verkoppelten Kinematiken unterscheiden sich die Koordinatensysteme der Antriebe grundsätzlich vom kartesischen Arbeitsraum des TCPs. Im sogenannten Gelenkraum ist die Pose des Endeffektors aus Winkeln oder Längen der Antriebe definiert. Um die Pose zwischen den beiden Räumen umzurechnen, sind mehrere Verfahren notwendig. Mit den kartesischen Werten der Pose  $\mathbf{x}$  lassen sich die Antriebswerte  $\mathbf{q}$  durch die inverse Kinematik, auch Rückwärtskinematik genannt, errechnen:

$$\mathbf{q} = IK(\mathbf{x}), \quad (2.6)$$

wobei  $\mathbf{x}$  die kartesischen Koordinaten und  $\mathbf{q}$  die Gelenkraumkoordinaten bezeichnet. Die Rückwärtskinematik ist für Parallelkinematiken eindeutig bestimmbar. Das Pendant dazu ist die direkte, oder Vorwärtskinematik, bei der aus den Gelenkwerten die kartesische Pose berechnet werden kann:

$$\mathbf{x} = DK(\mathbf{q}), \quad (2.7)$$

Während sie für serielle Kinematiken trivial ist, existieren für parallelkinematische Maschinen keine praxisrelevanten geschlossenen Lösungen [19]. Stattdessen kann die Lösung iterativ erlangt werden [8]. Im ersten Iterationsschritt wird  $\mathbf{x}_k$  geschätzt.

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{A} \cdot (\mathbf{q} - IK(\mathbf{x}_k)), \text{ für } \mathbf{A} \text{ häufig verwendet:} \quad (2.8)$$

$$\mathbf{A} = J_{kin}^{-1}(\mathbf{x}_k) \quad (2.9)$$

Dabei fallen je nach Kinematik aufwändige Berechnungen an, weil für jeden Iterationsschritt inverse Kinematik und Jacobimatrix  $J_{kin}$  notwendig sind. Die Jacobimatrix ist eine Matrix, mit Hilfe derer Geschwindigkeiten und kleine Lageänderungen vom Gelenkraum in den Arbeitsraum transformiert werden können:

$$\Delta \mathbf{x} = J_{kin}(\mathbf{x}) \cdot \Delta \mathbf{q} \quad (2.10)$$

$$\frac{d}{dt} \mathbf{x} = J_{kin}(\mathbf{x}) \cdot \frac{d}{dt} \mathbf{q}, \quad (2.11)$$

Sie ist vom Aufbau der Kinematik abhängig und nur im Umkreis einer Pose gültig. Damit die Iteration aus Gl. 2.8 schnell konvergiert, sollte  $\mathbf{x}_k$  daher in der Nähe der wirklichen Position liegen. Es wird so lange iteriert, bis  $\Delta \mathbf{q} = \mathbf{q} - I_K(\mathbf{x}_k)$  einen vorgegebenen Schwellenwert  $\varepsilon$  unterschreitet. Um das Gegenteil von Gl. 2.10, also aus Poseänderungen im Arbeitsraum die entsprechenden Winkeländerungen zu erhalten, muss die Jacobimatrix invertiert werden

$$\Delta \mathbf{q} = J_{kin}(\mathbf{x})^{-1} \cdot \Delta \mathbf{x}, \quad (2.12)$$

$$\frac{d}{dt} \mathbf{q} = J_{kin}(\mathbf{x})^{-1} \cdot \frac{d}{dt} \mathbf{x}, \quad (2.13)$$

was jedoch nicht immer möglich ist. Es gibt zu jeder verkoppelten Kinematik charakteristische Punkte, die Singularitäten genannt werden. Diese Punkte zeichnen sich dadurch aus, dass die Jacobimatrix Rangverlust erleidet. Sie ist bei Singularitäten und auch in der unmittelbaren Nachbarschaft nicht mehr invertierbar. Für Parallelkinematiken gibt es zwei Typen von Singularitäten. Bei Typ I verlieren die Kinematiken einen Freiheitgrad. Bei Typ II kommt ein Freiheitsgrad dazu. In beiden Fällen können die jeweiligen Kinematiken auch mit sehr hohen Antriebsmomenten nicht mehr vollständig kontrolliert werden.

## Anwendung von Hexapoden

Hexapoden, zu denen auch der verwendete Versuchsträger gehört, sind parallelkinematische Strukturen. Ursprünglich für einen Versuchsstand für Autoreifen entworfen, finden sie heute in vielen Bereichen Anwendung. Obwohl die Verbreitung noch weit hinter der Verwendung serieller Kinematiken zurücksteht, gibt es einige Werkzeugmaschinen, die auf Hexapoden basieren, zum Beispiel die Fräsmaschine *Mikromat 6X* [20]. Hexapoden sind jedoch in anderen Bereichen fest etabliert und deshalb gut erforscht. Durch die hohe Steifigkeit und die kompakte Größe werden sie in der Mess- und Positioniertechnik benutzt [21]. Das geschieht sowohl im Rahmen kleinstter Präzisionsbewegungen, als auch bei der exakten Positionierung schwerer Teleskope [22]. In der Logistik und Montage werden Hexapoden eingesetzt, dabei teilweise nicht fest montiert, sondern an Seilen aufgehängt, was einen mobilen Einsatz ermöglicht [23]. Zahlreiche Flugsimulatoren verwenden Hexapodplattformen [24]. Die NASA benutzt Hexapoden in der Raumfahrt als Teil eines Dockingmechanismus [25]. In der Orthopädie werden Hexapoden zur Behandlung schwerer Knochenfrakturen verwendet [26]. Als Teil von Operationsassistenten finden Hexapoden außerdem Verwendung in der klinischen Medizin [27].

## 2.2. Steuerung und Regelung von WZM

### 2.2.1. CNC

In Abschnitt 2.1.2 ist die Notwendigkeit erläutert, statt einer Punkt-zu-Punktbewegung eine Trajektorie exakt abzufahren. Dafür wird eine komplexe Bahnplanung benötigt. Dieser Prozess findet in der CNC statt. Die CNC ist eine Komponente der Werkzeugmaschinensteuerung. Die Ausgabe der Bahnplanung wird Trajektorie genannt und beinhaltet in definierten Zeitabständen für jede Achse einen Lagesollwert. Dafür wird der Weg zwischen Start- und Endposition nach in NC-Code programmierten Benutzervorgaben interpoliert. Um die Maschine zu schützen und die Werkzeuge zu schonen, müssen dabei statische und dynamische Grenzwerte berücksichtigt werden. NC-Code enthält neben den antriebsrelevanten Daten auch Logik- und Schaltbefehle. Diese Befehle werden im Logikteil der Steuerung behandelt, der sogenannten SPS (siehe Abschnitt 2.2.2). Die beiden Einheiten arbeiten eng zusammen. Es kann beispielsweise eingestellt werden, dass bestimmte Befehle ein Anhalten der Bewegung erfordern.

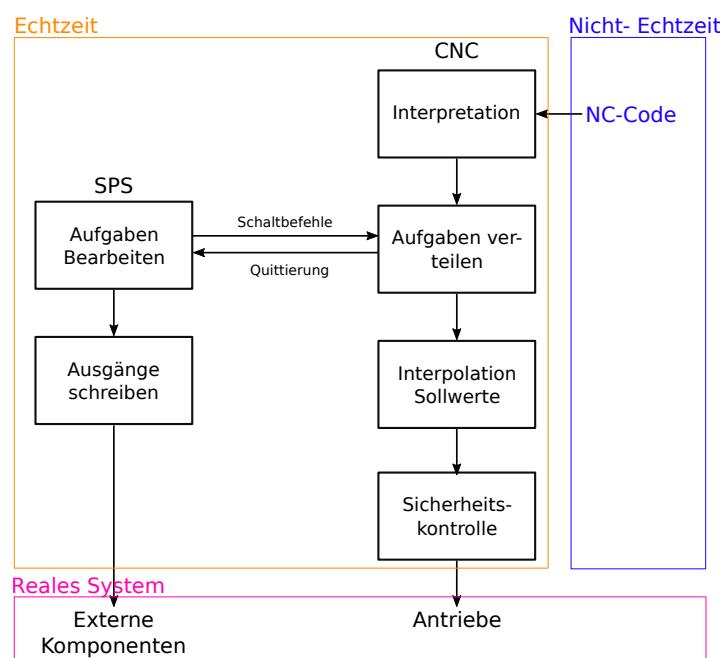


Abbildung 2.4.: Aufgabenteilung zwischen CNC und SPS

Neben der Bahnplanung übernimmt die CNC je nach Maschinentyp und Funktionsumfang noch weitere Aufgaben. Falls zum Beispiel eine Transformation zwischen Arbeits-, und Gelenkkraum notwendig ist, sorgt die CNC für deren Durchführung. Sicherheitsprüfungen verhindern ungewollte Aktionen und mindern das Havarierisiko. Außerdem verfügen moderne CNCs über die Möglichkeit, verschiedene Kompensationen auf den Sollwert aufzuschalten, wodurch sich die Präzision der Werkzeugmaschine erhöht. Normalerweise ist die Positionsregelung in den Antrieben als Kaskadenregler ausgeführt (siehe [28] für eine detaillierte Erklärung der Positionsregelung). Allerdings kann die CNC je nach Funktionsumfang und Konfiguration Teile der Positionsregelung übernehmen. Das Verhalten der CNC kann in der Regel nur im vom Herstel-

ler gesteckten Rahmen durch Ändern der Konfiguration oder NC-Befehle beeinflusst werden.

### 2.2.2. SPS

Im Gegensatz zur CNC ist die SPS ein frei programmierbarer Teil der Maschinensteuerung. Es stehen sechs verschiedene Normsprachen nach Norm IEC 61131 zu Verfügung, mit denen die Reaktion auf Logik-, Schaltbefehle und besondere Ereignisse festgelegt werden kann. Die SPS kümmert sich dabei um alles, was inhaltlich nicht mit den Antrieben verbunden ist. Teilweise koordiniert die SPS die Interaktion zwischen Bedienoberflächen und Steuerungssoftware [4]. Die SPS arbeitet zyklisch. In jedem Takt liest sie die Eingänge aus, die zu Beginn jedes Taktes in ein internes Prozessabbild gespeichert werden, damit keine Dateninkonsistenz auftritt [29]. Dann arbeitet die SPS ihr Programm, auf diese Prozessabbilder zugreifend, ab. Dabei besteht die Möglichkeit, Aufgaben an andere Teile der Steuerung abzugeben. Am Ende jedes Taktes schreibt die SPS alle Ergebnisse auf die Ausgänge, wo sie dann sämtlichen Komponenten der Steuerung zur Verfügung stehen. Die Kommunikation mit anderen Komponenten der Werkzeugmaschine geschieht über Bussysteme wie CAN-Bus. Immer öfter sind SPSen Module einer Software, statt eigene Geräte. Die sogenannten Soft-SPSen werden in Echtzeit auf Industrierechnern oder sogar auf herkömmlichen Rechnern ausgeführt.

### 2.2.3. Kraftregelung

#### Überblick

Verfügt eine Maschine nicht über eine Kraftregelung, so resultiert die Pose des Endeffektors ausschließlich aus der Lageregelung. Dabei wird der Schleppfehler, die Differenz zwischen Soll- und Istwert, über eine Kaskadenregelung (siehe [28]) in einen Momentenstellwert umgewandelt. Da die Kaskaden in der Regel über einen integrativen Anteil verfügen, steigt der Stromwert im Kontaktfall mit der Umgebung so lange an, bis zulässige Grenzwerte überschritten sind oder das Hindernis beseitigt ist. Weil der Kontakt bei Werkzeugmaschinen der Regelfall ist, birgt dieses Verhalten in den meisten Fällen Nachteile. Kraftregler sind in der Lage, den Kontakt mit der Umgebung nach genauen Vorgaben einzustellen, statt ihn wie die Lageregler als Störgröße zu behandeln.

Während die Kraftregelung in der Industrie noch nicht so verbreitet ist, gibt es in der Forschung hingegen eine Vielzahl durchgeföhrter Arbeiten und umgesetzter Ansätze. Ein Überblick befindet sich in [30]. Kraftregelungen werden überall dort eingesetzt, wo die Interaktion mit der Umgebung notwendiger Teil der Aufgabe ist (*ausgeprägter Kontakt*) oder wo die Umgebung unbekannt ist (*potentieller Kontakt*) [31]. Dabei muss die Regelungsaufgabe definiert werden, was für die Kraftregelung schwieriger ist, als für die reine Positionsregelung. Grundlage für die meisten verwendeten Definitionsansätze ist [32] die Definition eines sogenannten *Task-Spaces*, bzw. *Aufgabenraumes*. Eine anschauliche Erklärung befindet sich in [33]. Um die Aufgaben der Maschine zu generalisieren, ist der Einsatz von *Aktionsprimitiven*, einer Art Baukastensystem aus Elementaraufgaben entstanden [34]. Viele Regelungen verwenden ein Umgebungsmodell, um

die Interaktion besser einschätzen und im Voraus agieren zu können [35]. Unter anderem wird das zur Kontaktarteinschätzung verwendet [31]. Liang u. a. haben ein Verfahren entwickelt, um die im Kontaktfall unabhängigen von den abhängigen Koordinaten zu entkoppeln [36]. Rusin beschreibt einige Möglichkeiten zur Kontaktsuche [31]. Die Kraft im Kontaktfall wird häufig mit *PID*-Reglern eingestellt, wobei auch alternative Ansätze, beispielsweise mit *Fuzzy-Reglern* existieren [37]. Winkler vergleicht einige Varianten, wie die Stellgröße auf die Antriebe gebracht werden kann [35]. Viele der für die Steuerung und Regelung relevanten Themen im Allgemeinen und die Kraftregelung im Speziellen, finden sich in Sicalianos *Handbook of Robotics* [8].

## Messung

Essentiell für jede Regelung ist eine präzise und schnelle Istwerterfassung. Kräfte und Momente werden aktuell in der Regel über Kraft- und Momentensensoren (KMS) erfasst, die in den Kraftfluss eingebaut werden. Sie bestehen meist aus einem mechanisch verformbaren Teil und einer Messeinrichtung, die über Kenntnis der Steifigkeit einen Kraftwert liefert. Es existieren sowohl mehr-, als auch einachsige Sensoren. Für die Verformungsmessung stehen mehrere Varianten zur Verfügung.

**Resistive Sensoren** sind meist als Dehnmessstreifen ausgeführt (siehe Abbildung 2.5) und auf die verformbare Trägerstruktur aufgeklebt. Das Messprinzip beruht auf der Abhängigkeit des elektrischen Widerstandes von der Querschnittsfläche eines Leiters

$$R = \rho \cdot \frac{l}{A} \quad (2.14)$$

wobei  $\rho$  den spezifischen elektrischen Widerstand des Sensormaterials bezeichnet,  $l$  die Länge des leitenden Teils des Sensors und  $A$  die Querschnittsfläche. Bei Verformung des Kraftaufnehmers wird der angebrachte Sensor gestreckt oder gestaucht, was durch die Querkontraktion des leitenden Materials zu einer Querschnittsflächenänderung und damit zu einer Widerstandsänderung führt.

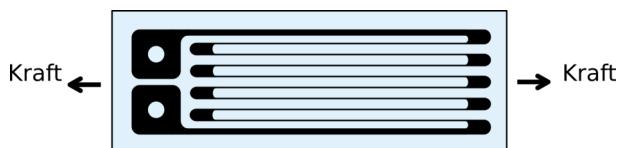


Abbildung 2.5.: Dehnungsmessstreifen [38]

Obwohl das Verfahren anfällig gegenüber Temperaturänderungen ist und über eine relativ niedrige Ansprechschwelle verfügt [39], ist das Verfahren am weitesten verbreitet. DIN 1319 definiert die Ansprechschwelle als „kleinste Änderung des Wertes der Eingangsgröße, die zu einer erkennbaren Änderung des Wertes der Ausgangsgröße eines Messgerätes führt“.

**Kapazitive Sensoren** sind deformierbare Kondensatoren. Wie beim resistiven Verfahren wird die Last über die Verformung einer Trägerkonstruktion aufgenommen. Das führt dazu, dass der Plattenabstand des Kondensators verändert wird, was eine Kapazitätsänderung zur Folge hat [40]:

$$C = \epsilon_0 \cdot \frac{A}{d}, \quad (2.15)$$

wobei  $\epsilon_0$  konstante Permittivitätszahl des Sensormaterials bezeichnet,  $d$  den Plattenabstand und  $A$  die Kondensatorfläche. Dabei ändert sich die Kapazität des Kondensators, was Rückschlüsse auf die anliegende Kraft möglich macht. Das Verfahren ist noch nicht sehr etabliert, was an den hohen Anforderungen an die Sensitivität der Kapazitätsmessung liegen kann.

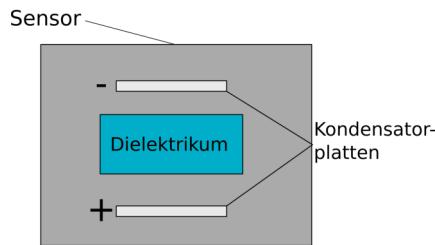


Abbildung 2.6.: Aufbau eines kapazitiven Kraftsensors

**Piezoelektrische Sensoren** weisen eine hohe Dynamik auf und eine niedrige Ansprechschwelle [39]. Sie können sehr steif ausgeführt werden und sind im Anwendungsbereich nicht temperaturabhängig [41]. Abb. 2.7 stellt dar, wie aus äußerer Krafteinwirkung eine Ladungsdifferenz wird, welche als Spannung gemessen werden kann. Es gibt 6D-Kraftmessdosen der Firma Kistler [42], die dieses Verfahren verwenden. Ein Problem des Piezokraftmessverfahrens ist jedoch, dass das Signal driftet [43].

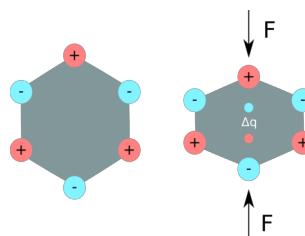


Abbildung 2.7.: Piezoelektrischer Effekt

Die Bauart des Verformungskörpers hängt im wesentlichen vom Verwendungszweck ab. Abbildung 2.8 zeigt Beispiele für mögliche Ausführungen. Ein inhärentes Problem der Verwendung von Kraftmessdosen ist, dass diese die Steifigkeit des Systems herabsetzen, auch wenn in [45] gezeigt werden konnte, dass sich die Einflüsse unter gewissen Umständen in Grenzen halten. Außerdem verkleinern sie den Arbeitsraum, wenn sie am Endeffektor angebracht sind. Als Alternative zur sensorgestützten KMS ist es auch möglich, angreifende Kräfte und Momente über die Motorströme abzuschätzen.



Abbildung 2.8.: Links: 6D-Variante von Kistler [42], rechts: 1D-Variante von ME-Systeme [44]

Das Verfahren setzt keine konstruktiven Änderungen voraus und verursacht keine Materialkosten, weil die Motorströme für die Lageregelung sowieso gemessen werden und deswegen verfügbar sind. Allerdings sind die Messwerte häufig sehr ungenau [6], weil Reibungseffekte in der Regel schlecht im Modell abbildbar und die Antriebe weit vom Prozess entfernt sind.

Kraftmessungen enthalten stets die Resultierende aus dynamischen und Kontaktkräften, weshalb ein Modell benötigt wird, um die dynamischen Kräfte zu erhalten. Modellfehler beeinträchtigen die Ergebnisse unterschiedlich stark. Die Motorstromkraftmessung ist sehr stark davon betroffen, weil die gesamte kinematische Kette und der Antriebsstrang zwischen der angreifenden Kraft und dem Messpunkt liegen.

### Transformation

Vor der Regelung ist es notwendig, alle betrachteten Größen in ein gemeinsames Koordinatensystem zu bringen. Während die Posetransformation in Abschnitt 2.1.2 beschrieben ist, geht es in diesem Abschnitt um die Transformation von Lasten. Dabei ist zwischen Kräften und Momenten zu unterscheiden. Momente haben keinen Bezugspunkt und wirken an jeder Stelle gleich. Kräfte hingegen greifen an einem Punkt an und erzeugen zusätzliche Momente, wenn man sie von einem Punkt aus betrachtet, der nicht auf der Kraftangriffsachse liegt. Folgende Abbildung illustriert das Problem im Zweidimensionalen:

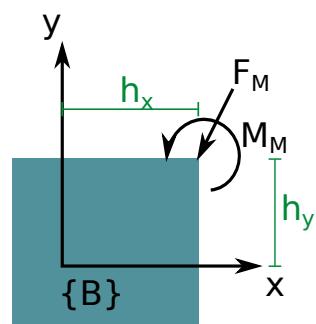


Abbildung 2.9.: Objekt mit im Punkt  $M$  angreifender Kraft  $F_M$  und Moment  $M_M$

Betrachtet man die Last im Ursprung des Koordinatensystems, so ergeben sich die jeweiligen Komponenten zu

$$F_{B,x} = F_{M,x} \quad (2.16)$$

$$F_{B,y} = F_{M,y} \quad (2.17)$$

$$M_B = M_M + F_{M,x} \cdot h_y + F_{M,y} \cdot h_x \quad (2.18)$$

Im dreidimensionalen sind es sechs Komponenten. Verallgemeinert kann die Zerlegung in die Richtungen des betrachteten Koordinatensystems mit der Rotationsmatrix  ${}^B R_M$  (siehe Abschnitt 2.1.1) und der Matrix  $\mathbf{h}$ , welche die momentbildenden Hebel beinhaltet, ausgedrückt werden [37]:

$$\mathbf{F}_B = {}^B R_M \cdot \mathbf{F}_M \quad (2.19)$$

$$\mathbf{M}_B = {}^B R_M \cdot \mathbf{M}_M + \mathbf{h} \times ({}^B R_M \cdot \mathbf{F}_M) \quad (2.20)$$

Die Rotationsmatrix vor dem Moment kommt daher, dass im Mehrdimensionalen auch Momente anteilig auf die verschiedenen Koordinatenachsen entfallen können. In Matrixschreibweise wird aus obigen Gleichungen

$$\begin{pmatrix} \mathbf{F}_B \\ \mathbf{M}_B \end{pmatrix} = \underbrace{\begin{pmatrix} {}^B R_M & 0 \\ \mathbf{h} \times {}^B R_M & {}^B R_M \end{pmatrix}}_{{}^B X_M^f} \cdot \begin{pmatrix} \mathbf{F}_M \\ \mathbf{M}_M \end{pmatrix}, \quad (2.21)$$

wobei  ${}^B T_M^f$  die Krafttransformation Punkt  $M$  zu Punkt  $B$  bezeichnet. In der Literatur wird  $\mathbf{h} \times {}^B R_M$  oft als  $\mathbf{S}(\mathbf{h}) \cdot {}^B R_M$  ausgedrückt, wobei die schiefsymmetrische Matrix  $\mathbf{S}(\mathbf{h})$  eine Umwandlung des Kreuzproduktes in ein Skalarprodukt ermöglicht.

Die Transformation einer kartesischen Kraft in den Gelenkraum findet auf Grundlage des Satzes der virtuellen Arbeit [1] analog zu Gl. 2.10 statt:

$$\mathbf{F}_q = J_{kin}(\mathbf{x})^T \mathbf{F}_x, \quad (2.22)$$

wobei  $\mathbf{F}_x$  den sechsdimensionalen Kraft- und Momentenvektor  $\{f_x, f_y, f_z, m_a, m_b, m_c\}$  im kartesischen und  $\mathbf{F}_q$  das Äquivalent im Gelenkraum beschreibt. Die Rücktransformation geschieht über die Inverse der transponierten Jacobimatrix:

$$\mathbf{F}_x = J_{kin}(\mathbf{x})^{-T} \mathbf{F}_q. \quad (2.23)$$

## Regelprinzipien

Im Rahmen dieser Arbeit werden ausschließlich positionsbasierte Kraftregelungen betrachtet. Das heißt, dass der Kraftregler aus einer Kraftabweichung ein Weginkrement generiert, das

dann als Stellgröße in die Lageregelung einfließt. Damit erreicht die Kraftregelung eine Veränderung der Pose und nimmt somit Einfluss auf die Kontaktkräfte. In den meisten Fällen erfolgt die Kraftvorgabe im kartesischen Arbeitsraum der Maschine. Allerdings ist es unumgänglich, die Stellwerte im Gelenkraum der Kinematik anzugeben, weil die Antriebe in den meisten Fällen über eine Einzelgelenkregelung verfügen. Bei verkoppelten Kinematiken ist die Transformation zwischen beiden Räumen nichtlinear (siehe Abschnitt 2.1.2). Daraus folgt, dass das Kraftregelungsmodul zwischen Ein- und Ausgabe eine Transformation vornehmen muss. Je nachdem wo und wie diese Transformation erfolgt, ergeben sich mehrere Kraftregelstrategien.

**Regelung im Gelenkraum** Bei der Kraftregelung im Gelenkraum werden alle Kräfte, mit Hilfe von Gl. 2.22 vor dem Eingang des meist als PID ausgeführten Kraftreglers in den Gelenkraum der Maschine umgewandelt.

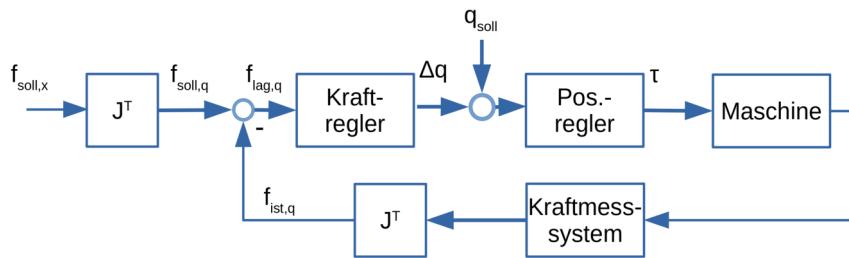


Abbildung 2.10.: direkte, parallel-hybride Kraftregelung im kartesischen Raum mittels invertierter Jacobimatrix[28]

Die Regelung ist sehr robust, weil die Jacobimatrix immer transponierbar ist, auch in der Nähe singulärer Positionen. Der Nachteil besteht in der Nichtlinearität der Stellgröße zur kartesischen Vorgabe; Eine in x-Richtung geregelte Kraft verursacht nicht nur eine Bewegung in x-Richtung, sondern auch in andere kartesische Richtungen.

**Kartesische Regelung über die inverse Jacobimatrix** Bei kartesischen Regelungen besteht der Unterschied darin, dass die Umwandlung vom kartesischen in den Gelenkraum nach Berechnung der Stellgröße geschieht. Die Bewegung erfolgt linear zum kartesischen Raum. Eine in x-Richtung geregelte Kraft verursacht also nur eine Bewegung in x-Richtung. Es gibt mehrere Verfahren, das Weginkrement umzurechnen. Hier wird zur Umwandlung die invertierte Jacobimatrix genutzt.

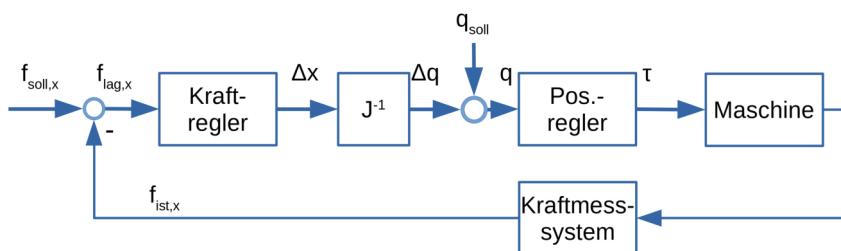


Abbildung 2.11.: direkte, parallel-hybride Kraftregelung im kartesischen Raum mittels invertierter Jacobimatrix[28]

Die Jacobimatrix ist, wie in Abschnitt 2.1.2 erklärt, nicht immer invertierbar. In der Nähe singulärer Posen, sowie für größere Inkremente [6] verliert eine derart vorgenommene Umrechnung erheblich an Präzision. Allerdings ist die Invertierung nicht rechenaufwändig und somit für alle Maschinentypen ohne große Zeitverluste durchführbar.

**Kartesische Regelung über die vollständig inverse Kinematik** Diese weitere Form der kartesischen Regelung unterscheidet dadurch von der inversen Jacobimatrix, dass vor der Umrechnung aus dem Inkrement eine absolute Pose gemacht wird, die dann durch die Rückwärtskinematik umgerechnet werden kann.

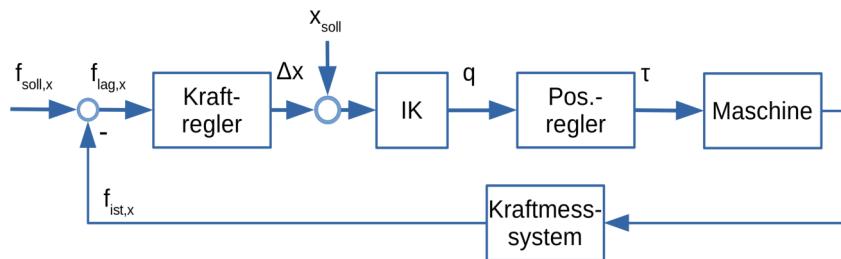


Abbildung 2.12.: direkte, parallel-hybride Kraftregelung im kartesischen Raum mittels vollständig inverser Kinematik[28]

Dafür müssen die parallelen Positionssollwertvorgaben ebenfalls im kartesischen Raum vorliegen. Falls das nicht gegeben ist, können sie durch die rechenintensive Vorwärtstransformation erlangt werden. Ein großer Nachteil dieser Variante ist der Rechenaufwand, der besonders für Parallelkinematiken aus der Berechnung der direkten und inversen Kinematik kommt und ungleich höher ist, als bei der Inversen der Jacobimatrix. Die Vorteile dieser Varianten bestehen darin, dass die Umrechnung immer präzise und berechenbar ist.

## Anwendungen

Bei Werkzeugmaschinensteuerungen ist die kommerzielle Integration von Kraftreglern noch nicht so weit fortgeschritten, wie bei Robotersteuerungen [29]. Es existieren einige Steuerungen, die eine Achse Kraftregeln können [6]. Um verkoppelte Kinematiken mit einer Kraftregelung zu versehen, ist es allerdings notwendig, alle Achsen kraftgeregt zu betreiben. Callegari u. a. haben eine Kraftregelungssimulation für eine parallelkinematische Maschine entworfen und simuliert, die Steuerung jedoch nicht auf eine reale Maschine mit Echtzeitbetrieb gebracht [46]. An der Universität in Clermont-Ferrand haben mehrere Forschungsarbeiten [47, 16] an Hexapoden zum Fräsen und für Materialversuche stattgefunden. Steuerung und Simulation sind in Matlab umgesetzt, die Positionsbestimmung im Arbeitsraum findet über Kameras statt. Le Flohic konnte die Steuerung allerdings nicht an der realen Maschine validieren, weil die Zykluszeit nicht eingehalten werden konnte [47]. Stattdessen wurde dafür eine andere Kinematik verwendet. Auch Bellakehal u. a. verzichten auf eine Validierung an der reellen Maschine [16]. An der TU Dresden haben ebenfalls Forschungsarbeiten zur Kraftregelung am Hexapod stattgefunden [7, 6]. Weil die vorliegende Arbeit konkret zum Ziel hat, die Leistungsfähigkeit dieser

Kraftregelung zu verbessern, wird die bisherige Lösung ausführlich in Abschnitt 2.3.2 beschrieben.

In der Robotik gibt es schon einige kommerzielle Lösungen für den Einsatz von Kraftregelungen in der Robotersteuerung. Als Beispiele seien ABBs *Integrated Force Control* und Kukas *FTCtrl* genannt. Kraftgeregelter Roboter finden sich vor allem in der Fertigung. Kraftgeregeltes Rührreibschweißen wird eingesetzt, um Auto- und Flugzeugteile zu fertigen [48]. Roboter von Fanuc werden zur Montage, zum Schleifen und Polieren verwendet [29]. Kooperative Roboter verwenden Kraftregelungen, um Positionierfehler auszugleichen [1], Montageroboter können somit Fertigungstoleranzen ausgleichen und auf unerwartete Kollisionen reagieren [49]. Die Anwendung in der Montage ist allerdings nicht so weit verbreitet, mögliche Gründe könnten neben der teuren Sensorik die Beschaffenheit der Roboterprogrammiersprachen und das teach-in sein, die den Einsatz durch geringe Flexibilität erschweren [34].

Ein weiteres Feld, in dem Kraftregelungen zunehmend Anwendung finden, ist die Medizintechnik. Sie helfen, bei Operationsrobotern das Verletzungsrisiko zu mindern [50]. Es existieren auch Endoskopiegeräte, die eine Kraftregelung verwenden, um aus Bedienereingaben die Werkzeugposition zu generieren [51]. Durch Kraftregelung können Ärzte beim Training durch Apparate mit Force-Feedback unterstützt werden [52].

## 2.3. Beschreibung des Versuchsträgers

### 2.3.1. Hardware

**Hexapod Felix I** Bei der zu regelnden Kinematik handelt es sich um den Hexapod *Felix I* der TU Dresden. Dieser Absatz beschreibt, auf den allgemeinen Beschreibungsformen der vorherigen Absätze aufbauend, jene mechanischen Eigenschaften der parallelkinematischen Werkzeugmaschine, die für Steuerung und Regelung relevant sind. Abbildung 2.13 zeigt die eine CAD-Darstellung der Konstruktion. Darin ist zu erkennen, dass der Hexapod neben der Hauptstruktur noch aus einer zweiten Hexapodplattform besteht. Diese ist nicht aktuiert und wird genutzt, um Abstand zwischen Handgelenken und Endeffektor zu schaffen.

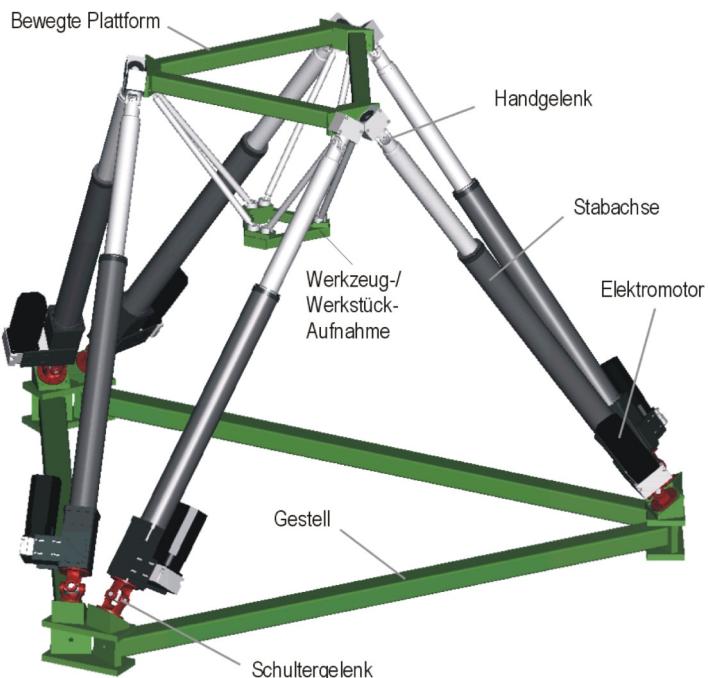


Abbildung 2.13.: CAD-Darstellung des Hexapods *Felix I* der TU Dresden [53]

Die parallele Struktur besteht aus sechs kinematischen, jeweils seriellen Teilketten. Jede dieser Teilketten weist fünf passive und einen aktiven Freiheitsgrad auf. Letzterer besteht in Richtung der Stabachsen. Über Riemen- und Spindeltrieb ist der Hexapod mit sechs Servomotoren verbunden, die jeweils eine Achse aktuieren. Der Gelenkraum des *Felix I* ist über die flexiblen Stablängen definiert:

$$\mathbf{q} = \begin{pmatrix} q_0 & q_1 & q_2 & q_3 & q_4 & q_5 \end{pmatrix}^T \quad (2.24)$$

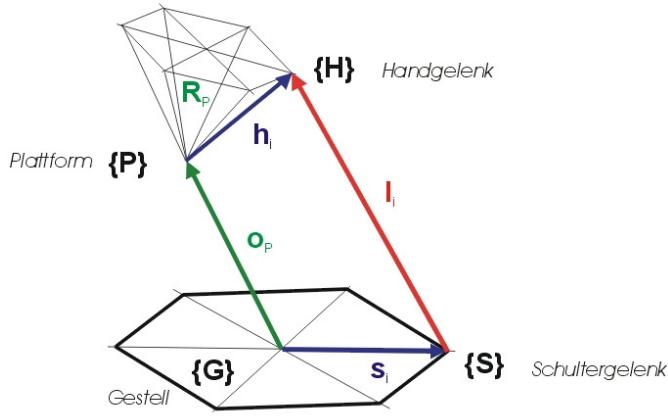


Abbildung 2.14.: Minimaldarstellung der kinematischen Kette als Basis für die Transformationen [53]

Sofern keine Werkzeuge angebaut sind, ist der Mittelpunkt  $\{P\}$  der Werkzeugaufnahme als TCP definiert. Die inverse Kinematik (siehe Gl. 2.8) lässt sich für den Hexapod nach [53] aus der vektoriellen Darstellung der Pose beschreiben:

$$\mathbf{l}_i = {}^B \mathbf{o}_P + {}^B \mathbf{R}_P \cdot {}^P \mathbf{h}_i - \mathbf{s}_i, \quad (2.25)$$

wobei  ${}^B \mathbf{o}_P$  der kartesischen Position  ${}^B \mathbf{R}_P$  entspricht,  $R_P$  der Rotation der Plattform gegenüber dem Basiskoordinatensystem und  ${}^P \mathbf{h}_i$ , sowie  $\mathbf{s}_i$  den Abständen der jeweiligen Gelenke zu ihren Koordinatensystemen. Durch Kenntnis der unveränderlichen Längen jeder kinematischen Teilkette kann von  $\mathbf{l}_i$  auf  $\mathbf{q}_i$  geschlossen werden. Dabei muss die Rückdrehung der Spindel durch die Einflüsse der anderen Achsen auf die Pose mit einberechnet werden [53]. Für die direkte Kinematik nach Gleichung 2.8, die in [53] ausführlich hergeleitet wird, ist es notwendig, die Jacobimatrix zu bestimmen. Weil die Berechnung aufwändig ist, wird die Inverse konstruiert, die dann durch einfaches Invertieren zur eigentlichen Matrix führt. Die Jacobiinverse für Hexapoden lautet nach [18]:

$$\mathbf{J}_{kin}(\mathbf{x})^{-1} = \begin{pmatrix} {}^B \mathbf{n}_1(\mathbf{x})^T & ({}^B \mathbf{h}_1(\mathbf{x}) \times {}^B \mathbf{n}_1(\mathbf{x}))^T \\ \vdots & \vdots \\ {}^B \mathbf{n}_6(\mathbf{x})^T & ({}^B \mathbf{h}_6(\mathbf{x}) \times {}^B \mathbf{n}_6(\mathbf{x}))^T \end{pmatrix}, \quad (2.26)$$

wobei  $\mathbf{n}_i$  die normierten Richtungsvektoren der Stabachsen bezeichnen und aus Gl. 2.25 folgen.

**Sensorik** Die Kinematik ist mit Positionsencodern ausgestattet, die es ermöglichen, die Position im Gelenakraum zu messen. Die Encoder sind auch in der Lage, Geschwindigkeiten zu messen. Um Beschleunigungen zu erlangen, wird numerisch abgeleitet.

Der Hexapod ist mit verschiedenen Kraftsensoren ausgestattet, um Kontaktkräfte für die Regelung zu erfassen. Dabei stehen drei sensorische Varianten zur Verfügung. Abbildung 2.15

zeigt die Positionen der Messstellen. Zwei dieser Varianten beruhen auf strukturintegrierter Kraftmessung. Eine Messvariante ist in die starre (1), die andere in die bewegte (2) Hexapodstruktur eingebaut. Am Endeffektor ist eine konventionelle 6D-Kraftmessdose integriert, die zur Referenzmessung verwendet wird. Eine ausführliche Beschreibung findet sich in [45].

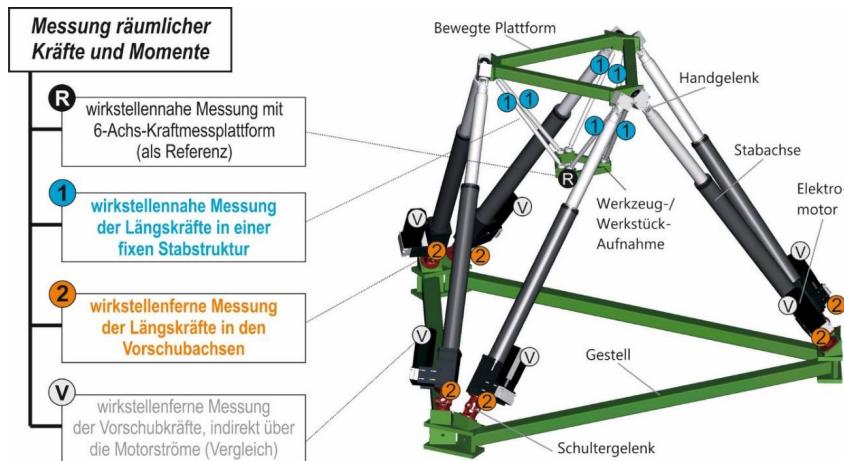


Abbildung 2.15.: Kraftsensoren des Hexapods *Felix I* [45]

Für die strukturintegrierte Messung ist in jeder Stabachse eine 1D-Kraftmessdose eingebaut. Somit kann die wirkende Kraft in jeder Achsrichtung erfasst werden. Man misst also den Kraftvektor  $\mathbf{F}_q$  der Gleichung 2.23, die dazu verwendet wird, um die kartesischen Kräfte am Endeffektor zu berechnen. Allerdings setzen sich die wirkenden Kräfte aus externen und dynamischen Kräften, sowie dem Gravitationseinfluss zusammen. Friedrich u. a. schlagen eine modellbasierte Kompensation vor, um die angreifenden Kräfte zu erhalten [45]. Dort wird außerdem eine Methode zur Parameteridentifikation des Modells angeführt. Es werden Sensoren des Typs ALF256 vom Hersteller Althen verwendet (siehe Abb. 2.16). Die wirkstellennahe Messung (1) geschieht mit der Variante für 5 kN, die wirkstellenferne Messung (2) mit der Variante für 2,5 kN. In [54] konnte gezeigt werden, dass die Verwendung strukturintegrierter Kraftmesselemente die Steifigkeit der Kinematik nicht wesentlich herabsetzt. Die Genauigkeit der strukturintegrierten Messung ist mit der konventionellen Messung vergleichbar.



Abbildung 2.16.: Am Hexapod verwendete Kraftmessdosen, Links: 1D-Kraftsensor ALF256[55] für Variante (1) und (2), rechts: 6D-Kraftsensor Omega190[56] für Variante (R)

Die konventionelle Messung wird mit dem 6D-Kraftsensor *Omega 190* des Herstellers Ati durchgeführt. Weil sich der Sensor am Endeffektor befindet, muss der Einfluss des Hexapods selbst nicht mit einem Modell ausgeglichen werden. Lediglich angebrachte Werkzeuge müssen berücksichtigt werden. Im Arbeitsraum des Hexapods ist ein Tisch montiert, auf dem Versuchsaufbauten befestigt werden können. Optional ist eine Ausstattung des Tisches mit Kraftsensoren möglich [54].

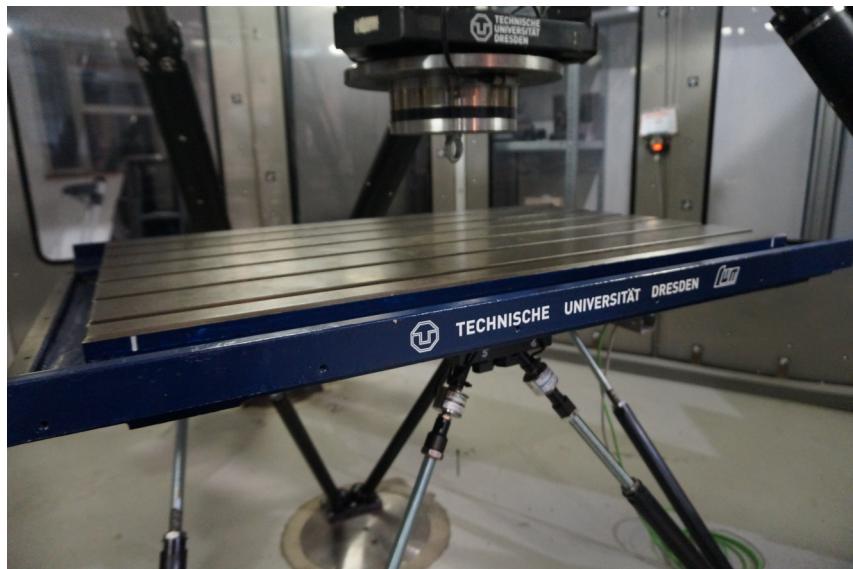


Abbildung 2.17.: Tisch im Arbeitsraum des Hexapods

### 2.3.2. Software

**TwinCAT** Der Hexapod ist mit der Steuerungssoftware *TwinCAT* des Herstellers Beckhoff ausgestattet. Es handelt sich dabei um eine Komplettlösung, die über eine in Microsofts *Visual Studio* integrierte Entwicklungsumgebung verfügt und die Steuerung auf gewöhnlichen PCs ausführen kann. Eine Besonderheit ist dabei, dass *TwinCAT* eine offene Steuerung ist. Das bedeutet, dass es möglich ist, eine Vielzahl an Komponenten, im Folgenden als *Module* bezeichnet, frei zu implementieren. Dafür stehen die Normsprachen nach IEC 61131 zur Programmierung von SPS-Modulen, sowie *Matlab* und C++ zur Verfügung. Es gibt auch die Möglichkeit, ein CNC-Modul der Firma ISG-Stuttgart zu verwenden. Die einzelnen Bausteine sind als *TcCOM*-Modul ausgeführt. Eine Kommunikation mit den Modulen ist auf unterschiedliche Weisen möglich, siehe dazu Abb.2.18. Untereinander können die *TcCOM*-Datenschnittstellen über das sogenannte *IO-mapping* miteinander verknüpft werden. Um Daten mit einem nicht echtzeitfähigen Programm auszutauschen, kann die *ADS*-Schnittstelle verwendet werden. SPS- und CNC-Module haben einen anderen Mechanismus, sie verwenden das *HLI* (High-Level-Interface). Damit können auch Daten ausgetauscht werden, die nicht zwangsläufig über das *TcCom*-Interface bereitgestellt werden. Andere *TcCom*-Module können darauf nicht zugreifen. Die *TcCOM*-Schnittstelle kann über verschiedene Bussysteme mit der Hardware verknüpft werden.

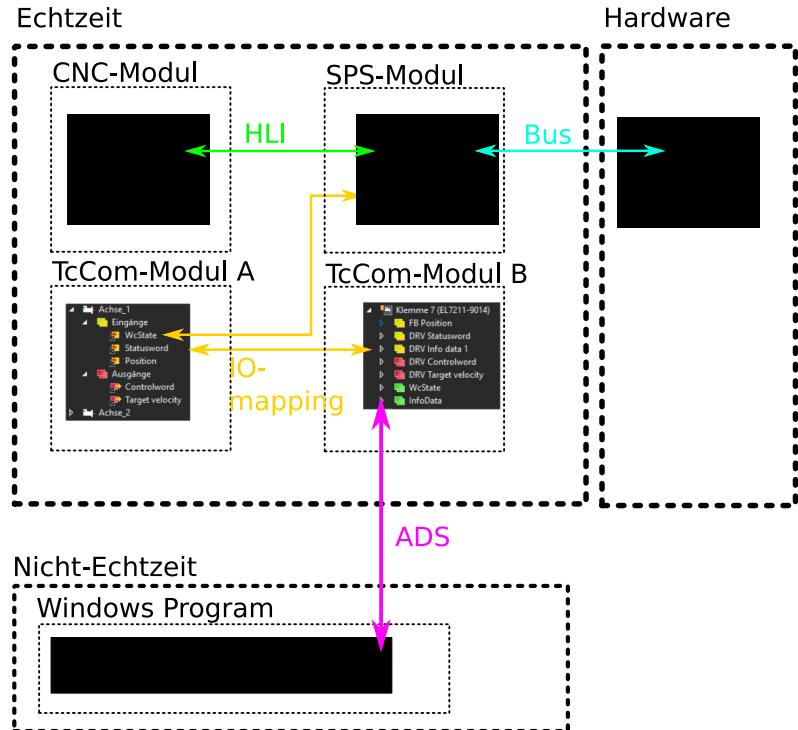


Abbildung 2.18.: Schematische Darstellung der Modulkommunikation in *TwinCAT*

*TwinCAT* ist eine echtzeitfähige Software [57]. Die Ausführung der TcCom-Module findet zyklisch statt. Dabei werden die Module in Tasks verwaltet. Jede Task hat eine Zykluszeit, die bestimmt, wie viel Zeit der Task für die Ausführung eines Taktes zur Verfügung steht. Die Echtzeitfähigkeit begründet sich auf der Nutzung des *kernel modes*. Dabei wird die Software als Treiber auf hardwarenahem Betriebssystemlevel ausgeführt. Die Folge ist neben der Einhaltung der Echtzeitbedingung nach DIN 44300 eine kurze Zykluszeit [4].

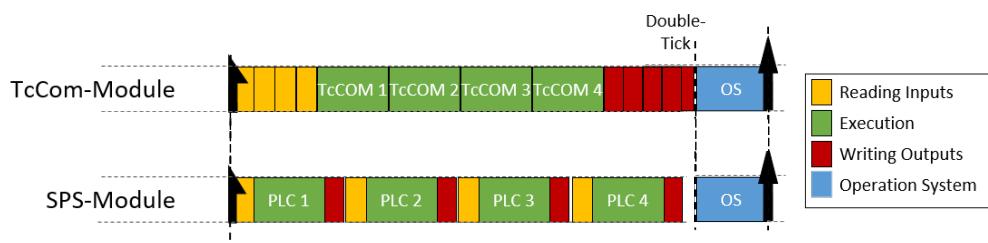


Abbildung 2.19.: Abarbeitung der Module in einem Zyklus in *TwinCAT* nach [4]

Im folgenden wird beschrieben, wie die Steuerung zu Beginn der Arbeit aufgebaut ist. Eine ausführliche Dokumentation findet sich in der dazugehörigen Diplomarbeit von Vogt [6]. In Abschnitt 5 erfolgt dann der Umbau dem Lösungsansatz entsprechend.

Tabelle 2.1.: Module der Steuerung im Ausgangszustand

	Modul	Zykluszeit	Aufgabe
CNC	GEO	1 ms	Bereitstellung der Weginformation
	SDA	1 ms	Bahninterpolation
	COM	1 ms	Bedienerkommunikation
SPS	HexapodSPS	10 ms	Schaltbefehle, HLI-Interface, Kommunikation mit CNC-Programm
	CAN_MBF	5 ms	Verwaltung des Maschinenbedienfeldes
	CNC-Feeder-SPS	1 ms	Aufschalten Kraftregelungs Wegoffset
C++	Transformation	wird aufgerufen	Rechnet zwischen Gelenkkraum und Arbeitsraum um, reentrant
	HexForce	1 ms	Verarbeitet die Kraftsensorwerte, transformiert sie in den Arbeitsraum und speichert die Ergebnisse, um sie anderen Modulen zur Verfügung zu stellen
	ForceControl	2 ms	Kraftreglung

**Bedienereingaben** Für die Bedienung des Hexapods sind in der Ausgangslage zwei Programme nötig. Die Eingabe für die Positionsregelung des Hexapods wird mit dem an der TU Dresden entwickelten Programm *HexaBOF* bewerkstelligt. Es ermöglicht die Anzeige der Positionsistwerte und die Kommandierung der Sollwerte über G-Code oder alternativ über die selbstentwickelte Sprache *Geo*, welche speziell auf die Anforderungen der Hexapodkinematik zugeschnitten ist. Die Vorgaben werden vor Weitergabe an die CNC in den Gelenkkraum transformiert.

Die Kraftregelung verwendet ein gesondertes Programm zur Vorgabe von Kraftsollwerten und Anzeige von Kraftistwerten [6]. Eine zeitgleiche Vorgabe und Interpolation von Kraft- und Positions値en ist somit nicht möglich.

**C++** Die Kraftmessung und -regelung ist in C++ umgesetzt. Das Modul *HexForce* beschäftigt sich mit der Messung der Kräfte. Dafür stehen die in Abschnitt 2.3.1 vorgestellten Sensorvarianten bereit. Um dynamische Kräfte von Kontaktkräften zu trennen, sind in diesem Modul verschiedene Kompensationsmodelle implementiert.

Das Modul *ForceControl* beinhaltet die Kraftregelung. Es stehen drei Varianten zur Verfügung:

- Regelung über die inverse Jacobimatrix,

- Regelung über die vollständig inverse Kinematik und
- Teach-in.

Für die Transformation der Ist-, und Sollwerte wird ein kinematisch-kinetisches Hexapodmodell verwendet. Die Stellgröße ist ein Weginkrement, das auf die Sollpose aufgeschlagen wird.

**CNC** Zur Sollwertinterpolation der Postionswerte wird das oben erwähnte CNC-Modul der Firma ISG genutzt. Die CNC bietet eine *externe Kommandierung* genannte Schnittstelle an, mit der man Offsets direkt auf die Sollwerte aufschalten kann (siehe [7]). Die CNC interpretiert den NC-Code und leitet nicht antriebsrelevante Inhalte wie M-Befehle an die SPS weiter. Das Bussystem *Sercos* verbindet die CNC mit den Antrieben. Lage-, Geschwindigkeits- und Stromregelung finden in den Antrieben statt.

**SPS** Kern der Logikprogrammierung am Hexapod *Felix I* ist eine SPS mit dem Namen *HexapodSPS*. Sie koordiniert die Kommunikation anderer Module mit der CNC über das HLI-Interface, sowie die Interaktion mit der HexaBOF. Wenn die CNC M-Befehle bereit stellt, verteilt die SPS diese an die entsprechenden Module und übermittelt für quittierungspflichtige M-Befehle eine Rückmeldung an die CNC. Außerdem stellt die SPS den Status und Resetanfragen der CNC für andere Module bereit.

Um Positionoffsets aus verschiedenen Quellen auf die Sollwerte der CNC aufzuschalten, wurde am IWM eine SPS entwickelt, die eine Bahnplanung berechnen kann. Dadurch stellt sie eine schonende Aufschaltung sicher. Der Vorteil dieses Ansatzes ist, dass somit eine breite Palette von Einsatzgebieten unkompliziert abgedeckt werden kann, weil die Bahnplanung automatisch eine glatte Bewegung generiert. Die Kehrseite der Flexibilität ist, dass die Bahnplanung sehr rechenintensiv ist, was zu langen Rechenzeiten führt. Dadurch entstehen Totzeiten im zweistelligen Millisekundenbereich. Die Interpolations-SPS [7] heißt *CNC-Feeder* und wird hier genutzt, um die Stellgrößen des Kraftreglers aufzuschalten.

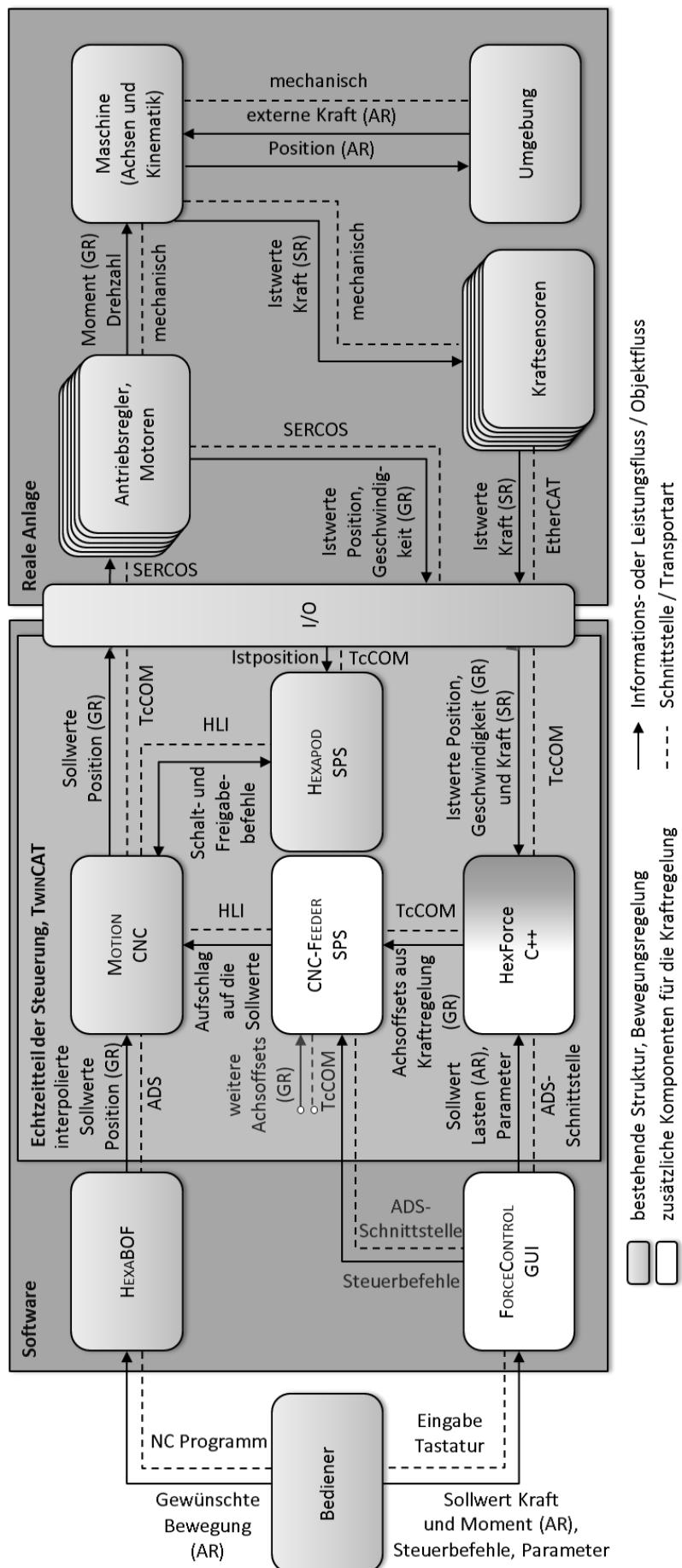


Abbildung 2.20.: Signalflussplan des Ausgangszustandes [6]

# 3. Anforderungen

## 3.1. Ziele

### Zielsetzung

Ziel des Projektes ist es, eine leistungsfähige Kraftregelung der Anlage *Felix I* zu gewährleisten, die Kraft- und Positions vorgaben gleichzeitig interpolieren und regeln kann. Es soll möglich sein, zwischen parallel-hybrider Kraft- und reiner Positionsregelung zu wechseln. Zu der bestehenden Positionsregelung im Gelenkraum soll außerdem eine kartesische Regelung im Arbeitsraum ermöglicht werden. Sowohl der Kraftregler selbst, als auch die dafür nötige Infrastruktur muss in die Steuerungssoftware *TwinCAT* integriert werden. Es soll dabei die Möglichkeit entstehen, flexibel zwischen verschiedenen Kraftmessverfahren und -Reglern zu wechseln. Die Bedienung soll sich am bisherigen Verfahren orientieren. Durch eine Zustandsmaschine soll eine transparente Kontrolle möglich sein.

### Produktziele

Durch die Software wird die fehlende Kraftregelungsintegration in traditionelle Steuerungssoftware behoben. Im Bezug auf Vorgängerarbeiten [6] steht besonders im Vordergrund, Kraft- und Positionssollwerte simultan vorzugeben und zu interpolieren, was bisher nicht möglich war. Durch die Lowlevel-Integration der Inkrementaufschaltung kommt die Steuerung ohne die redundante Bahnplanung des vorherigen Ansatzes aus, was die Totzeit im Vergleich erheblich reduziert. Außerdem wird die Handhabung durch die nahtlose Kommandierung per G-Code-Kommunikation vereinfacht. Die flexible Auswahl der Kraftmessmethode ermöglicht es, unterschiedlichen Anforderungen gerecht zu werden. Durch eine geeignete Zustandsmaschine kann ein zuverlässiges Verhalten der Steuerung garantiert werden. Die kartesische Positionsregelung ermöglicht eine einheitliche Sollwertvorgabe im Arbeitsraum und ein lineares Verhalten in Bezug auf die kartesischen Koordinatenachsen.

## 3.2. Allgemeine Beschreibung

### Produktumgebung

Das Produkt kommt auf der Werkzeugmaschine *Felix I* zum Einsatz. Die Maschine steht in einer Maschinenhalle. Die Kinematik ist räumlich von der Steuerung getrennt und eingehaust. Es sind Kräfte bis zu 5 kN zu regeln. Die Anlage wird in Fertigungsprozessen eingesetzt, was bedeutet, dass z. B. Metallspäne in den Arbeitsraum der Maschine gelangen können. An der Hexapodplattform sind Kabel zur Signalübertragung befestigt. Die Steuerung wird auf einem Standard-PC im Steuerungssoftwaresystem *TwinCAT* ausgeführt. Es werden Sercos-Antriebe verwendet, andere Varianten sind möglich.

### Produktfunktionen

Die Prozesskraft wird über einen parallel-hybriden Ansatz geregelt. Die Kommunikation zwischen CNC und Antrieb wird gewährleistet. Die Software führt Kraftregelung, Zustandsmaschine und Sollwertgenerierung aus. Das Modul schaltet das von der Kraftregelung berechnete Weginkrement auf die von der CNC generierte Trajektorie auf und gibt die so modifizierte Stellgröße im Gelenkkraum an die Antriebe weiter. Die Zustandsmaschine wird über G-Code Befehle gesteuert. Es besteht dabei die Möglichkeit, verschiedene Koordinatensysteme zu definieren, in denen dann in ausgewählte Richtungen geregelt wird.

### Benutzereigenschaften

Der Benutzer verfügt über Kenntnis des Steuerungssystems und der Hardware *Felix I*. Der Bediener ist typischerweise wissenschaftliches Personal, das speziell in den Umgang mit Maschine und Steuerung eingewiesen ist. Es handelt sich bei der Anlage um eine Forschungsmaschine. Kenntnisse der Regelungstechnik sind notwendig, um das System weiterzuentwickeln und Reglerparameter anzupassen. Im Gegensatz zur Vorgängerlösung [6] muss nicht auf die Einhaltung der Arbeitsraumgrenzen geachtet werden, weil eine interne Prüfung stattfindet. Die Verwendung des Produkts ist gefährlich. Sach- und Personenschäden können durch den ausgeprägten Kontakt schnell geschehen. Deshalb darf nur geschultes Personal die Anlage mit der Kraftregelung bedienen.

### Randbedingungen

Es dürfen keine Sprünge im Stellwert auftreten. Mechanische und dynamische Grenzwerte sind zu jedem Zeitpunkt einzuhalten. Die Kraftregelung darf bestehende Funktionalitäten des

Systems nicht beeinträchtigen. Für die Entwicklung ist ein voller Zugriff auf die Steuerung notwendig. Um für die Verwendung im Prozess eine ausreichend glatte Bewegung zu erzeugen, muss die Taktzeit von 2 ms unbedingt eingehalten werden.

## Annahmen und Abhängigkeiten

Es muss gewährleistet werden, dass *TwinCAT* nicht in der Exportversion vorliegt, weil sonst die maximale Achszahl der CNC auf vier beschränkt ist. Es muss sichergestellt sein, dass die vollständige Kommunikation zwischen Antrieb und CNC über Prozessabbilder geschieht. Sollte sich das in einer zukünftigen *TwinCAT*-Version ändern, so muss die Softwarelösung angepasst werden.

## 3.3. Spezifische Anforderungen

### Funktionale Anforderungen

- FA-1 Gleichzeitige Interpolation von Kraft- und Positions vorgaben
- FA-2 Ansteuerung der Kraftregelung über G-Code-Befehle
  - FA-2.1 Einzelne Auswahl der kartesischen kraftgeregelten Freiheitsgrade
  - FA-2.2 Vorgabe der Kraftsollwerte synchron zu den Positionssollwerten
  - FA-2.3 Definition des Aufgabenkoordinatensystems
- FA-3 Ermöglichung der Positions vorgabe im kartesischen Aufgabenraums
- FA-4 Umsetzung der Kraftregelung in allen sechs räumlichen Freiheitsgraden
- FA-5 Variable Definitions möglichkeit eines kartesischen Aufgabenraums
- FA-6 Krafttransformation vom Aufgabenraum in das Aufgabenkoordinatensystem
- FA-7 Bedienung der Kraftregelung über eine Zustands maschine
  - FA-7.1 kontrolliertes Ein- und Ausschalten
  - FA-7.2 Herausfahren des Kraftregelungsweginkrementes bei Abschalten
  - FA-7.3 Kontrolle auf zulässige Zustände der Kraftregelung
  - FA-7.4 Sperren gegen unzulässige Benutzereingaben
  - FA-7.5 automatisches und kontrolliertes Abschalten im Fehlerfall
- FA-8 Begrenzung des Kraftregelungsweginkrementes
- FA-9 Kontrolle auf die Regelgeschwindigkeit über den Override

## Leistungsanforderungen

- LA-1 Bessere Dynamik als bei der Regleraufschaltung über SPS [7]
- LA-2 Realisierung einer Reglertaktzeit von 2 ms

## Externe Schnittstellenanforderungen

- ES-1 CNC
  - ES-1.1 Übernahme der Positionssollwerte über das Achsinterface
  - ES-1.2 Übernahme der Kraftsollwerte über das Achsinterface
  - ES-1.3 Übergabe der aktuell anliegenden Istkraft über das Achsinterface
  - ES-1.4 Übergabe der Antriebsistwerte über das Achsinterface
- ES-2 SPS
  - ES-2.1 Übernahme der M- und H-Befehle
  - ES-2.2 Übernahme der Trafoinformation
  - ES-2.3 Übergabe Quittierung der M- und H-Befehle
- ES-3 Empfang der kartesischen Kraftistwerte im Plattformkoordinatensystem aus dem C++-Modul HexForce
- ES-4 Empfang der Antriebsistwerte aus den Antrieben
- ES-5 Übernahme von Zeitdaten aus dem *TwinCAT*-Echtzeitkern

## Randbedingungen

- RB-1 Lauffähigkeit in *TwinCAT* muss gewährleistet sein
- RB-2 Belegung der G-Codebefehle darf keine geschützten Wörter verwenden
- RB-3 Taktzeit 2 ms
- RB-4 Die Umsetzung soll in C++ erfolgen

## Qualitätsmerkmale

- QM-1 Die Kraftregelung ist ähnlich wie die Positionsregelung zu bedienen
- QM-2 Der Status der Kraftregelung ist gut nachvollziehbar
- QM-3 Jedes Element der Kraftregelung wird genauso schnell ausgeführt wie die Steuerungstasks in *TwinCAT*

## Sicherheitsanforderungen

SF-1 Die Kraftregelung darf nur innerhalb eines bestimmten Bereichs erfolgen. Dieser Bereich wird durch Positions- und Orientierungsgrenzen festgelegt. Der Nutzer soll den Wertebereich im G-Code anpassen können.

- a) Wenn außerhalb des Wertebereichs der Befehl zur Kraftregelung kommt, muss er abgelehnt werden
- b) Wenn die Kraftregelung die Pose so verändert, dass der Wertebereich verließe, muss die Stellgrößenaufschaltung gestoppt werden.
- c) Wenn die Stellgrößenaufschaltung gestoppt wird, muss sichergestellt werden, dass es nicht zum Wind-up kommt.

SF-2 Es darf zu keinem Zeitpunkt zu Unstetigkeiten in der Stellgröße kommen.

SF-3 Kontaktkräfte dürfen nur in den Richtungen auftreten, die durch den Selektionsvektor vorgegeben wurden. Sonst muss die Stellgrößenaufschaltung sofort gestoppt werden.

SF-4 Es darf keine Bewegung stattfinden, wenn der Override bei null ist.

SF-5 Die Kraftregelung darf nur im kartesischen Arbeitsraum, nicht im Gelenkraum stattfinden.

SF-6 Nach Deaktivieren der Kraftregelung darf sich der Endeffektor nicht mehr im Kontakt zur Umgebung befinden.

## Sonstige Anforderungen

SA-1 Die Zustandsmaschine soll modular und wiederverwendbar umgesetzt werden

# 4. Bedienkonzept

## 4.1. Sollwertvorgabe

Aus der Anforderung FA-2 geht hervor, dass die simultane Interpolation von Position und Kraft ein wesentliches funktionales Merkmal der angestrebten Lösung darstellt. Bisher wurden die Kräfte über ein externes Programm vorgegeben. Auf diese Weise ist es nicht möglich, eine Kraftinterpolation synchron auf die Trajektorie der CNC abzustimmen. Infolgedessen können in der bisherigen Lösung nur konstante Kräfte vorgegeben werden. Um die geforderte Funktionalität zu ermöglichen, sollen die Kräfte von nun an im G-Code programmierbar sein. Der G-Code in Quelltext 4.1 illustriert die Programmierung der Kraftregelung mittels G-Code. Die Auswahl der Freiheitsgrade, in denen die Kraftregelung aktiv sein soll, wird dabei durch den Kraftselektionsvektor **S** beschrieben, der in Abschnitt 5.4.4 näher erläutert wird. Alle Sollwerte beziehen sich immer auf die momentane Nutzerkoordinatentransformation, die in Abschnitt 4.2 erklärt wird.

Quelltext 4.1: Beispiel für simultane Kraft- und Positions vorgabe über G-Code: Bewegung in x-Richtung mit gleichzeitiger Kraft in z-Richtung

```
N0000 #TRAFO OFF ;(Etwaige alte Trafo deaktivieren)
N0005 #KIN_ID[65] ;(Trafo mit globalen Eulerwinkeln anwählen)
N0010 V.G.KIN_STEP[0].ID[65].PARAM[6] = 30 ;(BasisKS 30 Grad z drehen)
N0015 V.G.KIN_STEP[0].ID[65].PARAM[13] = 1 ;(Kraftregelung in x an)
N0020 #TRAFO ON ;(Trafo aktivieren)
N0025 M51 ;(Kraftregelung aktivieren)
N0030 G1 X0 ZF=10 ;(Sollposition x = 0mm, Sollkraft z = 10N)
N0035 G1 X10 ZF=5 ;(Sollposition x = 10mm, Sollkraft z = 5N)
N0040 M50 ;(Kraftregelung deaktivieren)
N0045 M30 ;(Programmende)
```

Folgende Übersicht stellt alle G-Codebefehle und -variablen dar, die zur Steuerung der Kraftregelung benötigt werden:

Tabelle 4.1.: verfügbare Befehle zur Steuerung der Kraftregelung

G-Codebefehl	Auswirkung
M50	Kraftregelung deaktivieren
M51	Kraftregelung aktivieren
M52	Weiche Regelparameter aktivieren
M53	Harte Regelparameter aktivieren
M54	variablen Task Space deaktivieren (default: radial)
M55	variablen Task Space aktivieren (default: radial)
M56	Kraftregelung in Normalenrichtung deaktivieren
M57	Kraftregelung in Normalenrichtung aktivieren (ben. M55)
#TRAFO ON / OFF	Kartesische Positions vorgabe an / aus
#KIN_ID[Trafold]	Auswahl der Transformationsart
V.G.KIN_STEP[0].ID[trafold].PARAM[1-6]	Var: Koordinatensystem Basis
V.G.KIN_STEP[0].ID[trafold].PARAM[7-12]	Var: Koordinatensystem Werkzeug
V.G.KIN_STEP[0].ID[trafold].PARAM[13-18]	Var: Kraftselektionsvektor
V.G.KIN_STEP[0].ID[trafold].PARAM[19-24]	Var: Untere Begrenzung Kraftregelraum
V.G.KIN_STEP[0].ID[trafold].PARAM[25-30]	Var: Obere Begrenzung Kraftregelraum
V.G.KIN_STEP[0].ID[trafold].PARAM[31-36]	Var: PI-Controller Tn, weich
V.G.KIN_STEP[0].ID[trafold].PARAM[37-42]	Var: PI-Controller Kp, weich
V.G.KIN_STEP[0].ID[trafold].PARAM[43-48]	Var: PI-Controller Tn, hart
V.G.KIN_STEP[0].ID[trafold].PARAM[49-54]	Var: PI-Controller Kp, hart
V.G.KIN_STEP[0].ID[trafold].PARAM[55]	x-Koordinate Mittelpunkt Radiale Regelung
V.G.KIN_STEP[0].ID[trafold].PARAM[56]	y-Koordinate Mittelpunkt Radiale Regelung
XF, YF, ZF, UF, WF, VF	Kraftsollwert

## 4.2. Benutzerdefinierte Koordinatensysteme

Nach Anforderung SF-5 (variable Nutzerkoordinatensystemdefinition) soll der Nutzer die Kraftregelung ausschließlich im kartesischen Raum verwenden. Innerhalb dieses kartesischen Raums müssen Kräfte und Posen in unterschiedliche Bezugssysteme umwandelbar sein, um den Anforderungen FA-5 und FA-6 (Kraftransformation in das Nutzerkoordinatensystem) gerecht zu werden. Dafür bietet die Software dem Benutzer die Möglichkeit, im G-Code eigene Koordinatensysteme zu definieren. In Tabelle 4.1 sind die dazu benötigten Parameter aufgeführt. Koordinatensystemdefinitionen sind im Eulerformat mit mm bzw. ° entsprechend dem Winkelformat (BLD, Euler, Kardan) der Transformation anzugeben. Details zu den verfügbaren Transformationen finden sich in Abschnitt 5.2. Es stehen zwei simultan verwendbare Koordinatensysteme bereit, die in Abbildung 4.1 dargestellt sind. In Abhängigkeit der Nutzertransformation variieren die Bezugspunkte der Sollwerte. Als erstes lässt sich ein fixes Basiskoordinatensystem definieren. Das kann beispielsweise ein Tisch sein, der sich im Arbeitsraum befindet und

als Bezugspunkt für die Programmierung dient. Dieses Koordinatensystem wird  $B_{usr}$  genannt. Außerdem kann ein mitbewegtes Koordinatensystem  $TCP_{usr}$  definiert werden, das den TCP im Plattformkoordinatensystem  $P$  darstellt. Somit kann der Benutzer auch nach einem Werkzeugwechsel die Sollposition eines veränderten TCP programmieren. Das Koordinatensystem, in dem die Sollwertprogrammierung stattfindet, wird im Folgenden unabhängig von der Definition des Koordinatensystems selbst als *Task Space* bezeichnet. Zunächst fallen  $TCP_{usr}$  und  $TS$  zusammen. In Kapitel 7.3.1 findet eine Erweiterung der Kraftregelung um variable *Task Spaces* statt.

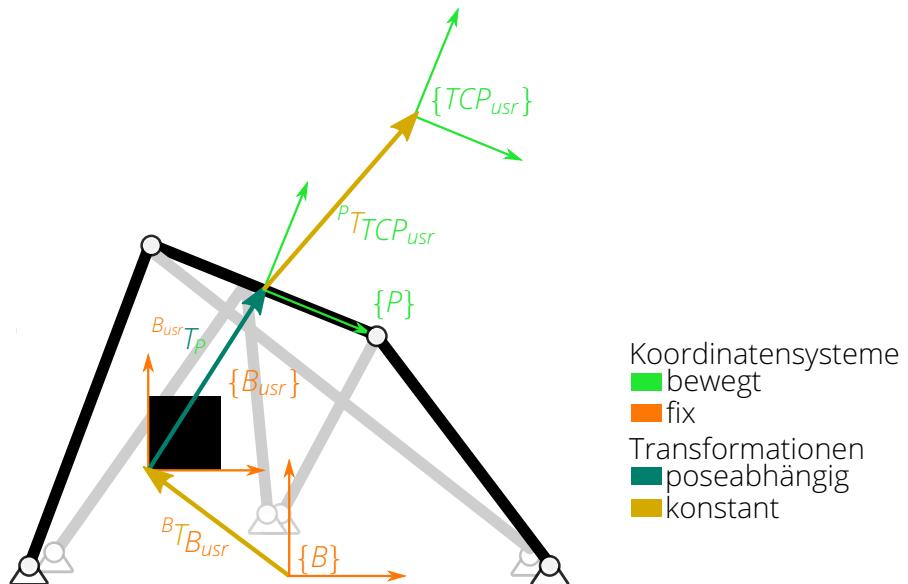


Abbildung 4.1.: Koordinatensysteme am Hexapod mit dazugehörigen Transformationen. Benutzerdefinierte Koordinatensysteme sind mit dem Index  $usr$  gekennzeichnet

Für die Regelung ist es notwendig, alle verwendeten Größen intern von der Darstellung des Endeffektors im Nutzerkoordinatensystemen in die Darstellung der Plattform in Maschinenbasiskoordinaten zu transformieren, zur Anzeige wird die Darstellung des TCP im Nutzerraum verwendet. Die notwendigen Transformationen erfolgen durch folgende Gleichungen:

$${}_{B_{usr}}T_{TCP_{usr}} = {}_{B_{usr}}T_B \cdot {}^B T_P \cdot {}^P T_{TCP_{usr}} \quad (4.1)$$

$${}^B T_P = {}^B T_{B_{usr}} \cdot {}_{B_{usr}}T_{TCP_{usr}} \cdot {}_{TCP_{usr}}T_P \quad (4.2)$$

Analog dazu finden die Kräfte folgende Transformationen statt:

$${}_{B_{usr}}F_{TCP_{usr}} = {}_{B_{usr}}X_B^f \cdot {}^B F_P \cdot {}^P X_{TCP_{usr}}^f \quad (4.3)$$

$${}^B F_P = {}^B X_{B_{usr}}^f \cdot {}_{B_{usr}}F_{TCP_{usr}} \cdot {}_{TCP_{usr}}X_P^f \quad (4.4)$$

## 5. Steuerungssoftware

Die im Rahmen dieser Arbeit umgesetzte Software baut wie die Vorgängerlösung [6] auf dem existierenden Steuerungssystem des Hexapods *Felix I* auf und ist darin fest integriert. Abbildung 2.20 zeigt die Struktur der Vorgängerlösung. Analog zu dieser Darstellung findet sich in Abbildung 5.1 die Gesamtstruktur des neuen Lösungsansatzes. Diese besteht aus modifizierten Komponenten des bestehenden Systems und einem neu entwickelten Softwaremodul. In Abschnitt 5.1 findet zunächst eine Erweiterung des CNC-Projektes statt, um die Sollwertvorgabe aus dem G-Code zu ermöglichen. Anschließend wird die CNC in Abschnitt 5.2 dazu befähigt, die kinematische Transformation online durchzuführen. Dem schließt sich die Entwicklung der Kraftregelung als C++-*TwinCAT*-Modul an, wie in Abschnitt 5.4 erläutert. Um die Kommunikation dieses Moduls mit dem HLI zu ermöglichen, findet anschließend in Abschnitt 5.3 eine Modifikation der SPS statt.

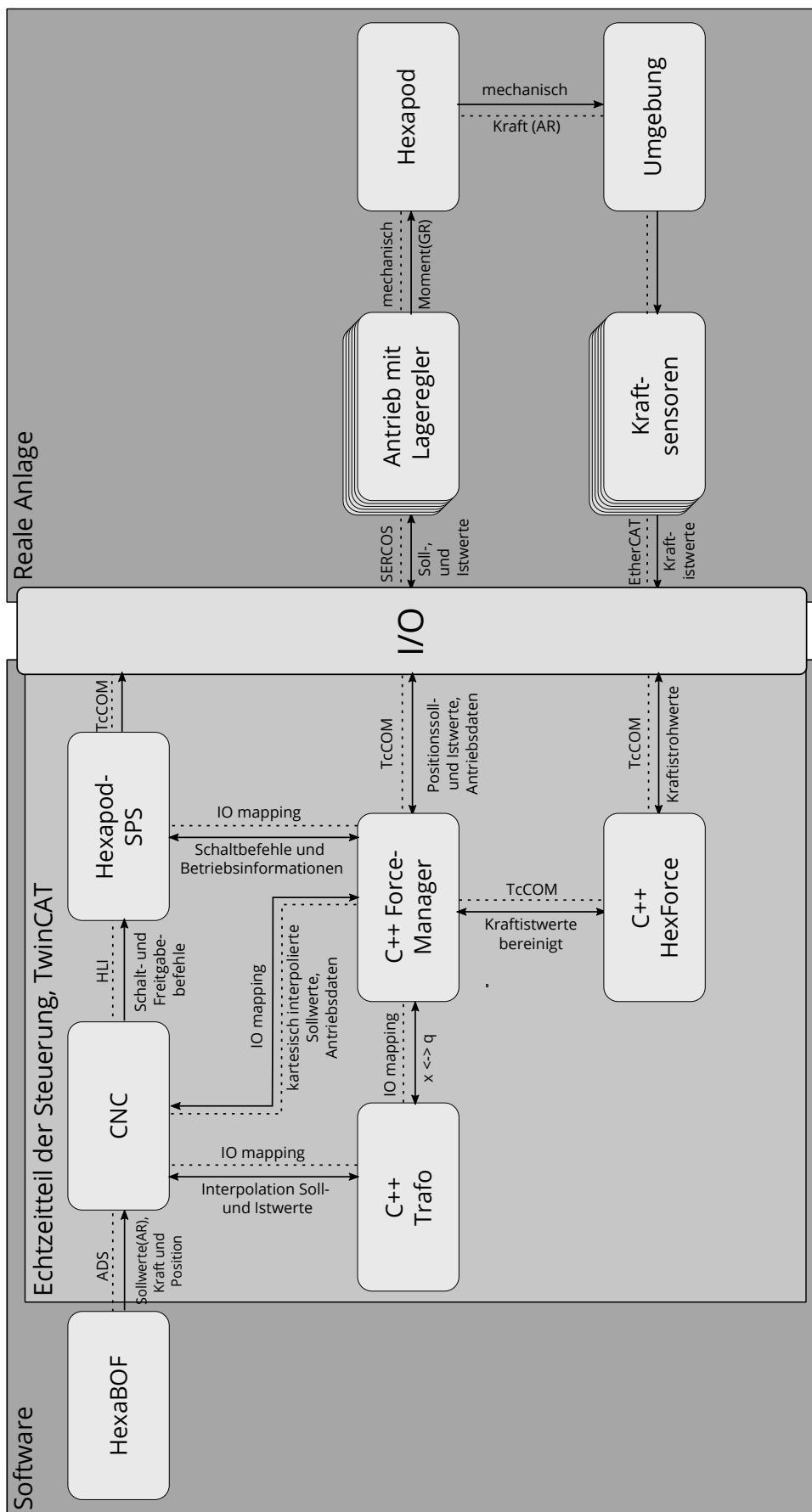


Abbildung 5.1.: Signalflussdiagramm des Gesamtsystems, die Klasse *ForceManager* implementiert die Kraftregelung

## 5.1. Anpassung der CNC

In Abschnitt 4 ist erläutert, wie der Nutzer die Software per G-Code steuern kann. Die CNC ist dafür verantwortlich, im NC-Programm enthaltene Vorgaben auszulesen, Sollwerte zu interpolieren und Istwerte zu überwachen. Das ist für die Lageregelung aktueller Stand der Technik (siehe dazu Abschnitt 2.2.1), auch in der Vorgängerlösung ist das derart umgesetzt. Um die Kraftregelung ebenfalls per G-Code zu steuern, muss die CNC die entsprechenden Befehle auch für die Kraftregelung verarbeiten. Allerdings ist die Vorgabe, Interpolation und Istwertüberwachung von Kräften in der verwendeten CNC von ISG-Stuttgart nicht vorhergesehen. Um eine Verwendung der CNC zu diesem Zweck dennoch zu ermöglichen, wird die Funktionalität der CNC genutzt, Positionsachsen zu verwalten. Dafür werden in der CNC sechs zusätzliche virtuelle Positionsachsen angelegt, die vom C++-Modul *ForceManager* als Kraftsollwerte im kartesischen Raum interpretiert werden. Dieses Modul liefert die Kraftsensordaten als Positionsistwerte zurück. Ist eine Kraftachse inaktiv, muss sie in den Nachführbetrieb geschalten werden, weil das Rauschen der Kraftsensoren sonst einen Fehler in der CNC hervorruft.

Durch die Anpassung der CNC ist die Infrastruktur *TwinCATs* nun für die Kraftregelung zugänglich. Abbildung 5.2 zeigt den daraus resultierenden Signalfluss.

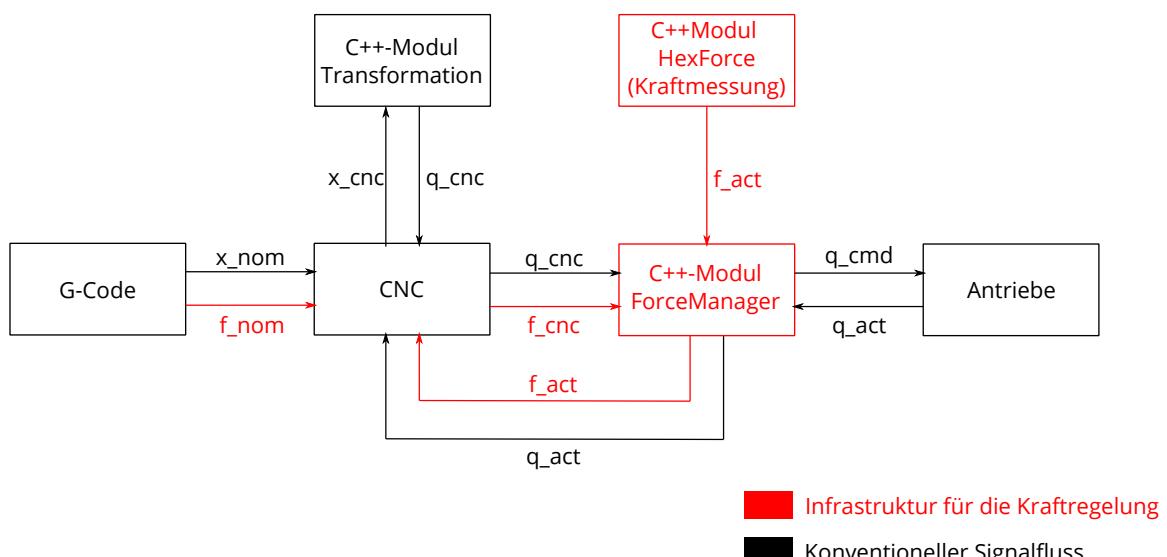


Abbildung 5.2.: Signalflussdiagramm der Kraft- und Positionsdaten

Die Positionsachsen sind mit  $x_{cnc} = \{X, Y, Z, U, V, W\}$  bezeichnet, während die als Kraftvorgabe interpretierten Achsen  $F_{cnc} = \{XF, YF, ZF, UF, VF, WF\}$  heißen.

Bisher ist die Sollwertvorgabe der Positionsregelung über den G-Code nur im Gelenkraum möglich. Die Kraftvorgabe ist jedoch nur im kartesischen Raum vorgesehen. Um einer konsistenten Bedienung den Weg zu ebnen, soll deswegen auch aus dem G-Code eine Positionsvergabe im kartesischen Raum ermöglicht werden. Dafür werden statische Vorwärts- und Rückwärtstransformation (siehe Abschnitt 2.1.2) in den CNC-Kern integriert. Das geschieht über das C++-Modul *TcNcTrafoExtCnc*, das im folgenden Absatz beschrieben ist.

## 5.2. C++-Transformationsmodul TcNcTrafoExtCnc

Wegen der nicht-linear verkoppelten Antriebe des Hexapods (siehe Abschnitt 2.1.2) spielt es für die Positionsregelung eine Rolle, ob die Sollwerte vor oder nach der Interpolation in den Gelenkraum transformiert werden. Die verschiedenen Möglichkeiten sind in Abschnitt 2.2.3 aufgeführt und illustriert. In der Vorgängerlösung werden die Sollwerte im Gelenkraum interpoliert. Dadurch ist eine im Arbeitsraum geradlinige Bewegung des Hexapods nur näherungsweise durch die Verwendung von Stützstellen zu erreichen [53]. Im Rahmen dieser Arbeit schafft das Transformationsmodul die Möglichkeit, die Transformation wahlweise nach der Interpolation vorzunehmen. Dadurch entstehen mehrere Vorteile:

- Stützstellen sind nicht mehr notwendig, um im Arbeitsraum eine geradlinige Bewegung umzusetzen.
- Die Sollpositionen können auch im G-Code kartesisch vorgegeben werden, bislang war dazu eine Offlinetransformation notwendig, die nur mittels der *GEO*-Sprache angefordert werden konnte.
- Die rechenaufwändige Transformation kann alternativ immernoch offline erfolgen, nachdem die Onlinetransformation per G-Code deaktiviert wurde (siehe Abb. 5.3).
- Die kartesische Sollposition kann vor der Transformation abgegriffen werden. Das bietet die Möglichkeit, unter der Berücksichtigung geeigneter Sicherheitsvorkehrungen die rechenintensive Vorwärtstransformation der Offsetaufschaltung in Zukunft zu ersetzen.

Damit die Transformation nach der Interpolation durchführbar ist, muss die CNC sie online im Kern aufrufen können. Die Aufgabe des Transformationsmoduls besteht deswegen darin, die Vorwärts- und Rückwärtsskinematik des Hexapods (siehe Abschnitt 2.1.2 und 2.3.1) reentrant für die CNC bereitzustellen. Dazu verwendet das Transformationsmodul die im Rahmen vergangener Arbeiten am LWM entstandene Hexapodbibliothek *CFHexapodTools*. Diese Bibliothek beinhaltet unter anderem echtzeitfähige Algorithmen zur Berechnung kinematischer und differenziell kinematischer, sowie kinetischer Transformationen, sowohl vom Gelenkraum in den Arbeitsraum, als auch umgekehrt. Das Transformationsmodul steht in drei verschiedenen Modi bereit, die dem Nutzer jeweils die Möglichkeit lassen, die Winkelvorgabe in

- globalen Eulerwinkeln (Kardanwinkeln) [id = 65],
- lokalen Eulerwinkeln [id = 66] oder
- modifizierten Eulerwinkeln (BLD) [id = 67] vorzunehmen.

Die Auswahl findet über den G-Code statt (siehe Tab. 4.1).

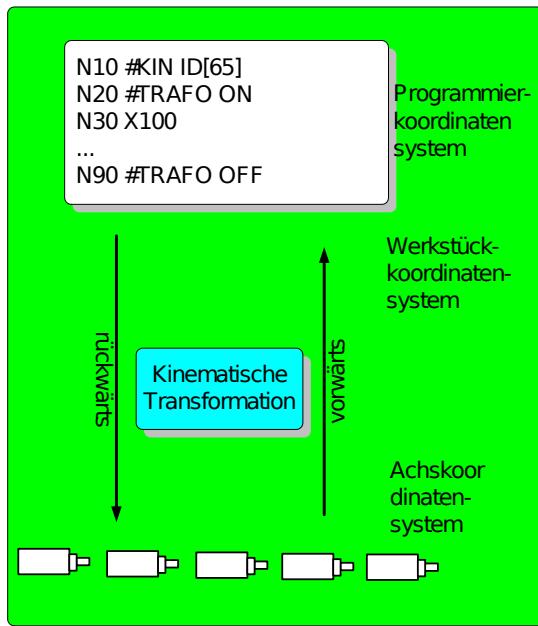


Abbildung 5.3.: Aktivieren und Deaktivieren der Trafo per G-Code [58]

Die Anbindung an die CNC geschieht durch das *TcCOM Trafo-Interface* [58]. Das Modul wird

- nach jedem Interpolationsschritt (kinematische Rückwärtstransformation) aufgerufen,
- um die Istposition für den Bediener im kartesischen Arbeitsraum anzuzeigen (kinematische Vorwärtstransformation) und
- wenn die CNC einen look-ahead durchführt (kinematische Rückwärtstransformation) [4].

Die Transformationen sind nur für die Positionsachsen wirksam. Die Kraftachsen werden nicht transformiert, weil eine Kraftregelung im Gelenkraum nicht zweckmäßig ist. Das bedeutet, dass Kräfte nur im kartesischen Arbeitsraum vorgegeben werden können.

Die in Tabelle 4.1 dargestellten *globalen Variablen V.G.* sind als Kanalparameter gespeichert (siehe dazu auch P-CHAN-00263 in [59]). Diese Parameter sind in normalen C++-Modulen nicht zugänglich. Das Transformationsmodul wird jedoch mit Parametern aufgerufen, welche diese Variablen enthalten. Um die Verwendung aus anderen Modulen zu ermöglichen, übernimmt das Modul die Kanalparameter in jedem Takt von der CNC und schreibt sie in einen gemeinsam genutzten internen Datenbereich. Das betrifft das Koordinatensystem der Basis, das Koordinatensystem des Werkzeugs, die Begrenzungen der Kraftregelung, die Reglerparameter und den Kraftselektionsvektor.

### 5.3. SPS

Die in Abschnitt 2.3.2 behandelte Schnittstelle *HLI* steht der SPS, nicht aber C++-Modulen bereit. Um ausschließlich darin verfügbare Funktionen trotzdem zugänglich zu machen, ist es notwendig, die HLI-Interaktion in der SPS *HexapodSPS* umzusetzen. Das HLI wird benutzt, um

der CNC Befehle zu erteilen oder Statusinformationen abzurufen. Für die Kraftregelung erfolgen drei Erweiterungen der SPS, um folgende Funktionen zu ermöglichen:

1. Nachführbetrieb: In Abschnitt 5.4.2 ist erläutert, unter welchen Bedingungen die Kraftachsen der CNC auf Schleppfehler überwacht werden sollen. Die Funktionalität lässt sich durch Verwendung der CNC-Funktion *Nachführbetrieb* umsetzen [60], die ausschließlich über das HLI zugänglich ist. Die SPS bietet für jede Achse getrennt die Möglichkeit, diese über das Umschalten eines Steuerbits in den Nachführbetrieb zu versetzen.
2. Der Nutzer kann im G-Code sogenannte *M-Befehle* angeben, wodurch sich die Kraftregelung wie in Abschnitt 4 steuern lässt. Die SPS setzt die C++-Module über das Eintreten neuer Befehle in Kenntnis und bietet eine Schnittstelle an, mit der die Befehle quittiert werden können.
3. Weil die Kraftregelung nur im kartesischen Raum erlaubt ist, muss den C++-Modulen der aktuelle Transformationsstatus bekannt sein. Die SPS kann auf Basis der globalen Variable *V.G.KIN\_STEP[0].ID[6x].TRAFO\_ACTIVE* auslesen, ob der Nutzer die Transformation im Gelenkraum (#TRAFO OFF) oder im kartesischen Raum (#TRAFO ON) angefordert hat und gibt diese Information weiter.

## 5.4. C++-Modul ForceManager

Das Modul *ForceManager* ist das Herzstück der Kraftregelung. Mit Ausnahme der Kraftmessung, die durch das C++-Modul *HexForce* [61] geschieht, finden im C++-Modul *ForceManager* alle zur Kraftregelung notwendigen Schritte statt. Das umfasst

- das Umleiten sämtlicher Verbindungen zwischen CNC und Antrieben,
- den Empfang und die Quittierung der Steuerungsbefehle aus der SPS
- die Kontaktsuche in den Koordinaten, die durch den Kraftselektionsvektor ausgewählt wurden,
- die Kraftregelung in den Koordinaten, die durch den Kraftselektionsvektor ausgewählt wurden,
- die Transformation der Positions- und Kraftwerte in das benutzerdefinierte Koordinatensystem  $TCP_{usr}$ ,
- das Herausfahren des Kraftregelungslageoffsets bei Abschalten der Kraftregelung oder Misserfolg der Kontaktsuche,
- das Aufschalten des Kraftregelungslageoffsets auf die interpolierten Sollpositionen der CNC und
- die Implementierung eines Sicherheitskonzepts, dass die Schäden im Fall unsachgemäßer Behandlung oder eines Programmierfehlers minimiert.

Das Modul besteht aus einer C++-Klasse, die über mehrere Unterklassen verfügt. Es ersetzt die Module *ForceControl* und *CNC-Feeder* aus der Vorgängerlösung [6, 7]. Die Hauptklasse stellt das Interface zur Kommunikation mit *TwinCAT* zur Verfügung, während sämtliche Funktionalität in den Unterklassen umgesetzt ist. Der Aufbau des *ForceManagers* ist in Abbildung 5.4 abgebildet.

Die Klasse *MCommandDispatcher* registriert Nutzerbefehle und weist die in Klasse *StateMachine* umgesetzte Zustandsmaschine an, das System kontrolliert in den korrekten Zustand zu versetzen. Die Zustandsmaschine interagiert mit der Klasse *ForceController*, die für die Stellgrößenberechnung zuständig und in Abschnitt 5.4.4 beschrieben ist. Die virtuelle abstrakte Klasse *IoHandler* stellt Ist- und Sollgrößen aus dem Steuerungssystem bereit und schreibt die Stellgröße des *ForceControllers*, sowie Statusinformationen zurück. In dieser Arbeit erfolgt die konkrete Implementierung durch den in Abschnitt 5.4.3 beschriebenen *DriveHandler* auf Basis der im Abschnitt 5.1 und 5.4.3 erläuterten *Man-in-the-middle-Aktion*. In Abbildung 5.4 findet sich eine Übersicht über den Aufbau des *ForceManagers* mit seinen Unterklassen.

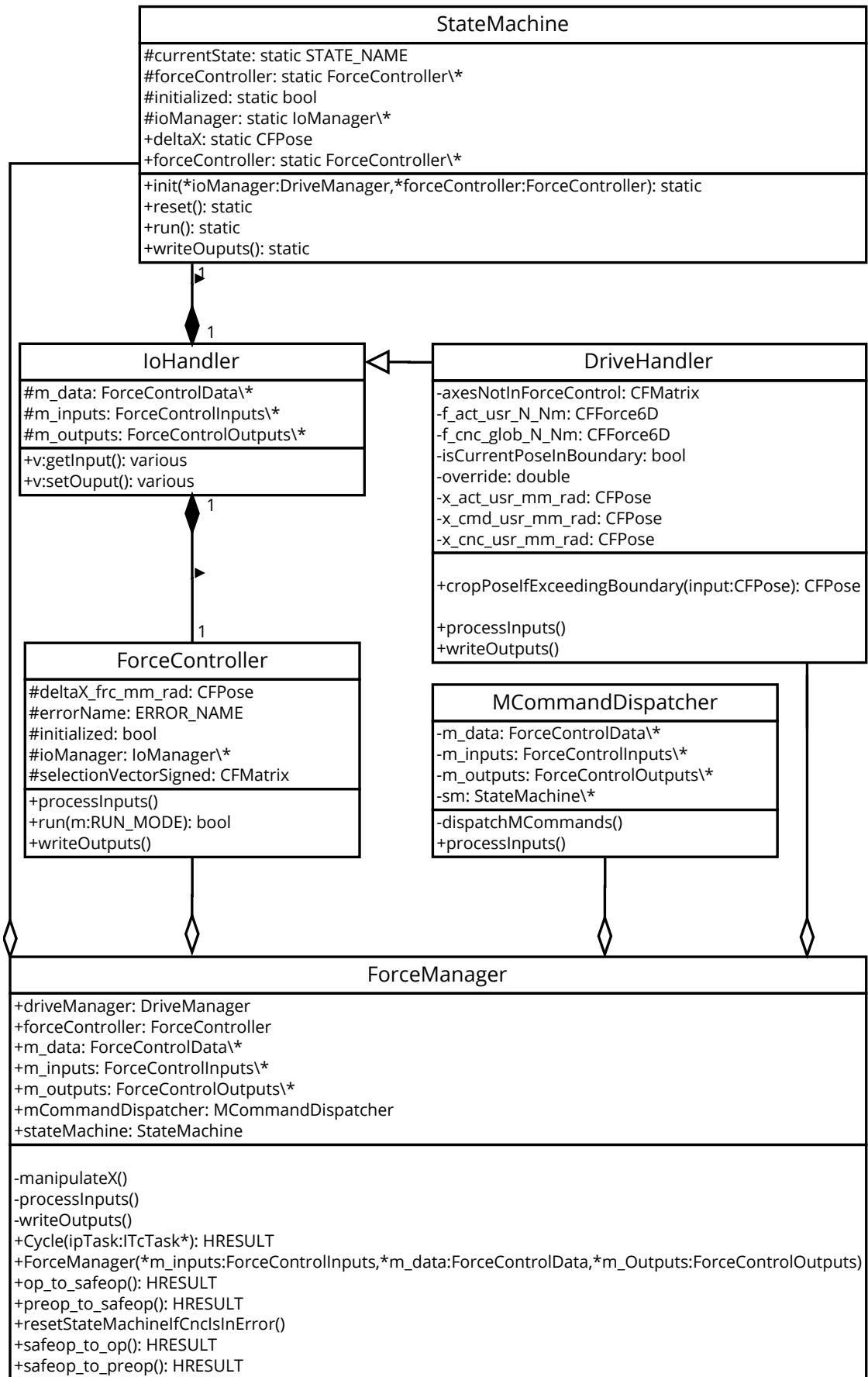


Abbildung 5.4.: UML-Klassendiagramm der Klasse `ForceManager`, zur besseren Übersichtlichkeit gekürzt

In jedem Takt führt *TwinCAT* die Funktion *CycleUpdate* aus. Die Funktion manipuliert die Positionssollwerte der CNC und führt die *Man-in-the-middle-Aktion* durch. Abbildung 5.5 bildet die dabei unternommenen Teilschritte ab. Die einzelnen, in jedem Schritt jeweils aktiven Funktionen der Unterklassen sind auf jeder rechten Seite des Aktivitätsdiagramms benannt.

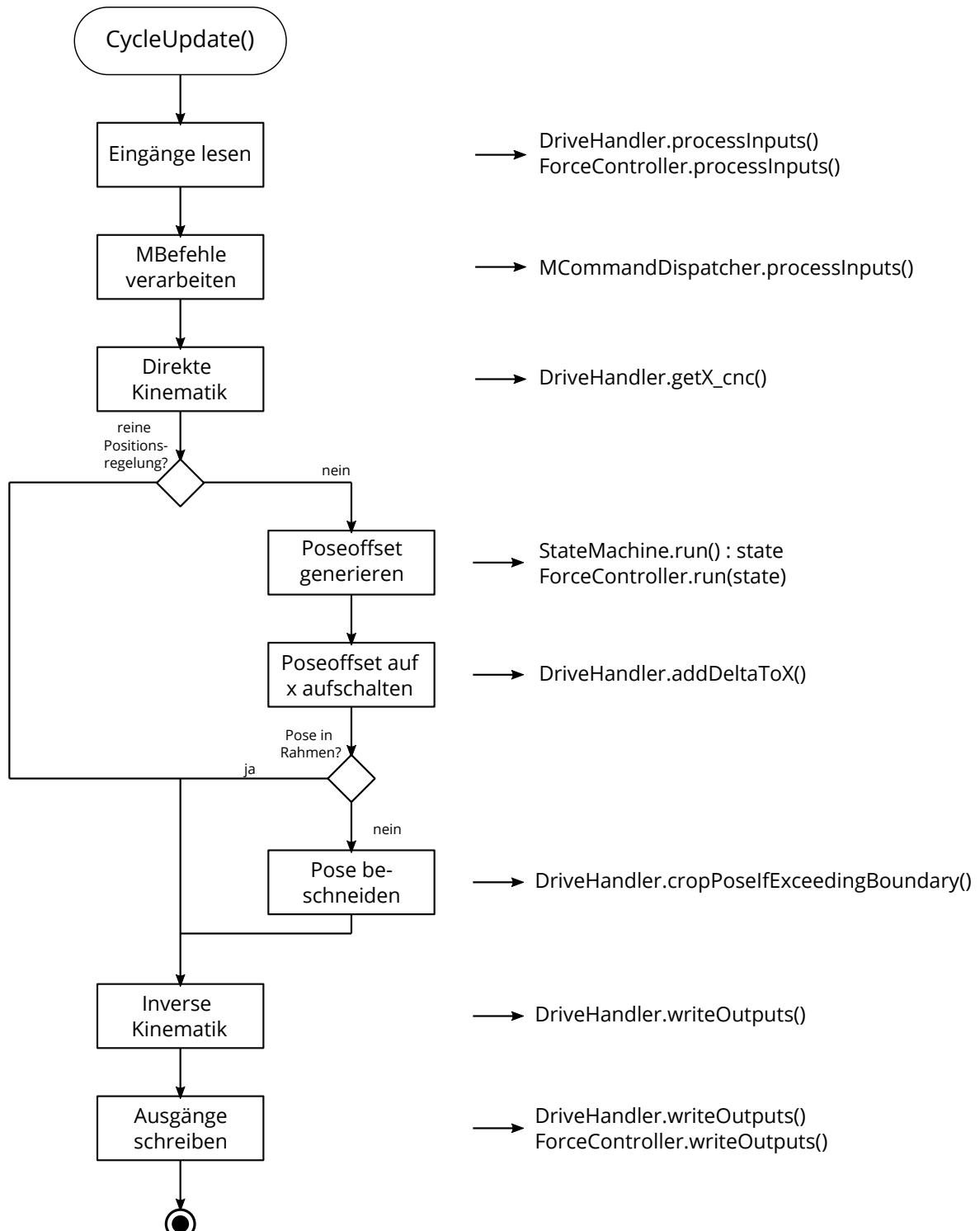


Abbildung 5.5.: UML-Aktivitätsdiagramm der Funktion *CycleUpdate*, zur besseren Übersichtlichkeit leicht gekürzt

Der *ForceManager* übernimmt nur die Koordination, stellt jedoch selbst keine Berechnungen an. Das wird in Abbildung 5.6 deutlich, die das Zusammenspiel der Unterklassen während des Durchlaufs der Funktion *CycleUpdate* als UML-Sequenzdiagramm dargestellt.

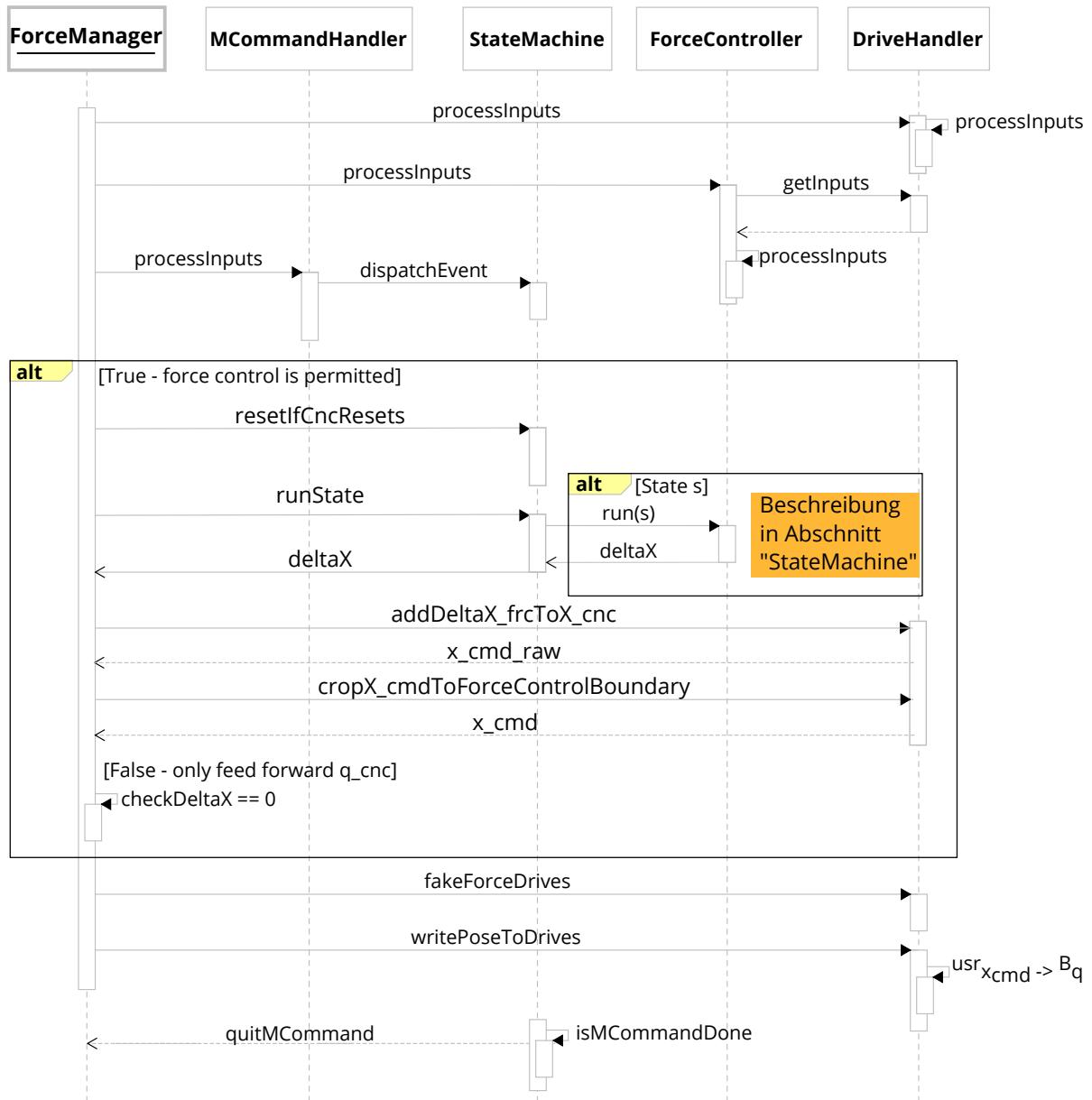


Abbildung 5.6.: UML-Sequenzdiagramm für die Interaktion innerhalb der Klasse *ForceManager*, zur besseren Übersichtlichkeit leicht gekürzt

Die folgenden Abschnitte 5.4.2 bis 5.4.4 gehen auf die Implementierung und das Zusammenspiel der Unterklassen des *ForceManagers* ein und nehmen dabei auf die UML-Diagramme Bezug.

### 5.4.1. C++-Submodul MCommandDispatcher

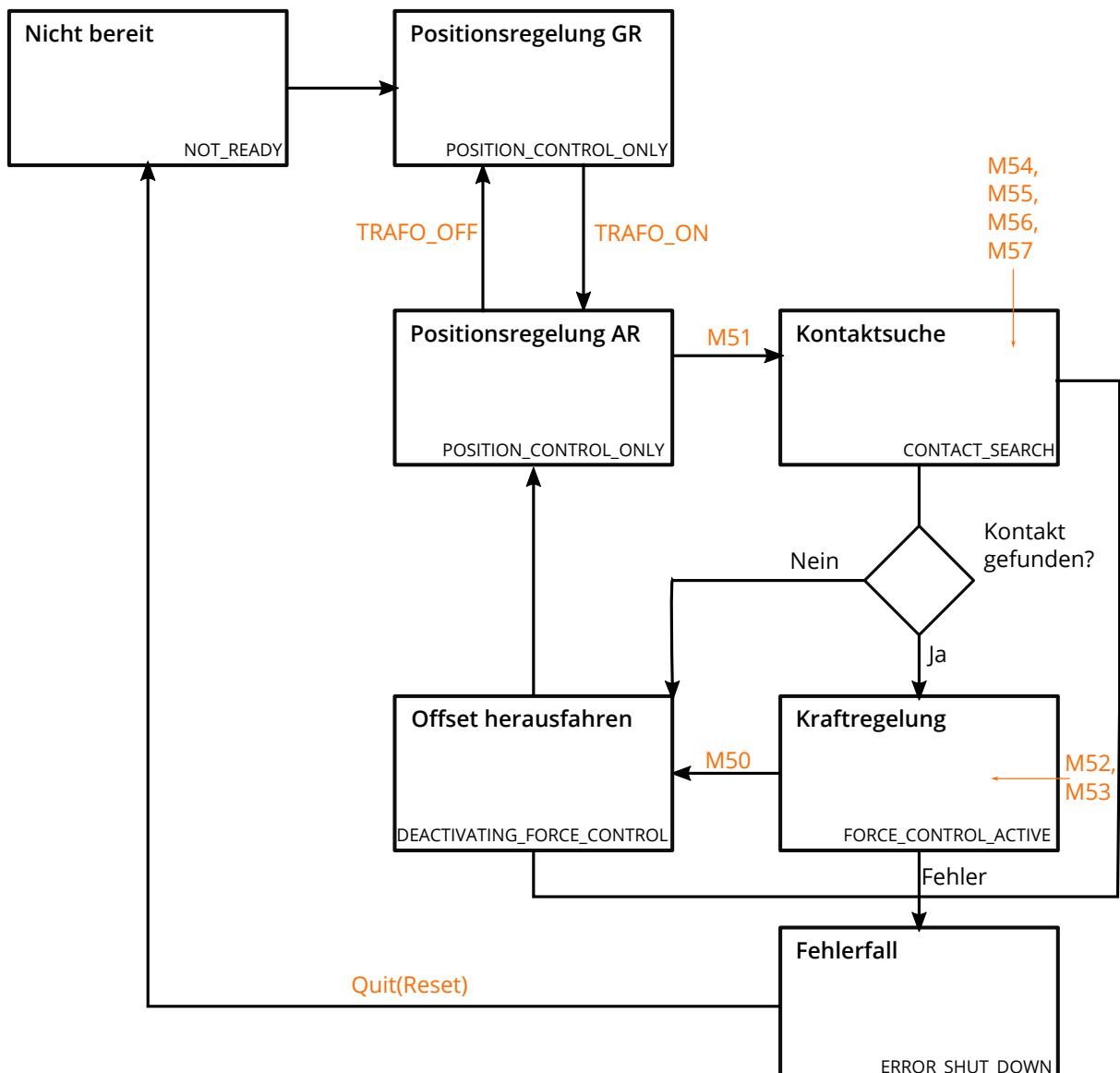
Die Bedienung der Kraftregelung erfolgt über quittierungspflichtige M-Befehle (siehe Tabelle 4.1). Der *MCommandDispatcher* kümmert sich darum, die Flanken der Eingangssignale zu verarbeiten und die entsprechenden Aktionen in der Zustandsmaschine aus Abschnitt 5.4.2 auszulösen. Dadurch wird eine Entkoppelung der M-Befehlbelegung von der eigentlichen Funktionalität erreicht. Infolge dessen kann die Zustandsmaschine die Befehle nach Beendigung selbst quittieren.

### 5.4.2. C++-Submodul StateMaschine

Die Bewegung des Hexapods kann in drei verschiedenen Kontexten geschehen:

1. Kein Kontakt: Hindernisse treten nicht auf und müssen nicht berücksichtigt werden.
2. Kontakt möglich: Es besteht noch kein Kontakt, es ist jedoch in der aktuellen Bewegungsrichtung mit einem Kontakt zu rechnen.
3. Ausgeprägter Kontakt: Es tritt eine Kontaktkraft zur Umgebung auf.

Laut Anforderung SF-2 (kein Stellgrößensprung) muss die Steuerung den Kontakt zur Umwelt kontrolliert aufbauen. Anforderung SF-6 (Kein Kontakt nach Abschalten) verlangt, dass die Maschine nach Deaktivierung der Kraftregelung nicht mehr im Kontakt zur Umgebung steht. Um ein Springen der Hexapodpose bei Ein- und Ausschalten der Kraftregelung sowie im Fehlerfall zu vermeiden, ist deswegen eine Zustandsmaschine notwendig, die das Weginkrement der Kraftregelung kontrolliert auf- und abbaut. In der Klasse *StateMachine* ist diese Funktionalität umgesetzt. Die Zustandsmaschine basiert auf der Bibliothek *TinyFSM* der Digital Integrity GmbH [62]. Das dazugehörige Ablaufdiagramm ist in Abbildung 5.7 dargestellt. Die Zustände sind im Anschluss erläutert.

Abbildung 5.7.: Zustände der Klasse *StateMachine*, Eingangssignale in Orange

### Zustand Positionsregelung AR/GR:

In diesem Zustand werden die Positionssollwerte der CNC  $q_{\text{CNC}}$  unverändert an die Antriebe weitergeleitet. Dabei befindet sich die Positionsregelung entweder im Modus *kartesischer Arbeitsraum* (AR) oder im Modus *Gelenkraum* (GR). Im kartesischen Raum darf die Kraftregelung aktiviert werden (siehe auch Abbildung 5.8), andernfalls ist das nicht zulässig. Der Benutzer kann per G-Codebefehl #TRAFO ON zwischen den beiden Modi umschalten. Die SPS (siehe Abschnitt 5.3) erkennt eine Änderung und setzt die Zustandsmaschine über den *IoHandler* davon in Kenntnis. Wenn das Signal zur Aktivierung der Kraftregelung durch den M-Befehl M51 erfolgt, wird geprüft, ob alle dafür notwendigen Bedingungen erfüllt sind. Es ist erforderlich, dass

- die Positionsregelung im kartesischen Raum stattfindet,
- alle vorherigen M-Befehle quittiert und zurückgesetzt wurden,

- sich die Endeffektorpose im freigegebenen Arbeitsraumanteil befindet und
- die Kraftregelung richtig initialisiert wurde, sowie keinen Offset aufweist.

Wenn eine der Bedingungen nicht erfüllt ist, verbleibt die Maschine in Positionsregelung. Das NC-Programm wird mit einer Fehlermeldung beendet.

### Zustand Kontaktsuche:

Zu Beginn der Kraftregelung muss sichergestellt werden, dass sich der TCP in kontrolliertem Kontakt mit der Umwelt befindet. Weil nicht davon ausgegangen werden kann, dass das der Fall ist, muss dieser Kontakt kontrolliert hergestellt werden bevor die eigentliche Kraftregelung stattfindet. Das entspricht einer Bewegung mit möglichem Kontakt. Die Kontaktsuche ist in der Klasse *ForceController* umgesetzt und wird von der Zustandsmaschine aufgerufen. Details zu der Umsetzung finden sich in Abschnitt 5.4.4. Die Aufgabe der Zustandsmaschine beschränkt sich auf die Ansteuerung des *ForceControllers* und die Verwaltung der Rückgabewerte. Das Zusammenspiel der Submodule ist in Abbildung 5.8 dargestellt.

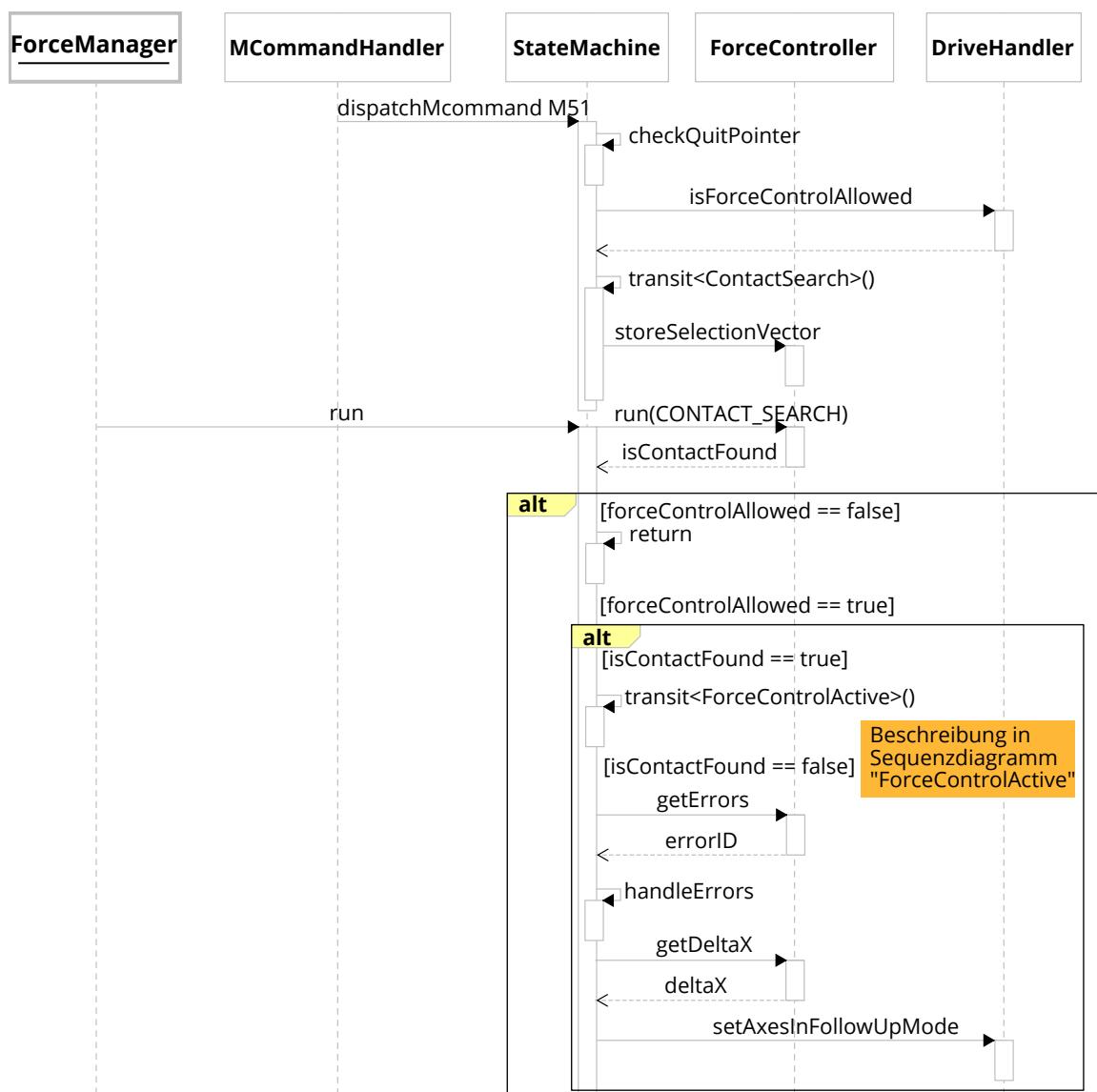


Abbildung 5.8.: Zur besseren Übersichtlichkeit leicht gekürztes UML-Sequenzdiagramm der Kontaktsuche

Bei Eintritt in die Kontaktsuche durch M-Befehl M51 übernimmt die Zustandsmaschine dafür den Kraftselektionsvektor  $S$  aus dem NC-Programm, der festlegt, in welche Richtungen eine Kraftregelung erfolgen soll. Änderungen dieses Vektors ab dem Punkt sind nicht mehr wirksam. Der Kraftselektionsvektor ist vorzeichenbehaftet, um in der Kontaktsuche festzulegen, in welche Richtung verfahren wird. Der Kraftselektionsvektor für eine Kontaktsuche in -Z-Richtung sieht damit beispielsweise wie folgt aus:

$$S = \begin{pmatrix} 0 & 0 & -1 & 0 & 0 & 0 \end{pmatrix}^T \quad (5.1)$$

Falls der Kontakt nicht gefunden wurde, schaltet die Zustandsmaschine automatisch in den Zustand *Offset herausfahren*; Wenn unerwartete Kräfte auftreten, in den Zustand *Nothalt*. Sobald der Kontakt hergestellt ist, wird die Kraftregelung aktiviert und der M-Befehl quittiert.

#### Zustand Kraftregelung:

Wenn der Kontakt zur Umgebung erfolgreich aufgebaut wurde, beginnt die Kraftregelung. Die Aufgabe dieses Zustandes liegt darin, die vorgegebene Sollkraft über ein Weginkrement am System einzustellen. Der dafür zuständige Regler ist in der Klasse *ForceController* implementiert (Abschnitt 5.4.4). Nur in diesem Zustand soll die CNC die aktuell anliegenden Kräfte überwachen. Um das zu veranlassen, setzt die *StateMachine* den *DriveHandler* davon in Kenntnis, welche Kraftachsen für die Kraftregelung aktiviert und im Kontakt sind. Das Zusammenspiel der beteiligten Klassen ist in Abbildung 5.9 dargestellt. Sobald die Kraftregelung aktiv ist, deaktiviert die Zustandsmaschine den Nachführbetrieb der aktiven Kraftachsen (Abschnitt 5.3).

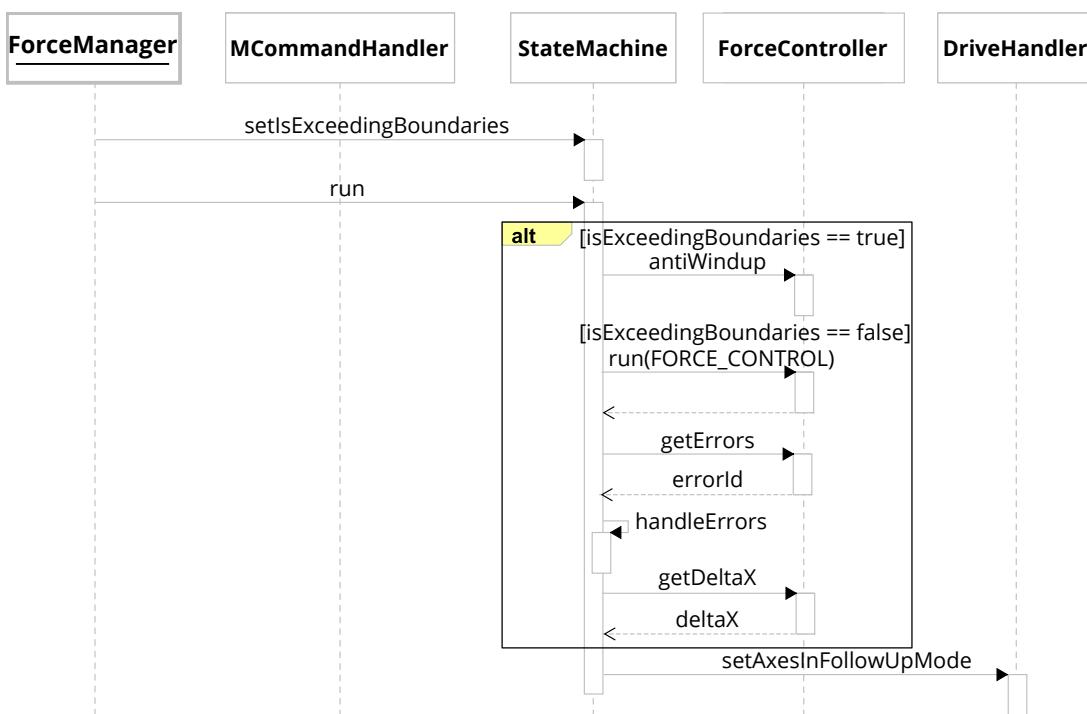


Abbildung 5.9.: Zur besseren Übersichtlichkeit leicht gekürztes UML-Sequenzdiagramm der Kraftregelung

Während der Kraftregelung wird der Kraftselektionsvektor als boolscher Vektor interpretiert: die Vorzeichen spielen keine Rolle mehr. Wenn der Kontakt verloren geht, schaltet die Zustandsmaschine automatisch wieder in die Kontaktsuche. Durch entsprechende M-Befehle (siehe Tabelle 4.1) können voreingestellte Reglerparameter aktiviert werden. Wenn die TCP-Pose den vorgegebenen Rahmen verlässt (siehe Sicherheitsanforderungen in Kapitel 3), hält die Bewegung an und die Zustandsmaschine unterbricht die Kraftregelung, um ein wind-up des Kraftreglers zu verhindern.

#### Zustand Offset herausfahren:

Wenn der Befehl zum Deaktivieren der Kraftregelung erfolgt oder eine Kontaktsuche erfolglos endet, muss der Offset abgebaut werden, damit es nicht zu einer sprunghaften Poseänderung kommt. Das passiert in diesem Zustand, indem die entsprechende Funktion in der Klasse *ForceController* aufgerufen wird. Erst wenn kein Offset mehr existiert, wird wieder in die Positionsregelung geschalten. Abbildung 5.10 zeigt das dazugehörige Sequenzdiagramm.

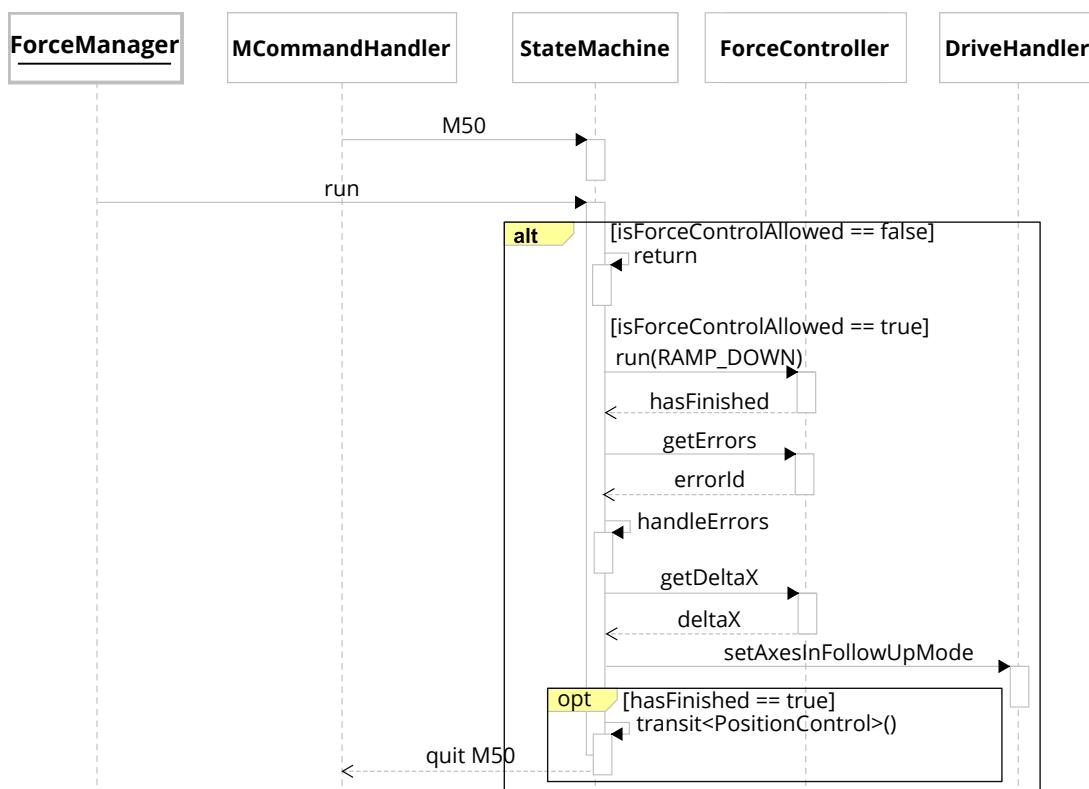


Abbildung 5.10.: Zur besseren Übersichtlichkeit leicht gekürztes UML-Sequenzdiagramm des Zustandes *Offset herausfahren*

#### Zustand Fehlerfall:

Falls ein kritischer Fehler auftritt, hält die Steuerung an. Ein eventuell noch vorhandener Offset wird nicht herausgefahren. Die Zustandsmaschine setzt ein Fehlerbit im *ForceManager*, damit alle betroffenen Submodule auf den kritischen Fehler reagieren können.

### 5.4.3. C++-Submodul DriveHandler

Die Klasse *DriveHandler* implementiert das abstrakte Interface *IoHandler*, das die Interaktion mit Ein- und Ausgabegeräten sowie dem ADS-Layer verwaltet. Wie in Abschnitt 5.1 erläutert, soll die konkrete Implementierung in diesem Lösungsansatz durch eine *man-in-the-middle-Aktion* Zugriff auf den Signalfluss der Steuerung erlangen. Dafür muss der Signalfluss zwischen CNC und Antrieben umgeleitet werden, wie in Abschnitt 5.1 erläutert. Das Modul *ForceManager* ruft den *DriveHandler* zyklisch auf, um CNC und Antriebe wie in Abbildung 5.2 dargestellt, miteinander zu verbinden. Dabei erhält das Modul *ForceManager* die Möglichkeit, durchgeleitete Informationen zu manipulieren. Die Klasse *DriveHandler* kapselt dafür die Low-Level-Implementierung. Durch die Verwendung des Interfaces ist es problemlos möglich, eine andere Strategie zur Beeinflussung der Steuerungsgrößen zur verwenden, ohne die anderen Submodule des *ForceManagers* anzupassen. Die betreffende Kommunikation mit den anderen Submodulen ist in Abbildung 5.5 dargestellt. Im Folgenden sind die wichtigsten Funktionen des *DriveHandler* aufgeführt:

#### Eingänge einlesen

Zu Beginn jedes Taktes lädt der *DriveHandler* sämtliche antriebsbezogenen Daten aus den entsprechenden *TwinCAT*-Modulen, skaliert und stellt sie dem *ForceManager* und seinen Untermodulen bereit. Folgende Informationen stehen nach dem Einlesen zur Verfügung

- Position: Innerhalb der CNC sind Längen in der Einheit 100 nm beschrieben. Die Sollwerte der CNC ( $q_{cnc}$ ) liegen im Gelenkraum vor und werden hier in mm umgerechnet. Danach findet eine Transformation in den kartesischen Arbeitsraum ( ${}^{glob}x_{cnc}$ ) statt. Dieser Schritt ist für die Kraftregelstrategie notwendig, siehe Abschnitt 5.4.4. Hat der Nutzer ein Aufgabenkoordinatensystem programmiert (siehe Abschnitt 4), so transformiert der *DriveHandler*  ${}^{glob}x_{cnc}$  zu  ${}^{usr}x_{cnc}$ . Posen die im Nutzerkoordinatensystem dargestellt sind, weisen im Quelltext die Variablenendung *usr* auf, Posen im Basiskoordinatensystem der Maschine zeichnen sich durch die Endung *glob* aus.
- Analog zu den Sollwerten wandelt der *DriveHandler* die Istwerte aus den Antrieben ( $q_{act}$ ) in den kartesischen Raum ( ${}^{glob}x_{act}$ ) um und transformiert sie in den Nutzerraum ( ${}^{usr}x_{act}$ ). Diese Berechnungen sind sehr aufwendig und müssen eventuell für Anwendungen, in denen sie nicht benötigt werden, deaktiviert werden.
- Kraft: Weil die Sollkraft aus der CNC stammt und dort als Achse behandelt wird, ist der *DriveHandler* dafür verantwortlich, die Werte dafür auszulesen und dem *ForceController* bereitzustellen. Die Umwandlung von 100 nN in 1 N findet hier analog zu den Positions-sollwerten statt. Die Nutzertransformation der Kräfte findet im *ForceController* statt.
  - Override: Der Bediener soll nach Anforderung FA-9 (Regelgeschwindigkeitskontrolle über Override) die Möglichkeit erhalten, zu jedem Zeitpunkt die Geschwindigkeit des Hexapods zu beeinflussen. Dafür ist es notwendig, die Kraftregelung an den Override-Wert zu

koppeln. In der CNC liegt der Wertebereich des Potentiometers bei 0..10000. Der *DriveHandler* skaliert die Eingabe auf 0..1.

- Benutzerdefinierte Koordinatensysteme: Der Benutzer erhält, wie in Abschnitt 4 erläutert, die Möglichkeit, im G-Code eigene Koordinatensysteme zu definieren. Das Transformationsmodul schreibt die dafür notwendigen Kanalparameter in einen gemeinsamen Speicher. Der *DriveHandler* bereitet diese Daten auf, sodass sie für die Transformation der Istwerte und im *ForceController* verwendet werden können.
- Überprüfung der Istposition: Zu Beginn jeden Taktes erstellt der *DriveHandler* einen Status aus der aktuellen Istpose, der bestimmt, ob eine Kraftregelung zulässig ist. Dafür wird geprüft ob die Pose innerhalb des Sicherheitsraumes der Kraftregelung liegt (siehe Anforderungen SF-1) und ob die Trafo online erfolgt. Der Nutzer kann die Grenzen des für die Kraftregelung zugänglichen Arbeitsraum im G-Code programmieren (siehe Tabelle 4.1). Die Prüfung der Istpose findet im Nutzerkoordinatensystem statt. Nur wenn alle Bedingungen erfüllt sind, kann in Kraftregelung geschalten werden.
- Reglerparameter: In Abschnitt 4 ist beschrieben, wie der Benutzer im G-Code zwei Reglerparametersets definieren und dann zwischen ihnen hin- und herschalten kann. Abschnitt 5.2 beschreibt, dass die Kanalparameter in einem gemeinsamen Speicherbereich abgelegt werden. Klassen, die vom *IoHandler* ableiten, implementieren eine Funktion, die aktuell angewählten Parameter dort ausliest und dem *ForceController* bereitstellt.

## Offset aufschalten

Der *ForceController* generiert ein Lageoffset  $\Delta x$ , der auf den Sollwert der CNC  $x_{cnc}$  aufzuschalten ist. Das geschieht hier im *DriveHandler*. Daraus resultiert die Stellgröße  $x_{cmd}$ . Die Orientierung von  $\Delta x$  liegt als Eulerwinkel vor. Weil diese durch mitgedrehte Achsen definiert sind, kann die Orientierungsänderung nicht direkt zu  $x_{cnc}$  addiert werden (siehe auch Abschnitt 2.1.1). Die Orientierungsdifferenz muss zuerst in eine Rotationsmatrix umgeformt werden. Bereits in der Vorgängerarbeit wurde dafür ein geeignetes Verfahren ausgewählt [6]. Somit wird sichergestellt, dass  $x_{cnc}$  von der gewählten Winkeldefinition unabhängig bleibt.

$$x_{cmd} = x_{cnc} \cdot T_{\Delta x}, \text{ wobei} \quad (5.2)$$

$$T_{\Delta x} = \begin{pmatrix} R_{\Delta x} & o_{\Delta x} \\ 0^T & 1 \end{pmatrix} \quad (5.3)$$

Durch die Entkoppelung des *ForceControllers* vom *DriveHandler* durch das Interface *IoHandler* kann von der Kraftregelung unabhängig entschieden werden, ob der Offset mit der *vollständig inversen Kinematik* oder mit der *inversen Jacobimatrix* in den Gelenakraum überführt werden soll. Vor- und Nachteile beider Regelstrategien sind in Abschnitt 2.2.3 beschrieben. In dieser Arbeit findet der Ansatz der vollständig inversen Kinematik Anwendung. Für diese Regelstrategie muss  $\Delta x$  im Nutzerraum mit  $x_{cnc}$  kombiniert werden. Der *DriveHandler* stellt deswegen si-

cher, das alle Posen zuvor in das Nutzerkoordinatensystem transformiert sind. Abbildung 5.11 zeigt den Informationsfluss und die dazugehörigen Transformationen. Abschließend wird im Basiskoordinatensystem geprüft, ob der Hexapod die Pose anfahren kann.

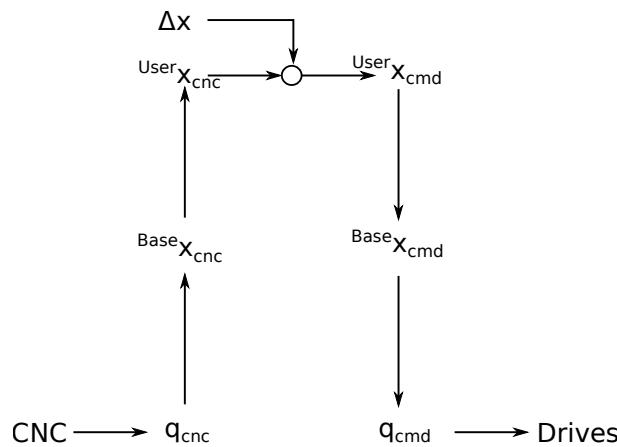


Abbildung 5.11.: Transformationen für das Aufschalten des Poseoffsets der Kraftregelung auf die Positionssollwerte

### Ausgänge schreiben

Am Ende jeden Taktes beschreibt die Klasse *DriveHandler* die antriebsrelevanten Ausgänge des Moduls *ForceManager*. Das umfasst zum einen die manipulierten Antriebssollwerte  $x_{cmd}$ , die vor dem Aufschalten in den Gelenkraum transformiert und auf 100 nm skaliert werden. Zum anderen verbindet der *DriveHandler* die im Rahmen der *man-in-the-middle-Aktion* zwischen den Antrieben und den dazugehörigen CNC-Achsen aufgetrennten Verknüpfungen. Die eingehenden Signale sind in Tabelle 5.1, die ausgehenden Verbindungen in Tabelle 5.2 dargestellt.

Übersicht über die im Rahmen der *Man-in-the-middle-Aktion* eingelesenen Daten

Quelle	Name	Einheit	Bemerkung
Antriebe	Statuswort	-	
	Sercosphase	-	
	Istposition	mm	Gelenkraum
	Istgeschwindigkeit	mm/s	Gelenkraum
	Istmoment	Nm	Gelenkraum
CNC	Steuerwort	-	
	Sollpose	mm	Gelenkraum
	Sollkräfte und -momente	N, Nm	kartesischer Raum
C++-Modul HexForce	Istkräfte und -momente	N, Nm	kartesischer Raum

Übersicht über die im Rahmen der *Man-in-the-middle-Aktion* geschriebenen Daten

Ziel	Name	Einheit	Bemerkung
CNC	Statuswort	-	
	Sercosphase	-	
	Istposition	mm	Gelenkraum
	Istgeschwindigkeit	mm/s	Gelenkraum
	Istmoment	Nm	Gelenkraum
	Istkräfte und -momente	N, Nm	kartesischer Raum
Antriebe	Steuerwort	-	
	Sollpose	mm	Gelenkraum, durch Kraftregelung modifiziert
SPS	Nachführbetrieb	-	Steuerbitarray

Hat die Zustandsmaschine im Hauptmodul einen kritischen Fehlerfall gemeldet, schreibt der *DriveHandler* keine neuen Sollpositionen auf die Antriebe. Dadurch bleibt der Endeffektor stehen.

#### 5.4.4. C++-Submodul ForceController

Der *ForceController* ist dafür verantwortlich, die Kraftregelung durchzuführen. Dazu wird er zyklisch vom C++-Modul *StateMachine* aufgerufen, sofern die Kraftregelung erlaubt und aktiviert ist. Er generiert den Poseoffset  $\Delta x$ , die Stellgröße der positionsbasierten Kraftregelung. Der *IoHandler* ist dafür verantwortlich, den Offset auf die Sollwerte aufzuschalten. Analog zur Klasse *DriveHandler* ist die Funktionsstruktur des *ForceControllers* während eines Zyklus in drei Bereiche gegliedert: Eingänge lesen, Daten verarbeiten und Ausgänge schreiben.

##### Eingänge einlesen

Am Zyklusanfang bezieht der *ForceController* kraftregelungsrelevante Daten und verarbeitet sie. Darunter fallen:

- Istkräfte: Es stehen verschiedene Kraftsensoren zur Verfügung, die während der Laufzeit ausgewählt werden können. Abschnitt 2.3.1 enthält eine Übersicht der wählbaren Istwertquellen. Auf Basis dessen stellt das C++-Modul *HexForce* in jedem Takt die aktuell im Koordinatensystem  $P$  der Hexapodplattform angreifenden Kontaktkräfte zur Verfügung. Um diese für die Kraftregelung nutzbar zu machen, transformiert der *ForceController* die Istwerte zuerst in den *Task Space*. In diesem Abschnitt fällt der *Task Space* mit dem vom Nutzer vorgegebenen Koordinatensystem zusammen. In Abschnitt 7.3.1 findet eine Erweiterung der *Task Space*-Definition statt. Die Kräfte werden auf den TCP transformiert.

Die Transformation ist in Abbildung 5.12 dargestellt, während die berücksichtigten Koordinatensysteme in Abbildung 4.1 visualisiert sind. Der *ForceController* bietet dem Nutzer die Möglichkeit Ist- und Sollkräfte mit einem PT1-Filter zu filtern, wie in Abbildung 5.12 dargestellt.

- Sollkräfte: Der *IoHandler* stellt dem *ForceController* die CNC-generierten Kraftsollwerte in N bereit. Im Gegensatz zu den Istkräften müssen die Sollwerte nicht in das Nutzerkoordinatensystem transformiert werden, weil anzunehmen ist, dass der Nutzer die Werte in Bezug auf das Nutzerkoordinatensystem programmiert hat.
- Positionsfreigabe: Der *IoHandler* prüft die in Abschnitt 3 definierten Sicherheitskriterien und stellt sie dem *ForceController* als Information über den momentanen Status bereit. Der *ForceController* prüft die Freigabe ab, bevor die Kraftregelung ausgeführt wird.

### Daten verarbeiten

Die Hauptfunktion des *ForceControllers* liegt in der Berechnung des Poseoffsets. Je nach Zustand des Systems (siehe dazu Abschnitt 5.4.2) ist dafür eine andere Aktion nötig. Die Zustandsmaschine ruft die Funktion zur Offsetberechnung deshalb mit verschiedenen Kontexten auf. Im Folgenden wird erläutert, was im jeweiligen Fall geschieht.

- a) Kontaktsuche: Die Kontaktsuche ist als Geschwindigkeitsregelung angelegt. Es wird in alle per Kraftselektionsvektor angewählten Richtungen, in denen noch kein Kontakt vorliegt, pro Takt ein Weginkrement aufgeschalten, das mit dem Override skaliert ist. Ein Kontakt gilt dann als gefunden, wenn in allen angewählten Richtungen eine Kontaktkraft anliegt, die die vorgegebene Schwellkraft überschreitet. Falls eine vom Benutzer im NC-Code vorgegebene Strecke überschritten wurde, ohne dass ein Kontakt auftrat, gilt die Kontaktsuche als fehlgeschlagen und wird abgebrochen. Der *ForceController* signalisiert der Zustandsmaschine daraufhin, dass die Kontaktsuche nicht erfolgreich war. Sie veranlasst dann, dass der aufgebaute Offset wieder herausgefahrt wird.

Tabelle 5.3.: Standardwerte der Schwellkräfte und -momente für die Kontaktsuche

x	y	z	a	b	c
20 N	20 N	20 N	0,8 Nm	0,8 Nm	0,8 Nm

Wenn in einer inaktiven Richtung eine Kraft auftritt, die ein gesetztes Limit übertritt, dann erfolgt ein Signal an die Zustandsmaschine, sofort in den Nothalt zu schalten.

Tabelle 5.4.: Standardwerte der Kraft- und Momentenlimits

x	y	z	a	b	c
600 N	600 N	600 N	100 Nm	100 Nm	100 Nm

- b) Kraftregelung: Ist der Kontakt zur Umwelt hergestellt, beginnt die eigentliche Kraftregelung. Sollkraft  ${}^T S F_{nom}$  und Istkraft  ${}^T S F_{act}$  befinden sich im *Task Space*. Abbildung 5.12 illustriert den Signalfluss inklusive der vorangegangenen Transformationen. Die Differenz

$${}^T S F_{lag} = diag(S) \cdot ({}^T S F_{nom} - {}^T S F_{act}) \cdot Ovr \quad (5.4)$$

wird mit dem Override  $Ovr$  skaliert. Damit kann der Benutzer die Geschwindigkeit der Regelung anpassen und gegebenenfalls auf Null setzen. Der Kraftselektionsvektor  $S$  wird vorzeichenbereinigt. Ein PI-Regler übernimmt die kartesische Kraftregelung.

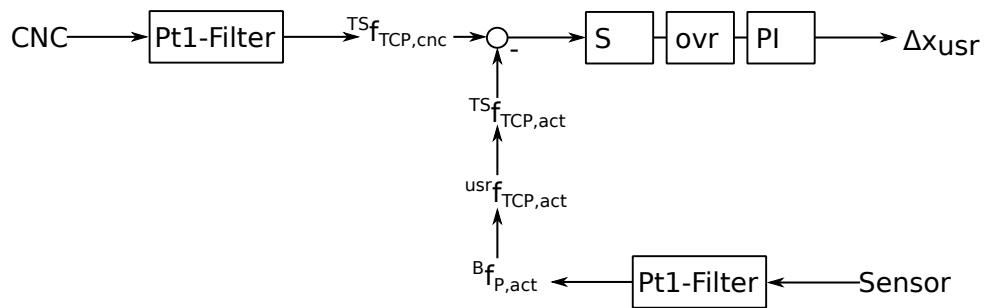


Abbildung 5.12.: Signalfluss und Transformationen im *ForceController*

Dabei kann per M-Befehl aus zwei voreingestellten Reglerparametersets gewählt werden. Wenn Kräfte in Richtung deaktivierter Koordinatenachsen auftreten, wird ein Nothalt veranlasst. Die dafür zugrunde liegenden Kraft- und Momentenlimits sind in Tabelle 5.4 aufgeführt.

- c) Offset abbauen: Analog zur Kontaktsuche wird angewiesen, den Offset auch wieder geschwindigkeitsgeregt abzubauen. Die Kräfte werden dabei überwacht, denn es darf in keiner Richtung zu einem Anstieg der Kraft kommen. Bei Verletzung dieser Bedingung schaltet die Zustandsmaschine sofort in den Nothalt.

### Ausgänge schreiben

Anders als beim *DriveHandler* sind an die Positionsachsen der Kraftregelung keine realen Antriebe gekoppelt. Der Offset selbst wird nicht direkt auf die Antriebe geschalten. Stattdessen übernimmt die Zustandsmaschine den Offset und leitet ihn an den *ForceManager* weiter. Dieser veranlasst dann das Aufschalten im *DriveHandler*, sowie in Abbildung 5.6 dargestellt. Außerdem übergibt der *ForceController* die in den *Task Space* transformierten Kräfte in jedem Takt an den *IoHandler*, der die Rückführung zur CNC übernimmt.

## 5.5. Fazit

In diesem Kapitel konnte gezeigt werden, dass die Umsetzung einer Kraftregelung innerhalb *TwinCATs* möglich ist. Dazu erfolgten erst Modifikationen des bestehenden Systems, um die dafür notwendige Infrastruktur zu schaffen. Anschließend fand die Entwicklung eines Kraftreglers statt, der kontrolliert steuerbar ist. Die Implementierung setzt das Bedienkonzept aus Kapitel 4 um. Zunächst ist die Kraftregelung nur entlang der globalen kartesischen Achsen des benutzerdefinierten Aufgabenkoordinatensystems möglich. Im folgenden wird die entwickelte Software erprobt. In Abschnitt 6 erfolgt dafür die Inbetriebnahme und in Abschnitt 7 die experimentelle Validierung der Regelung.

# 6. Inbetriebnahme

## Überblick

Nachdem sich Abschnitt 5 mit der Struktur und Umsetzung der implementierten Kraftregelung beschäftigt, gilt es nun, die Funktionalität zu erproben. Dieser Abschnitt beschreibt die Inbetriebnahme der Kraftregelung am Hexapod. Es soll gezeigt werden, dass sich die Steuerung wie vorhergesehen verhält. Dazu findet erst eine virtuelle Inbetriebnahme statt, um das Risiko für Mensch und Maschine zu minimieren. Anschließend wird die Steuerungssoftware auf die reale Maschine übertragen. Im Rahmen dessen finden kleine Anpassungen am System statt, um die Durchführung der sich diesem Abschnitt anschließenden Experimente zu ermöglichen.

### 6.1. Virtuelle Inbetriebnahme

Die Programmierung einer Kraftregelung ist durch den zwingend notwendigen Kontakt zur Umwelt immer mit Risiken behaftet. Durch die Entscheidung, die Stellgrößen der Kraftregelung ohne Bahnplanung auf die Antriebe zu schalten, ist noch stärker Vorsicht geboten. Um die Gefahr für Mensch und Maschine so gering wie möglich zu halten, erfolgt die Implementierung der Software unter Einhaltung des in Abschnitt 3 definierten Sicherheitskonzeptes. Um in jeder Phase der Entwicklung zu gewährleisten, dass keine Verletzung der Sicherheitsanforderungen auftreten, sind automatisierte Tests in den Entwicklungsprozess eingebunden. Bei Änderungen am Quelltext wird dadurch in den meisten Fällen sofort ersichtlich, ob ein sicherheitskritischer Bug entstanden ist, bevor die Software auf die Maschinensteuerung gelangt. Außerdem stellen die Tests eine sehr zuverlässige Lowlevel-Dokumentation dar, weil sie bei Obsoleszenz Fehler melden [63]. Die Tests zur Software können auf der beigelegten DVD eingesehen werden. Zur besseren Verständlichkeit sind im weiteren Verlauf der Arbeit daraus Auszüge zu finden. Das verwendete Testframework heißt *GoogleTest* und wird von Google für C++ entwickelt. Weil es nicht möglich ist, den zyklisch ausgeführten Quelltext innerhalb *TwinCATs* zu testen, geschieht das mit Offlinetests außerhalb *TwinCATs*. Die Testsuite ist mit dem Programm *Qt-Creator* geschrieben. Weil viele Funktionen des *ForceManagers* auf die Daten anderer Steuerungskomponenten oder der Maschinenumwelt angewiesen sind, werden alle

benötigten Informationen und Schnittstelleninteraktionen in einem Minimalmodell der Steuerung und Umgebung simuliert. Mit dem Befehl *MockCyclic.run()* durchläuft das virtuelle Modell dafür die in Abbildung 6.1 gezeigten Zustände des zyklischen Programmteils. Abbildung 6.2 zeigt den Signalfluss während der Simulation.

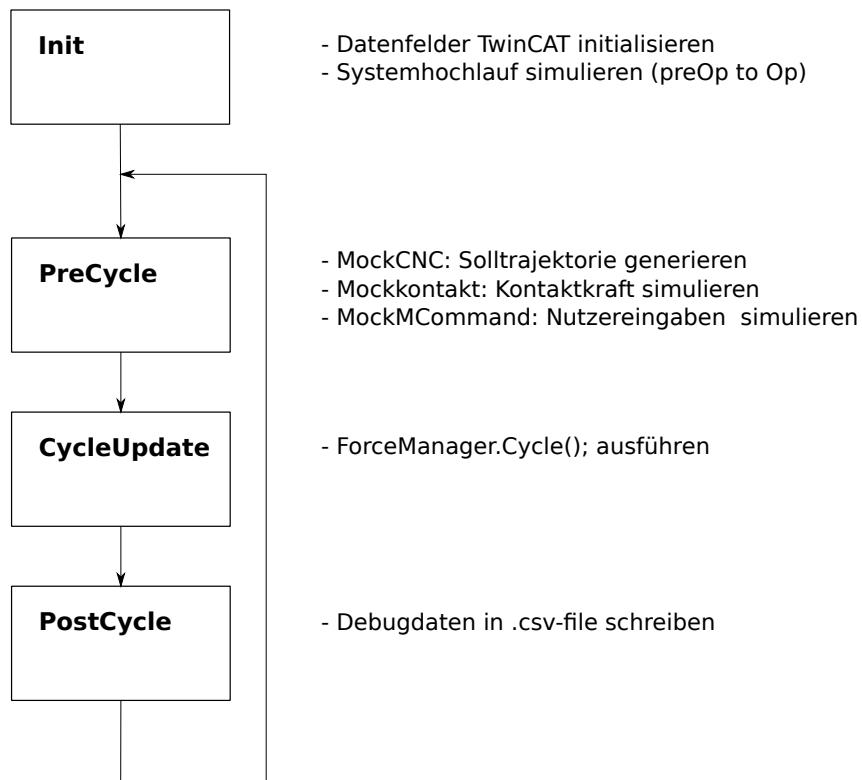


Abbildung 6.1.: Virtuelles Prozess- und Steuerungsmodell

Das C++-Modul *ForceManager* und seine Submodule werden durch das Modell über die selben Schnittstellen bedient, wie später in *TwinCAT*. Das virtuelle Modell simuliert folgende Komponenten des realen Systems:

- MockCNC: Über diesen Baustein wird die Bahnerzeugung der CNC simuliert. Standardmäßig stehen eine Gerade und ein Kreis in der x-y-Ebene zur Verfügung. Der Benutzer kann aber auch weitere Trajektorien vorgeben. Die Vorgabe kann gezielt aktiviert bzw. abgeschaltet werden. Folgendes Quelltextbeispiel illustriert, wie ein Testszenario mit individueller Bahn erstellt wird:

Quelltext 6.1: Vorgabe einer benutzerdefinierten Trajektorie

```

// generate a sine trajectory in x
Class SinusX : public FSMockTrajectory{
public:
    CFPose run(double cycleTime)override{
        CFPose result (sin(t_local*2*pi), 0., 0., 0., 0., 0.);
        t_local += cycleTime;
        return result;
    }
}

```

```

    }

TEST_F(Test_Trajectory, sineX){
    SinusX trajectory;
    mockCyclic.setPositionTrajectory(&trajectory);
    // necessary flag to activate the generation of a fake trajectory
    mockCyclic.shouldGeneratePositionTrajectory = true;
    mockCyclic.run();
}

```

Diese Funktionalität kann auch genutzt werden, um Kraftsollwerte mit individuellem Verlauf vorzugeben. Dafür sind lediglich folgende Befehle anzupassen:

Quelltext 6.2: Vorgabe eines benutzerdefinierten Sollkraftverlaufs

```

mockCyclic.setPositionTrajectory(&trajectory);
mockCyclic.shouldGeneratePositionTrajectory = true;

```

- MockContact: Um verschiedene Kontaktscenarien zu testen, besteht die Möglichkeit, in jedem der sechs verschiedenen Freiheitsgrade eine Kraft zu generieren. Damit können Szenarien verschiedener Komplexität modelliert werden, wie zum Beispiel virtuelle Gegenstände, an denen die Kraftregelung ausprobiert werden kann. Der Standardmechanismus für das Erzeugen der Kräfte ist dabei ein Federmodell auf Basis der Endeffektorposition und einer poseabhängigen Kontaktsteifigkeit, die ein Kraftpotential darstellt, wie in Abbildung 6.3 illustriert. Perspektivisch können aber auch andere Effekte berücksichtigt werden. Jeder Freiheitsgrad lässt sich über ein Aktivierungsflag an- bzw. abwählen. Für die Erprobung dieser Arbeit ist ein mit der z-Achse koaxialer Zylinder modelliert. Das folgende Beispiel illustriert, wie das Modell konfiguriert wird, sodass es bei Kontakt des Endeffektors mit der Mantelfläche eine von x- und y-Position abhängige Kraft an die Sensorschnittstelle der Kraftregelung zurückmeldet.

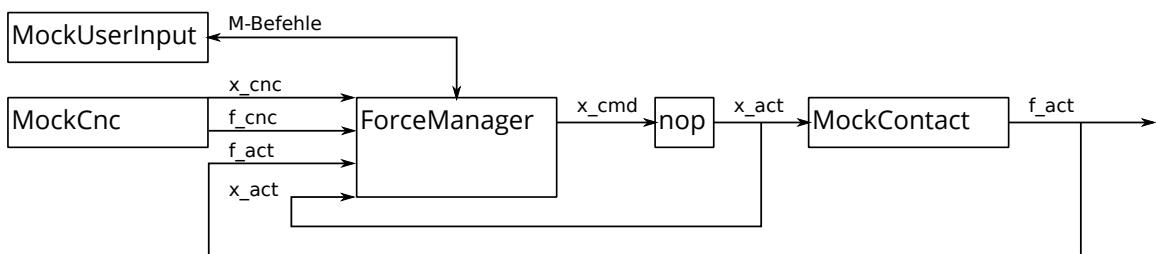


Abbildung 6.2.: Virtuelles Prozess- und Steuerungsmodell

```

// activate the preconfigured cylinder in x & y
mockCyclic.setForceDummies({1, 1, 0, 0, 0, 0});

// set up Trajectory into the obstacle
LinearTrajectory trajectory;
double processTime = 1.;
trajectory.setStart(-130, -100); // x, y in mm
trajectory.setEnd(-50, -100); // x, y in mm
trajectory.setProcessTime(processTime); // s
mockCyclic.setTrajectory(&trajectory);
// activate the generation of the fake trajectory
mockCyclic.shouldGenerateTrajectory = true;

// generate position and force for one cycle
mockCyclic.run();
fx = m_inputs.dataFromIO.forceSensors.platform.Fx;
fy = m_inputs.dataFromIO.forceSensors.platform.Fy;
f_Kontakt = sqrt(fx*fx + fy*fy)

```

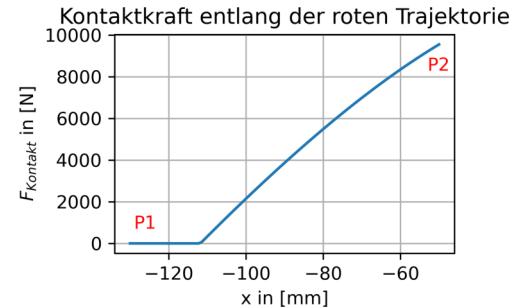
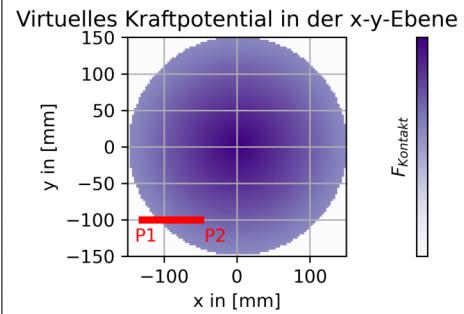


Abbildung 6.3.: Beispiel zur Erzeugung einer simulierten Kraft in der x-y-Ebene

- MockUserInput: M-Befehle aus dem G-Code können hiermit samt der Quittierung simuliert werden. Bei der Vorgabe muss die Zyklusnummer angegeben werden, zu welcher der M-Befehl aktiv sein soll. In jedem Takt, in dem ein quittierpflichtiger M-Befehl nicht quittiert aber aktiv ist, wird die Zyklusnummer aller anderen M-Befehle inkrementiert. Dadurch wird immer nur ein M-Befehl gleichzeitig abgearbeitet: die Simulation weist somit das gleiche Verhalten auf wie die CNC. M-Befehle mit der Zyklusnummer -1 führt das Modell im nächsten freien Takt aus. Folgendes Beispiel illustriert, wie M-Befehle erzeugt werden können:

Quelltext 6.3: Simulation von M-Befehlen

```

MCommand sendM51(0, 51);
MCommand sendM50(1, 50);
// overrides all Mx with the list specified
mockMCommandHandler.setMCommands({sendM51, sendM50});

// runs the model for two cycles
mockCyclic.run(2); // -> gets M51 in first cycle, M50 in second if M51 has been quit

MCommand sendM53(-1, 53);
// adds the specified Mx to the list
mockMCommandHandler.setMCommands(sendM53);
mockCyclic.run(); // -> immediate execution after m50 quit

```

Mit dem virtuellen Modell kann überprüft werden, ob die Implementierung der Kraftregelung Logikfehler aufweist. Außerdem kann das Parametrieren der Regelung aus dem ADS, die Quittierung erfolgreich umgesetzter M-Befehle und die Interaktion mit den restlichen *TwinCAT*-

Interfaces getestet werden. Es ist nicht beabsichtigt, das Verhalten der realen Maschine mit der hohen Präzision eines Systemmodells abzubilden.

Mit folgendem virtuellen Testprozess soll nach Abschluss der Programmierarbeiten entschieden werden, ob die Software sich in der Testumgebung wie erwartet verhält und damit auf die Maschine übertragen werden kann. Während die meisten Tests der Suite unit und integration tests sind, übernimmt dieser Test die Funktion eines system tests [64]. Zu Beginn des virtuellen Experiments wird der Endeffektor in geringem Abstand in z-Richtung über dem Zylinder positioniert. Durch einen M-Befehl ausgelöst beginnt die Kontaktsuche in negativer z-Richtung. Nach gefundenem Kontakt soll eine Kontaktkraft zyklisch von 220 N bis 370 N auf und abgebaut werden. Der virtuelle Kontakt findet zwischen Plattform und Zylinder statt.

Quelltext 6.4: Virtuelle Kraftregelung in z-Richtung mit Kontaktsuche

```
TEST_F(Test_ForceManager, forceControlExactForceInZ){
    CFForce6D f_init(0., 0., 370., 0., 0., 0.);
    CFForce6D f1(0., 0., 220., 0., 0., 0.);
    CFForce6D f2(0., 0., 370., 0., 0., 0.);
    setInputSelectionMatrix(m_data, CFPose(0., 0., -1., 0., 0., 0.));
    mockCyclic.setForceDummies({0, 0, 1, 0, 0, 0});
    switchOnContactSearch();
    // configure force commands for building up the initial force
    LinearTrajectory buildUpInitialForce;
    buildUpInitialForce.setStart(CFForce6D());
    buildUpInitialForce.setEnd(f_init);
    buildUpInitialForce.setProcessTime(5.);
    // configure force commands for ramping the force down to f1
    LinearTrajectory buildUpF1;
    buildUpF1.setStart(f_init);
    buildUpF1.setEnd(f1);
    buildUpF1.setProcessTime(5.);
    // configure force commands for ramping the force up to f2
    LinearTrajectory buildUpF2;
    buildUpF2.setStart(f1);
    buildUpF2.setEnd(f2);
    buildUpF2.setProcessTime(5.);

    // activate the force command input
    mockCyclic.shouldGenerateForceTrajectory = true;
    mockCyclic.setForceTrajectory(&buildUpInitialForce);
    while( ! buildUpInitialForce.isDone())
        mockCyclic.run();
    mockCyclic.setForceTrajectory(&buildUpF1);
    while( ! buildUpF1.isDone())
        mockCyclic.run();
    mockCyclic.setForceTrajectory(&buildUpF2);
    while( ! buildUpF2.isDone())
        mockCyclic.run();

    // Check if force as been attained
}
```

```

CFMatrix f_result = fm.forceController.getF_act_usr();
EXPECT_FLOATS_NEARLY_EQ(matrixToFloat(f_result), matrixToFloat(CFMatrix(m_inputs.
    dataFromCNC.forceData.f_nom, 6)&0.0001), 0.3);
EXPECT_EQ(fm.getState(), FORCE_CONTROL_ACTIV);
}

```

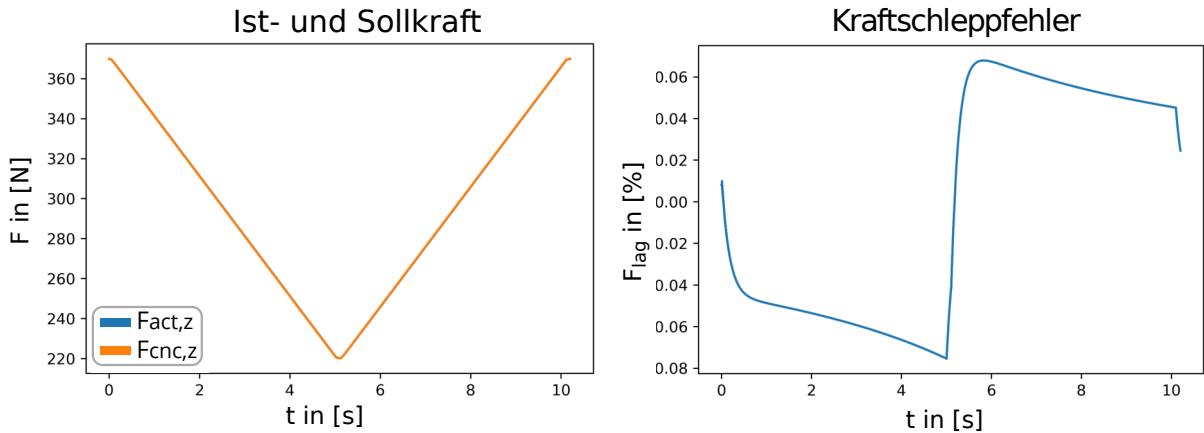


Abbildung 6.4.: virtuelles Experiment, Links: Kraftvorgabe und -istwert, Rechts: relative Regelabweichung

## 6.2. Reale Inbetriebnahme

Nachdem die virtuelle Kraftregelung inklusive der Kontaktsuche erfolgreich verlaufen ist, kann die Software nun auf das reale System übertragen werden. Das Ziel der realen Inbetriebnahme besteht darin, die Simulationsergebnisse zu validieren und für das reale System passende Reglerparameter zu finden. Außerdem gilt es, die Auswirkung nicht in dem Modell berücksichtigter Einflüsse zu quantifizieren. Analog zum virtuellen Versuch soll eine variable Kraft in negativer z-Richtung eingestellt werden. Für die Kraftmessung verwendet der Hexapod während der gesamten Arbeit, sofern nicht anders spezifiziert, die strukturintegrierten Sensoren der Plattform, die in Abbildung 2.15 die Nummer 1 tragen. Um den Hexapod vor Beschädigungen zu schützen, soll der Kontakt dabei jedoch nicht direkt zwischen Plattform und Umgebung auftreten. Dafür hat im Rahmen dieser Arbeit die Konstruktion eines speziell an die Kraftregelung angepassten Endeffektors stattgefunden, der von unten an die Kraftmessdose geschraubt werden kann. Er besteht aus zwei Blechen, in denen eine zur x-z-Ebene parallele Rolle gelagert ist. Die Dimensionierung der Bleche stellt für alle folgenden Experimente sicher, dass der Kontakt immer über die Rolle eingeleitet wird und es nie zum Kontakt der Plattform mit der Umgebung kommt. Die Rolle soll Reibungseffekte minimieren. Der Hexapod soll während der Inbetriebnahme einen Kontakt auf den Tisch aufprägen, wie in Abbildung 6.5 dargestellt.

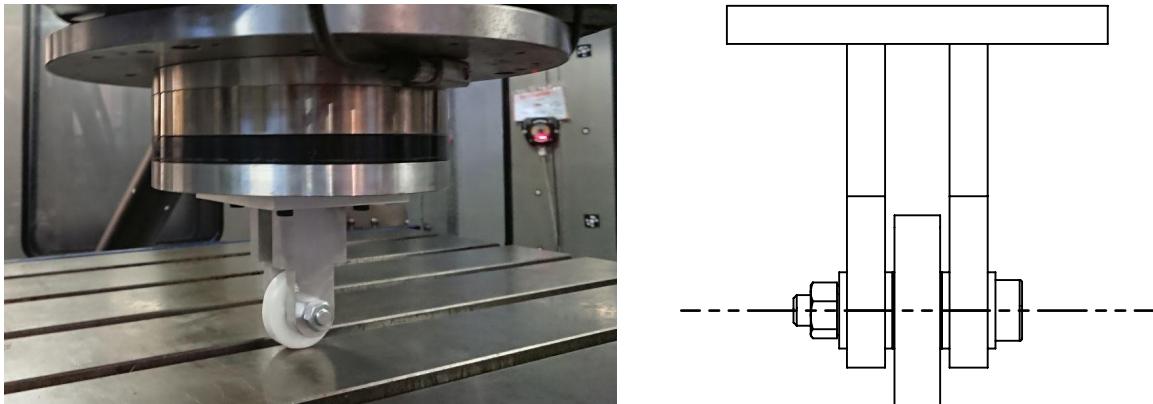


Abbildung 6.5.: Links: Versuchsaufbau der realen Inbetriebnahme, rechts: Endeffektor

Für den Versuch wird der Endeffektor wenige Zentimeter über dem Tisch positioniert. Dann beginnt die Kontaktsuche. Sobald ein stabiler Kontakt gefunden und die Aktivierung der Kraftregelung quittiert ist, beginnt die CNC damit, Kraftsollwerte in z-Richtung zu liefern. Zunächst fährt sie den gewünschten Kraftbereich an. Danach weist sie die Kraftregelung an, die Kraft zyklisch zu verringern und erhöhen. Dazu verwendet das Versuchsprogramm, das in Quelltext 6.5 aufgeführt ist, die Linearinterpolation der CNC.

Quelltext 6.5: Versuchsanweisung über G-Code: Variierende Kraftvorgabe in z-Richtung

```
#TRAFO ON
#VAR
V.P.x0 = -230. ; x-position in [mm]
V.P.y0 = 120. ; y-position in [mm]
V.P.z0 = -250. ; z-position in [mm]
V.P.fz0 = 225. ; lower force in [N]
V.P.fz1 = 370. ; upper force in [N]
#ENDVAR
G90 ; absolute coordinates
G93 ; F -> s
G1 F2 X=V.P.x0 Y=V.P.y0 Z=V.P.z0 ; Home position
G4 X1 ; wait to avoid measurement artifacts
M72 ; tare forces (platform)
G4 X1 ; wait to avoid measurement artifacts
M51 ; find contact
$FOR P2=0, 35, 1
G1 F5 X=V.P.x0 ZF=V.P.fz0 ; Force control goal F0
G1 F5 X=V.P.x1 ZF=V.P.fz1 ; Force control goal F1
$ENDFOR
M52 ; ramp down contact search
G90 ; absolute coordinates
G1 F2 Z=V.P.z0 ; Home position
M30 ; end
```

Das in Quelltext 6.5 beschriebene Experiment ist so konzipiert, dass die Störgrößen möglichst gering bleiben. Das wird dadurch erreicht, dass die Position in x- und y-Richtung nicht variiert.

Die Reglereinstellung geschieht deswegen in der Annahme des reinen Führungsverhaltens. In Abbildung 6.7 sind die Kraftverläufe während des Versuchs abgebildet. An den Umkehrpunkten ist eine Schwingungsanregung zu sehen. Diese tritt nur im realen Versuch auf, die virtuelle Entsprechung in Abbildung 6.4 weist dieses Verhalten nicht auf. Je härter die Reglerparameter eingestellt sind, desto stärker fängt der Hexapod zu schwingen an. Durch Filterung der Kraftwerte mit einem PT1-Filter mit der Zeitkonstante 100 ms lässt sich die Schwingung erheblich reduzieren. Im ursprünglichen Versuchsaufbau (Abbildung 6.5) treten diese Schwingungen trotz der Filterung so schnell auf, dass eine praxistaugliche Reglereinstellung nicht möglich ist.  $T_n$  beträgt 10 s und  $K_p$  beträgt 0,017 mm/N. Weil sowohl Umgebung, als auch Hexapod mit einer Gesamtsteifigkeit von 6 335 N sehr steif sind, entstehen bereits bei kleinen Positionsoffsets große Kräfte. Die Berechnung der Gesamtsteifigkeit erfolgt aus den Daten des Führungsverhaltensexperiments über den anstieg der Istkraft über den geregelten Weg (siehe Abbildung 6.9). Ein für diese Steifigkeit eingestellter Regler kann bei der vorliegenden Taktrate von 2 ms nicht schnell genug folgen. Abhilfe schafft ein Federpaket (siehe Abbildung 6.6), das zwischen Kraftmessdose und Rolle geschraubt wird. Die untergelegten Kautschukscheiben wirken schwingungsdämpfend. Nach dem Umbau weist das Gesamtsystem eine Steifigkeit von 119 N/mm auf.

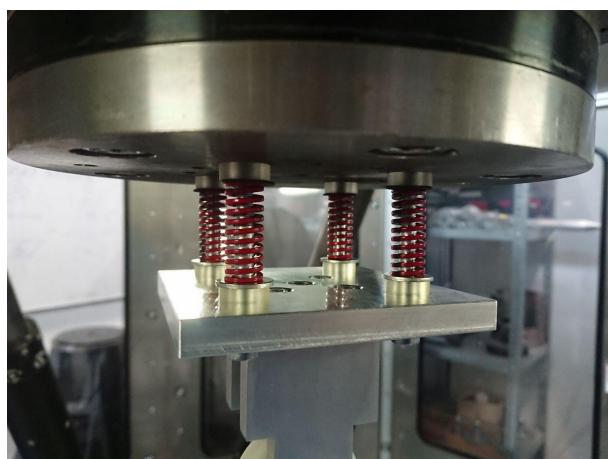


Abbildung 6.6.: Umbau zur Erhöhung der Nachgiebigkeit

Nach den Umbauten erfolgt die Durchführung für die verringerte Gesamtsteifigkeit des Systems. Die Ergebnisse sind in Abbildung 6.7 zu sehen.

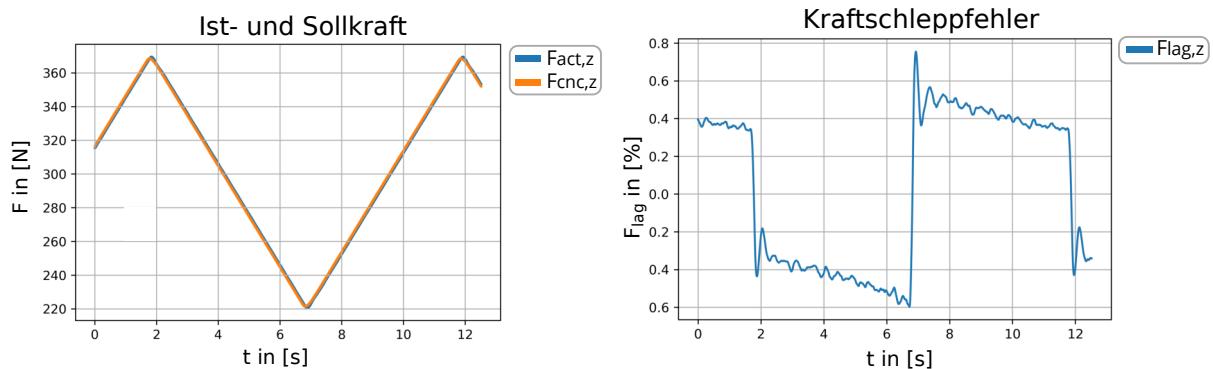


Abbildung 6.7.: Eine Periode der Sollkraftänderung für die Reglereinstellung, Links: Kraftvorgabe und -istwert, Rechts: relative Regelabweichung

Der Versuch zeigt, dass die Kraftregelung funktioniert. Nach Einbau der Federn und gleichzeitiger Verwendung des Filters lässt sich die geforderte Kraft präzise Einstellen. Das Führungsverhalten ist gut geeignet, um grob einzuschätzen, in welcher Größenordnung sich stabile Reglerparameter befinden. Wesentlich für die Reglergüte ist jedoch das Störverhalten, da der Regler hierbei stärker gefordert wird und die Störeinflüsse das System außerdem zu Schwingungen anregen können. Letzteres führt dazu, dass die Reglerparameter mitunter weicher ausfallen können, als beim Führungsverhalten. Deswegen schließt sich der Analyse des Führungsverhaltens eine Reglereinstellung im Störbetrieb an. Dafür wird das Programm 6.5 leicht angepasst. Statt der Kraftvorgabe ist nun die Wegvorgabe variabel. Die Änderungen sind in Quelltext 6.6 dargestellt. Der Endeffektor soll in z-Richtung kraftgeregelt, in x-Richtung jedoch weggregelt verfahren. Durch Oberflächenrauhigkeiten, Unebenheiten im Tisch und unrunde Stellen der Rolle entstehen Störeinflüsse, die sich in der Regelung niederschlagen. Die hierbei ermittelten Reglerparameter kommen für den weiteren Verlauf dieser Arbeit zum Einsatz.

Quelltext 6.6: Änderungen gegenüber Quelltext 6.5 für den Versuch zum Störverhalten

```
#TRAFO ON
#VAR
; x-yosition in [mm]
V.P.x0 = -230.          ; x start position in [mm]
V.P.x1 = -200.           ; x end position in [mm]
V.P.fz0 = 370.            ; lower force in [N]
V.P.fz1 = 370.            ; upper force in [N]
#ENDVAR
...
```

Nach den Umbauten findet die Reglereinstellung durch das Verfahren nach Ziegler/Nichols[65] im Störverhalten statt. Abbildung 6.8 zeigt die Kraftverläufe des Störverhaltens. Im Vergleich zum Führungsverhalten sind hierbei Schwingungen und Störgrößeneinfluss deutlich erkennbar. K<sub>p</sub> ergibt sich zu 0,25 mm/N und T<sub>n</sub> zu 0,5 s.

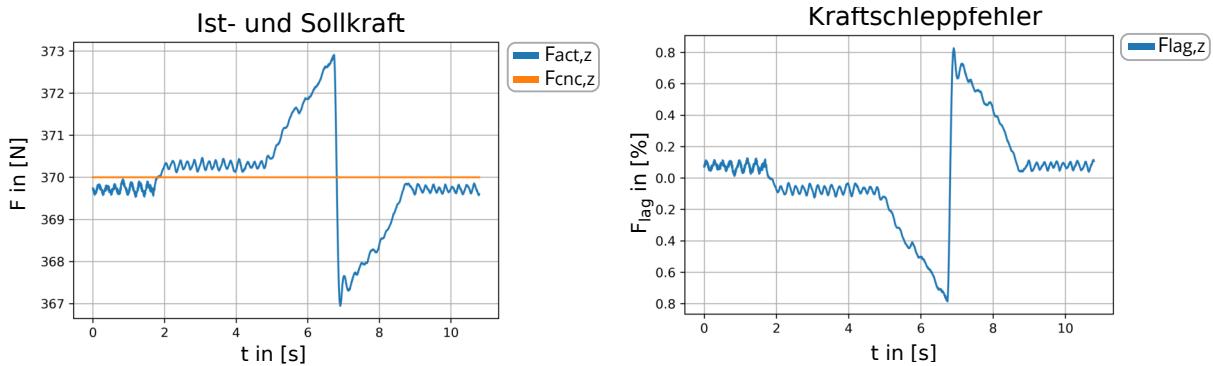


Abbildung 6.8.: Kraftverläufe im Störbetrieb, Links: Kraftvorgabe und -istwert, Rechts: relative Regelabweichung

Beim Durchfahren des unteren Kraftbereichs hat sich gezeigt, dass das System bei der Regelung niedriger Kräfte zu schwingen beginnt (siehe dazu Abbildung 6.9). Das könnte mit dem in Abbildung 6.9 dargestellten Hystereseverhalten des Systems zusammenhängen. Infolgedessen werden im Folgenden in z-Richtung nur Sollkräfte oberhalb von 100 N vorgegeben.

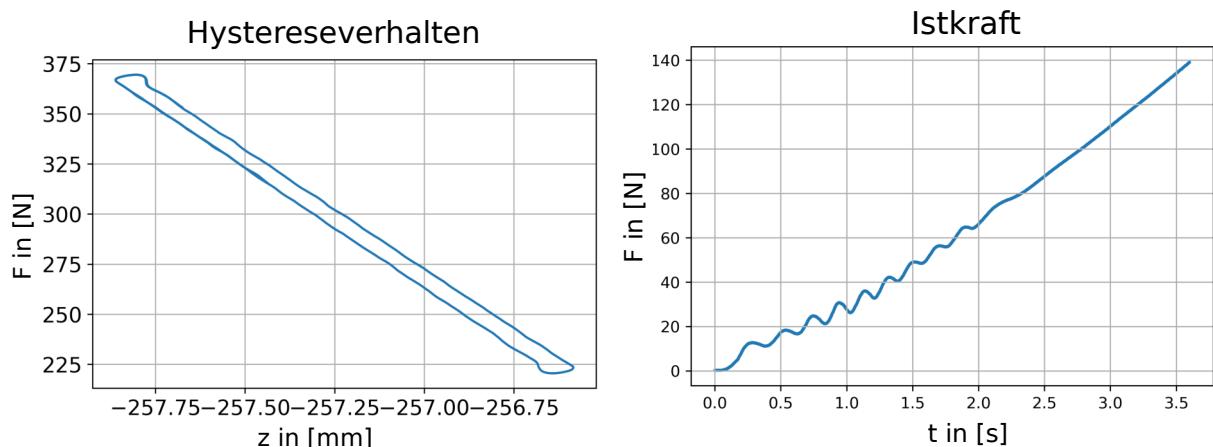


Abbildung 6.9.: Links: Hystereseffekt in der Istkraft des Versuchs zum Führungsverhalten, rechts: Schwingungen bei Kräften unterhalb von 100 N

Weil der untere Bereich bei jeder Kontaktsuche zwingend durchfahren wird, muss die Schwingung reduziert werden. Der einfachste Weg den schwingungsanfälligen Bereich zu durchqueren liegt darin, zeitweilig weiche Parameter am Kraftregler einzustellen. Sobald der kritische Bereich verlassen ist, können harte Reglerparameter übernommen werden. Dafür besteht die Möglichkeit, im G-Code zwei Kraftreglerparametersets anzugeben, zwischen denen mit M52 und M53 hin- und hergeschalten werden kann. Quelltext 6.7 erweitert dafür die Versuchsanweisungen aus Quelltext 6.5.

Quelltext 6.7: Vorgabe weicher und harter Reglerparameter

```

...
M51 ; find contact
M53 ; NEW: switch on soft params
G1 F5 ZF=V.P.fz0
M52 ; NEW: switch on hard params
...
G1 F5 X=V.P.x1 ZF=V.P.fz1 ; F0
...

```

Die reale Inbetriebnahme hat die Funktionalität der Kraftregelung am Hexapod validiert. Die Umbauten erzielen eine Verbesserung der Leistungsfähigkeit, weil sie das Einstellen härterer Regelparameter ermöglichen.

Die im Rahmen der Inbetriebnahme durchgeführten Experimente zeigen außerdem, dass das System in Kraftregelung zu Schwingungen neigt. Um die Ursache dessen näher zu erörtern, findet ein weiteres Experiment statt, in dem Schwingungen gezielt angeregt werden. Dafür wird eine sinusförmige Sollkraft mit über die Zeit linear steigender Frequenz im Führungsbetrieb vorgegeben. Abbildung 6.10 zeigt die Analyse im Frequenzbereich. Dafür werden Ein- (Kraftsollwert, mit PT1 gefiltert) und Ausgangssignal (Kraftistwert, mit PT1 gefiltert) separat mit der Fouriertransformation in den Bildbereich transformiert. Anschließend wird der transformierte Ausgang durch den transformierten Eingang geteilt und logarithmisch aufgetragen.

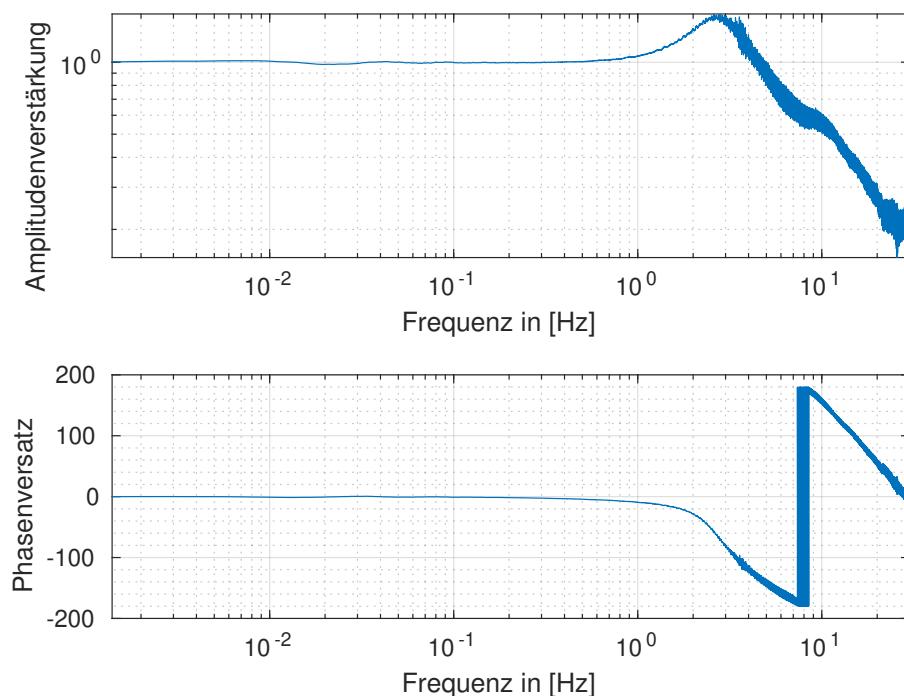


Abbildung 6.10.: Bodediagramm des Schwingungsversuchs

Deutlich erkennbar ist eine Amplitudenüberhöhung bei ca. 3 Hz. Die Einstellung der Filterzeit beeinflusst Größe und Frequenz der Maximums der Amplitudenüberhöhung. Experimentell hat sich ergeben, dass sich mit einer Filterzeit von 100 ms die Schwingungen am stärksten reduzieren lassen. Dabei ist unklar, ob die Anregung aus der Mechanik der Maschine oder dem Regelkreis kommt. Um den Ursprung der Schwingung zu finden und zu beheben ist es notwendig, das System eingehend zu analysieren. Das würde den Rahmen dieser Arbeit sprengen. Der Schwingungsversuch bietet nachfolgenden Untersuchungen einen Ansatzpunkt. Im Moment kann nur vermutet werden, dass die mechanische Trägheit, Spiel in den Antrieben, Totzeiten im Regelkreis, die Art und Einstellung der Antriebsregler sowie mechanische Eigenfrequenzen für die Schwingungen und damit die Begrenzung der Reglerparameter für die schwingungsanfälligkeit des Systems verantwortlich sind.

# 7. Experimentelle Validierung

## Überblick

Der vorangegangene Abschnitt beschreibt die Inbetriebnahme der Kraftregelung am trivialen Fall. Durch die kleinen Stellwege und die geringen Störeinflüsse sind die im Rahmen der Inbetriebnahme durchgeföhrten Experimente für reale Prozesse jedoch nicht repräsentativ. Um evaluieren zu können, ob die Kraftregelung am Hexapod mit der gewählten Regelstrategie zur Anwendung in praxisrelevanten Prozessen taugt, sind weitere Versuche notwendig. In diesem Abschnitt sind Entwurf, Durchführung und Analyse mehrerer Experimente beschrieben. Zunächst erfolgt die Kraftregelung im globalen kartesischen Raum (Abschnitt 7.1). Nach der Validierung folgt in Abschnitt 7.2 ein Vergleich der Eignung für die Regelung zwischen strukturiertegrieter und konventioneller Kraftmessung. Anschließend setzt sich Abschnitt 7.3 mit der Kraftregelung in variablen Koordinatensystemen auseinander.

## 7.1. Kraftregelung im kartesischen Raum

Zu Beginn der Validierung müssen geeignete Versuche konzipiert werden. Die Experimente sollen verdeutlichen, dass

- die Kraftregelung die Interaktion mit einer teilweise unbekannten Umwelt ermöglicht,
- die gleichzeitige Interpolation von Weg und Kraft die Regelgüte verbessert,
- praxisrelevante kraftgeregelte Prozesse zweckmäßig per G-Code beschrieben werden können und
- der Kraftregler in der Lage ist, in angemessener Zeit Stellwege im Zentimeterbereich zu erbringen.

Charakteristisch für Fertigungsprozesse sind die hohen Störgrößen, die aus der Werkstückbearbeitung resultieren. Trotz möglichst hoher Prozessgeschwindigkeiten ist der Kraftschleppfehler  $F_{lag}$  gering zu halten, um Toleranzen zu erfüllen und einen stabilen Prozess zu gewährleisten. Der Testprozess soll all diese Herausforderungen abbilden, gleichzeitig aber möglichst

abstrakt sein, damit sich die Ergebnisse leicht auf eine breite Prozessvielfalt übertragen lassen. Um die Interpretation der Ergebnisse zu erleichtern und Versagensfälle eindeutig auf ihre Ursache zurückführen zu können, soll der Prozess so einfach gestaltet sein wie möglich. Um sich als Demonstrationsprozess zu eignen ist es wichtig, dass der Prozess sich schnell und unkompliziert auf- sowie abbauen lässt. Ebenso muss er deswegen gut kontrollierbar und sicher in der Durchführung sein.

Im Folgenden wird der Versuchsprozess erläutert, der diese Anforderungen erfüllt. Auf dem Tisch des Hexapods (siehe Abbildung 2.17) wird eine Geometrie mit stetiger Kontur befestigt, die in Abbildung 7.1 zu sehen ist und im Folgenden als *Testgeometrie 1* bezeichnet wird. Die Befestigung, erfolgt durch eine Stiftverbindung mit einer am Tisch angeschraubten Halterung wie in Abbildung 7.1 dargestellt. Mit der in Abschnitt 6.2 beschriebenen Rolle soll der Hexapod diese für die Steuerung unbekannte Kontur verfolgen. Dafür wird eine Kraft in z-Richtung vorgegeben, die mithilfe der Kraftregelung aufgebracht werden soll. Gleichzeitig findet eine Wegregelung in x-Richtung statt. Die Startposition wird per Hand eingestellt und liegt wenige Zentimeter über dem linken Auslauf der Testgeometrie 1.

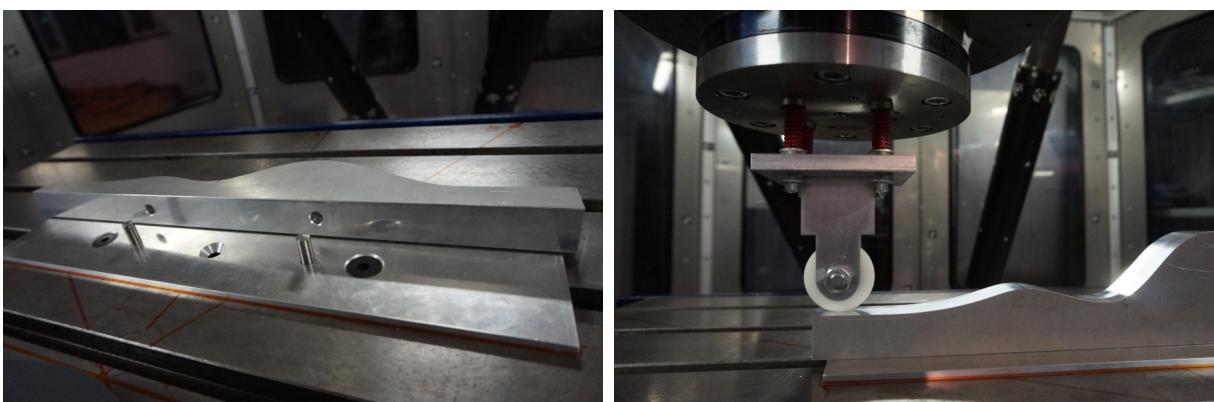


Abbildung 7.1.: Vertikale Testgeometrie, Links: Halterung, Rechts: Versuchsaufbau

Die im Verlauf der x-Richtung stark variable Höhe der Testgeometrie verursacht über die Trajektorie große Störeinflüsse in z-Richtung: Wenn die Kraftregelung direkt nach der Kontaktstufe ausgeschaltet wird, treten schon nach wenigen Zentimetern Kräfte über 600 N auf. Um den Störgrößeneinfluss zu quantifizieren, erfolgt die Versuchsdurchführung in zwei Varianten:

1. Validierung der parallel/hybriden Kraftregelung: Zunächst folgt der Hexapod der Kontur des Objekt ohne Vorsteuerung in z-Richtung. Das entspricht dem Verhalten des Systems während der Interaktion mit einem unbekannten Objekt. Die Ergebnisse sind in Abbildung 7.2 dargestellt.
2. Validierung der parallel/hybriden Kraftregelung mit zusätzlicher Vorsteuerung: Weil der neue Steuerungsansatz eine gleichzeitige Interpolation von Weg und Kraft erlaubt, ist es möglich, eine Vorsteuerung in z-Richtung umzusetzen. Dieses Verhalten lässt sich natürlich nur durch zumindest ungefähre Kenntnis der Geometrie erzielen. Allerdings hat das

Ausmessen der vorgegebenen Stützstellen nur wenige Minuten in Anspruch genommen. Es wäre auch denkbar die Kontur des Objektes einmal ohne Vorsteuerung aufzunehmen und dann für jede x-Position eine Vorsteuerung in z-Richtung zu generieren. Abbildung 7.2 illustriert die Ergebnisse der einfachen Vorsteuerung mit vier Stützpunkten. Der dafür notwendige G-Code ist in Quelltext 7.1 aufgeführt.

Quelltext 7.1: Für die Konturverfolgung genutzte Programmanweisungen über G-Code: Vorsteuerung in z-Richtung auf dem Hinweg, Rückweg ohne Vorsteuerung

```
#TRAFO ON
#VAR
;(start position)
V.P.x0 = -240. ; x-yosition in [mm]
V.P.y0 = 120. ; y-yosition in [mm]
V.P.z0 = -250. ; x-yosition in [mm]
V.P.fz0 = 250. ; goal force [N]
;(feed forward position interpolation grid points)
V.P.x1 = -100. ; x-yosition of p1 in [mm]
V.P.x2 = -35. ; x-yosition of p2 in [mm]
V.P.x3 = 55. ; x-yosition of p3 in [mm]
V.P.x4 = 175. ; x-yosition of p4 in [mm]
V.P.z1 = -235. ; y-yosition of p1 in [mm]
V.P.z2 = -250. ; y-yosition of p2 in [mm]
V.P.z3 = -195. ; y-yosition of p3 in [mm]
V.P.z4 = -250. ; y-yosition of p4 in [mm]
;(goal position)
V.P.x5 = 225. ; x-yosition of p in [mm]
V.P.z5 = -250. ; x-yosition of p in [mm]
#ENDVAR
G90 ; absolute coordinates
G93 ; F -> s
G1 F2 X=V.P.x0 Y=V.P.y0 Z=V.P.z0 ; Home position
G4 X1 ; wait to avoid measurement artifacts
M72 ; tare forces (platform)
G4 X1 ; wait to avoid measurement artifacts
M51 ; find contact
$FOR P2=0, 1, 1
    G1 F50 X=V.P.x1 Z=V.P.z1 ZF=V.P.fz0 ; p1
    G1 F22 X=V.P.x2 Z=V.P.z2 ZF=V.P.fz0 ; p2
    G1 F35 X=V.P.x3 Z=V.P.z3 ZF=V.P.fz0 ; p3
    G1 F44 X=V.P.x4 Z=V.P.z4 ZF=V.P.fz0 ; p4
    G1 F18 X=V.P.x5 Z=V.P.z5 ZF=V.P.fz0 ; p5
$ENDFOR
M52 ; ramp down contact search
G90 ; absolute coordinates
G1 F2 Z=V.P.z0 ; Home position
M30 ; end
```

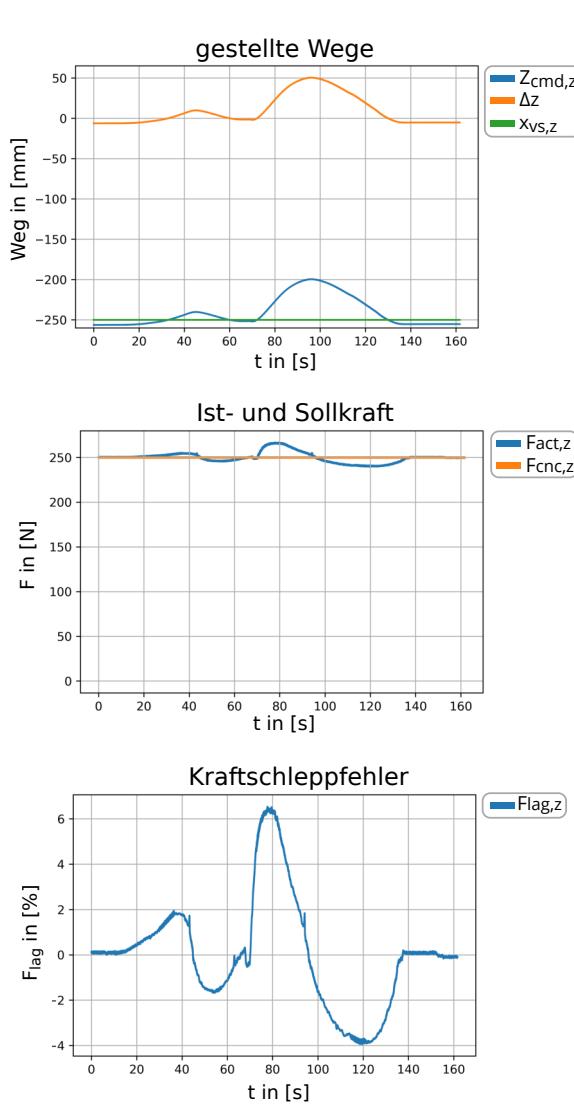


Abbildung 7.2.: Konturverfolgung der Testgeometrie 1 ohne Vorsteuerung

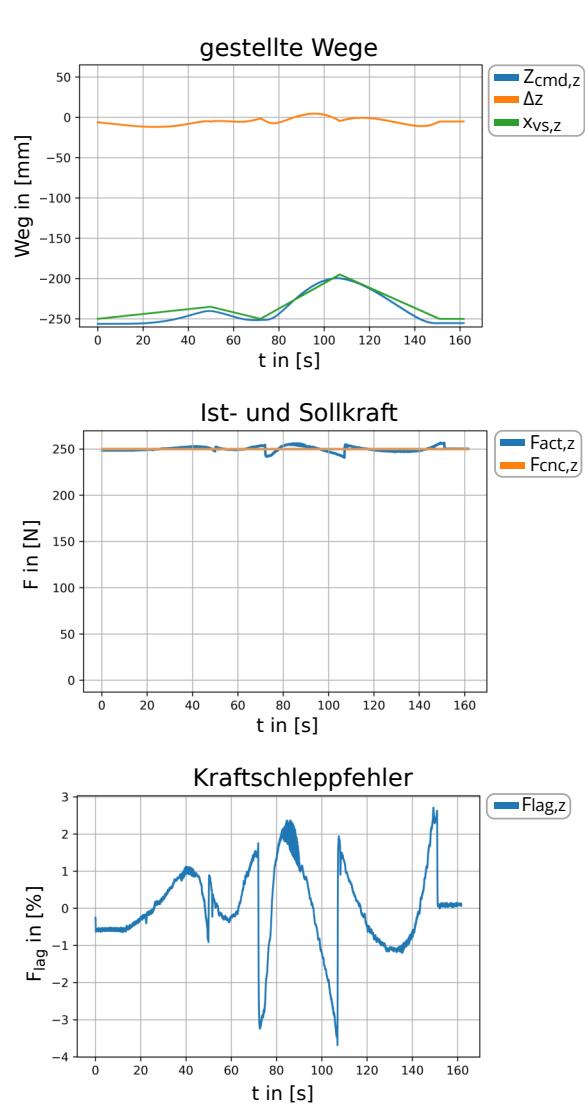


Abbildung 7.3.: Konturverfolgung der Testgeometrie 1 mit Vorsteuerung

Für die Vorsteuerung wird die ungefähre Form des Objekts im G-Code vorgegeben und wegeregelt angefahren. Der zeitliche Verlauf dieser Vorgabe ist als  $x_{vs,z}$  dargestellt. Der den Wegregler dominierende Kraftregler muss nur noch die Differenz zwischen realem Objekt und Vorsteuerung ausgleichen. In den Wegdiagrammen wird deutlich, dass der Regler deswegen wesentlich weniger Weg ( $\Delta z$ ) stellen muss als ohne Vorsteuerung. Gut zu erkennen ist im Kraftverlauf, dass die größte Abweichung ohne Vorsteuerung am Punkt des steilsten Anstiegs der Geometrie liegt, während die Abweichung mit Vorsteuerung in den Linearinterpolationsstützstellen der Vorsteuerung am größten ist. Durch die Vorgabe einer in der ersten Ableitung stetigen Vorsteuerung ließen sich die Unstetigkeiten der Istkraft und damit auch die Regeldifferenz wesentlich verkleinern. In beiden Varianten kann eine Regeldifferenz deutlich unter 10 % erzielt werden - allerdings nur bei der relativ niedrigen Geschwindigkeit von 1,61 mm/s in x-Richtung. Das ist bei der geringen Regelbandbreite (siehe Abbildung 6.10) jedoch nicht verwunderlich.

Tabelle 7.1 führt einige Kennzahlen der beiden Varianten auf.

Tabelle 7.1.: Vergleich der zwei Varianten

Vorsteuerung	Max. Zustellgeschwindigkeit ( $\frac{\Delta z}{t}$ ) <sub>max</sub>	Max. Regelabweichung $F_{lag,max}$
Nein	3,12 mm/s	16,43 N $\hat{=}$ 6,6 %
Ja	1,56 mm/s	-8,83 N $\hat{=}$ 3,5 %

Die Kennzahlen aus Tabelle 7.1 verdeutlichen, dass die Regelung im Vergleich zur Vorgängerarbeit einen erheblichen Dynamikzugewinn erzielt. Vogt [6] schreibt, dass die Maximalabweichung in seinem Beispielprozess 7,3 % beträgt, wobei jedoch nur eine Zustellgeschwindigkeit von 0,1 mm/s erforderlich ist. Bei vergleichbarer Abweichung kann hier also mit 3,12 mm eine fast 30-fache Dynamik verfahren werden. Zu vermuten ist, dass das aus der verringerten Totzeit durch die vermiedene zusätzliche Bahnplanung resultiert.

## 7.2. Vergleich verschiedener Sensoren

Um die Eignung der verschiedenen Kraftmesseinrichtungen des Hexapods zur Kraftregelung miteinander vergleichen zu können, soll der Prozess des vorherigen Abschnitts mit der kommerziellen 6D-Kraftmessplattform durchgeführt werden. Das System ist hierbei etwas weniger schwingungsanfällig. Dadurch ist es möglich, härtere Reglerparameter einzustellen. Kp ergibt sich zu 0,31 mm/N und Tn zu 0,2 s. Der Verlauf der Kräfte ist im Folgenden dargestellt. Dabei ist die Vorsteuerung des letzten Abschnitts deaktiviert.

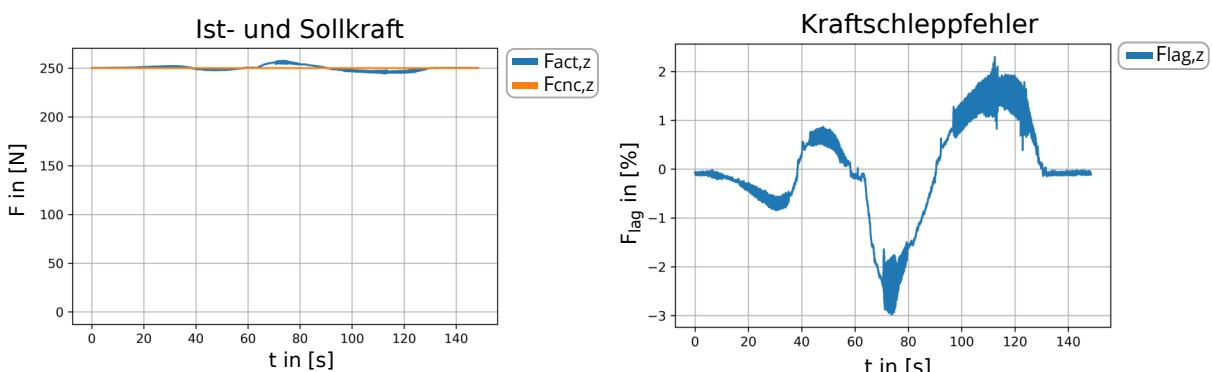


Abbildung 7.4.: Kraftverläufe der Konturverfolgung in z-Richtung mit konventionellem Kraftsensor ohne Vorsteuerung, Links: Kraftvorgabe und -istwert, Rechts: relative Regelabweichung

Durch die Verwendung des konventionellen Kraftmesssensors reduziert sich die maximale Regelabweichung gegenüber der strukturintegrierten Messung ohne Vorsteuerung um das dreifache, weil das System weniger schwingungsanfällig ist und dadurch die Verwendung härtere Reglerparameter zulässt. Das hängt vermutlich damit zusammen, dass die konventionelle Kraftmesseinrichtung näher am Prozess befestigt ist. Dadurch liegt weniger schwingungsfähig-

ge Struktur zwischen Kontakt und Messstelle, die zu einem Phasenversatz des gemessenen Signals zum wahren Signal führt und damit die Regelbandbreite verringert. Dennoch ist das Ergebnis der strukturintegrierten Messung akzeptabel, sodass letztere für alle weiteren Versuche genutzt wird.

### 7.3. Kraftregelung in variablen Koordinatensystemen

Die vorangegangenen Abschnitte beschreiben die Implementierung und Erprobung einer Kraftregelung entlang der globalen kartesischen Achsen. Obwohl der Beispielprozess in Abschnitt 7.1 die Leistungsfähigkeit der Kraftregelung bestätigt, ist sie für alle Prozesse ungeeignet, in denen sich die Richtung der Prozesskraft im Lauf des Prozesses ändert. Besonders in der Fertigungstechnik ist jedoch genau dieses Szenario der Fall, weil die Bearbeitung oftmals senkrecht zur Werkstückoberfläche geschieht. Im Beispielprozess aus dem vorangegangenen Abschnitt treten genau diese Kräfte auf, wie in Abbildung 7.5 zu sehen ist. Die Querkräfte rufen eine Verkipfung des Endeffektors hervor, die durch Kraftregelung in Oberflächennormalrichtung verhindert werden kann. Um die Kraftregelung dazu zu befähigen, soll in diesem Abschnitt die Möglichkeit geschaffen werden, den *Task Space* aufbauend auf dem *User Space* stetig an den Prozess anzupassen. Die beiden Begriffe sind in Abschnitt 4 erläutert. Ziel ist eine Kraftregelung in Oberflächennormalenrichtung. Die Evaluierung soll analog zu den bisherigen Versuchen als Konturverfolgung umgesetzt werden.

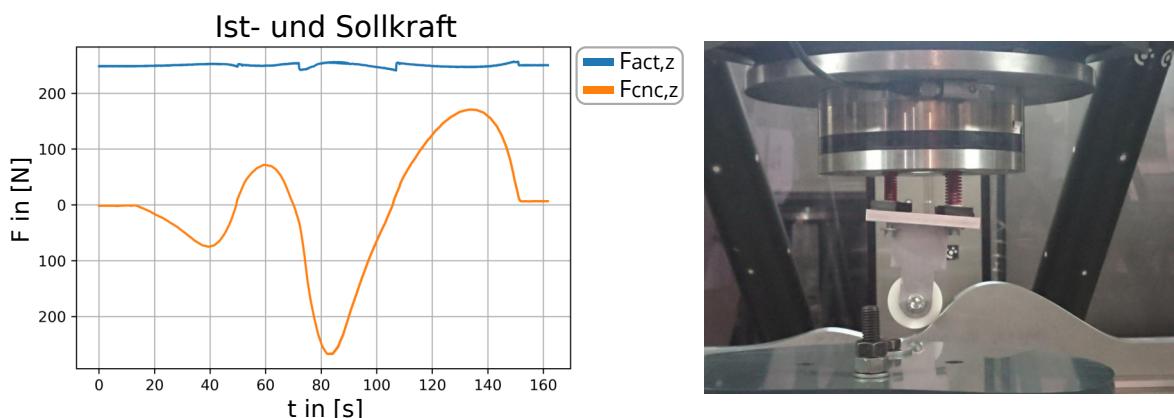


Abbildung 7.5.: Links: Querkraft während der Konturverfolgung durch Kraftregelung in z-Richtung aus Abschnitt 7.1, Rechts: Aus der Querkraft resultierende Verkipfung ohne Führung

Die Umsetzung erfolgt in zwei Etappen. Abschnitt 7.3.1 validiert die Regelung im variablen *Task Space* anhand einer vereinfachten posebasierten Transformationsvorschrift. Aufbauend auf den dabei erlangten Ergebnissen findet die Transformation in Abschnitt 7.3.2 auf Grundlage der gemessenen Kräfte statt.

### 7.3.1. Kraftregelung in Radialrichtung

Um die Kraftregelung in variablen Koordinaten zu betreiben, muss die Steuerung das Koordinatensystem des *Task Spaces* in jedem Takt anpassen. Der Ursprung des *Task Spaces* fällt weiterhin mit dem *TCP* zusammen. Dessen Orientierung soll jedoch ab jetzt von der Kraftreglungsaufgabe abhängen. In dieser Arbeit soll die x-Achse in Richtung der Oberflächennormale gedreht werden. Für die Konturverfolgung wird die Sollkraft also als x-Kraft im G-Code programmiert. Später soll die Rotation auf beliebigen Berechnungsvorschriften erfolgen und damit der jeweiligen Kraftreglungsaufgabe gerecht werden. Abbildung 7.6 illustriert das gedrehte Koordinatensystem. Um die Regelung im variablen *Task Space* zu erproben, werden zunächst eine Reihe an Vereinfachungen getroffen.

1. Die abzutastende Oberfläche soll sich senkrecht zur x-y-Ebene des *Task Spaces* befinden. Das kann durch die Rotation des Nutzerkoordinatensystems *USR* auch für Aufgaben gewährleistet werden, in denen die Geometrie geneigt ist. Dadurch kann eine Kraftregelung in Normalenrichtung umgesetzt werden, ohne dass der Hexapod seine Orientierung während der Kraftregelung ändern muss. Um das zu erreichen, wird ein flacher Zylinder auf dem Tisch montiert, dessen Mantelflächen abzutasten sind. Eine Darstellung befindet sich in Abbildung 7.6.
2. Durch die Wahl eines Zylinders sind Oberflächennormalen- und Radialrichtung identisch.
3. Die Transformation vom *User Space* in den *Task Space* basiert auf einer Rotation um die z-Achse, welche den Richtungsvektor der x-Achse auf den Mittelpunkt des Zylindergrundkreises dreht. Die Wahl dieser positionsbasierten Transformationsberechnung verhindert eine mögliche durch Sensorrauschen verursachte Destabilisierung des Prozesses.

Der Prozess verlangt eine Anpassung des Endeffektors. Eine Schraube ersetzt die Bleche und lagert eine Rolle, die einen reibungslosen Kontakt entlang der Zylindermantelfläche zulässt. Abbildung 7.6 enthält eine Darstellung des neuen Endeffektors und illustriert die für die Regelung relevanten Koordinatensysteme. Die Berechnung des Winkels  $\alpha$ , der den *Task Space* rotiert, erfolgt nach folgender Vorschrift:

$$\alpha_{\text{rad}} = \text{atan2}(y_{\text{center}} - y_{\text{act}}, x_{\text{center}} - x_{\text{act}}) + \pi, \quad (7.1)$$

wobei  $P_{\text{center}}$  den Mittelpunkt des Grundkreises beschreibt. Dabei besteht das Problem, dass  $\mathbf{x}_{\text{act}}$  ausschließlich zu diesem Zweck gebraucht wird und nur über die rechenaufwändige Vorwärtstransformation beschafft werden kann. Die zusätzlichen Rechenkosten sind hier akzeptabel, müssen jedoch eventuell in Zukunft bedacht werden.

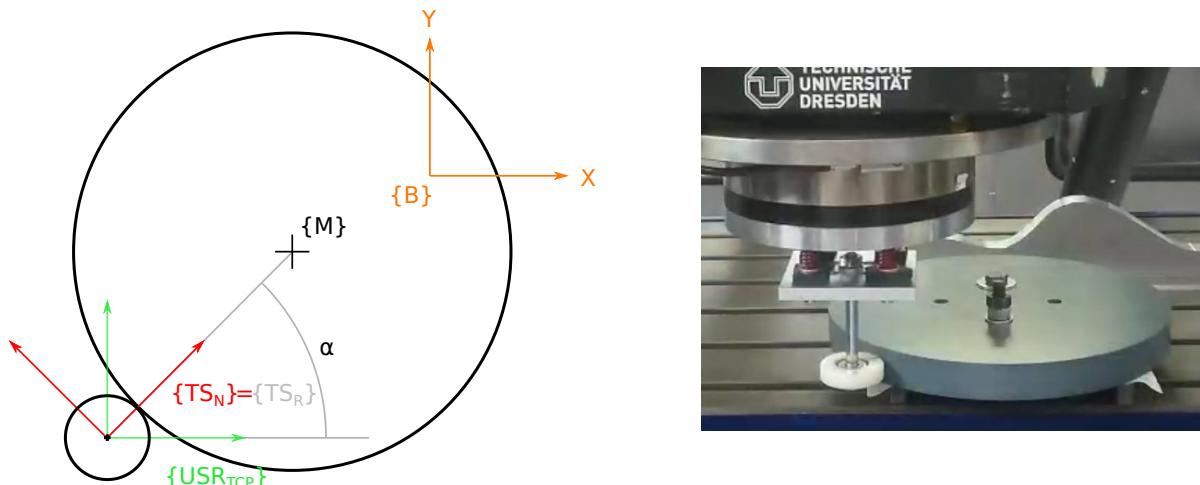


Abbildung 7.6.: Links: Koordinatensysteme der radialen Kraftregelung,  $TS_N$  steht für Task Space in Oberflächennormalenrichtung und  $TS_R$  in Radialrichtung, Rechts: eingespannter flacher Zylinder für das Experiment der Konturverfolgung

Vor Erprobung auf der Maschine findet ein virtueller Vorversuch mit dem Namen statt. Aus Platzgründen ist der dazugehörige Quelltext hier nicht aufgeführt, sondern auf der beigefügten DVD unter dem Namen *Test\_ForceManager.forceControlInTaskSpaceRadialDirection* einzusehen. Die Kontaktsuche geschieht in Radialrichtung. Nach gefundenem Kontakt fährt die Kraftregelung den Kraftwert von -50 N an. Das negative Vorzeichen resultiert aus der zur Kontaktfläche gedrehten Koordinatenachse des *Task Spaces*. Sobald der Kraftwert eingestellt ist, beginnt die CNC mit der Positions vorgabe einer Kreisbahn, die in diesem Prozess mit 240 mm einen größeren Durchmesser aufweist als der tatsächliche Zylinder, bei dem es 200 mm sind. Im Verlauf des Umlaufs wächst die Sollkraft per Linearinterpolation auf -75 N an, wie in Abbildung 7.8 farblich illustriert. Die Kraftregelung zieht den Endeffektor in Richtung Mittelpunkt und erhält somit den Kontakt. Dafür muss der Nutzer die Koordinaten des Grundkreismittelpunktes manuell übergeben. Das geschieht an der realen Maschine über den G-Code. Die entsprechenden Befehle sind in Tabelle 4.1 aufgeführt. Mithilfe dessen kann der Versuchsvorgang durch den G-Code beschrieben werden, wie in Quelltext 7.2 dargestellt.

Quelltext 7.2: Versuchsanweisung über G-Code: Variierende Kraftvorgabe in z-Richtung

```
#VAR
V.P.x0 = -50.
V.P.y0 = -137.5
V.P.z0 = -180.
V.P.zOperate = -270.
V.P.xOperate = V.P.x0 - 115
V.P.yOperate = V.P.y0 - 115
#ENDVAR
...
V.G.KIN_STEP[0].ID[65].PARAM[43] = V.P.x0
; center point x
V.G.KIN_STEP[0].ID[65].PARAM[44] = V.P.y0
; center point y
#TRAFO ON
```

```

G93 ; F in [s]
;### start on center of zylinder ###;
G1 F1 Z=V.P.z0 ; avoid crashes z
G1 F1 X=V.P.x0 Y=V.P.y0 ; Home position x y
G1 F1 X=V.P.xOperate Y=V.P.yOperate ; work starting position x y
G1 F1 Z=V.P.zOperate ; work starting position z
G4 X0.5 ; avoid decelleration artifacts
;### start of force control ###;
M70 ; tare forces (transducer)
M72 ; tare forces (platform)
M53 ; switch on TaskSpace
M51 ; contact search
G4 X0.01 ; avoid tare artifacts in c.-search
G1 F5 XF=-50
G161 ; absolute coordinates for circle
G03 X=V.P.xOperate I=V.P.x0 J=V.P.y0 XF=-75 F90
; circle with forceControl
G4 X1
M52 ; ramp down
M54 ; switch off TaskSpace
M30 ; stop

```

Bei der Durchführung resultiert der Mittelpunkt aus einer Schätzung. Dadurch sind realer Grundkreis und Wegvorgabe nicht konzentrisch. Das schlägt sich im Stellweg des Kraftreglers nieder, wie in Abbildung 7.8 und 7.12 zu erkennen ist.

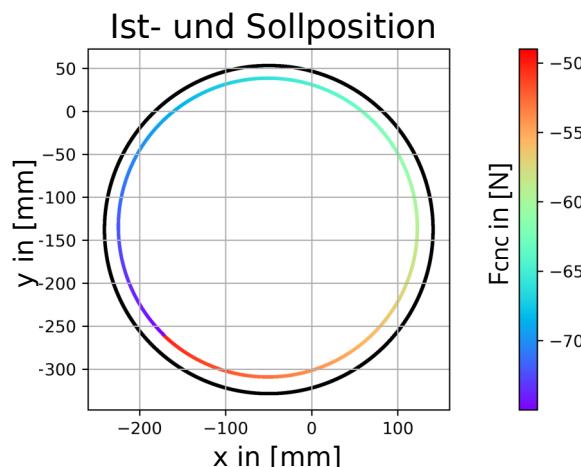


Abbildung 7.7.: Schwarz: Sollposition, Bunt: Istposition mit Kraftvorgabe an jedem Punkt

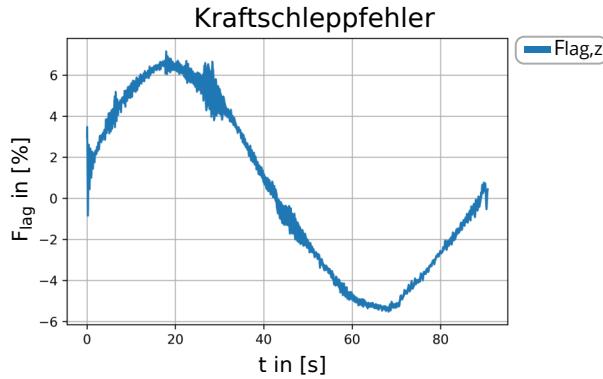


Abbildung 7.8.: Infolge der Exzentrizität schwankende Kraftschleppfehler

Mit etwa 6 % ist die Maximalabweichung so niedrig wie im vertikalen Versuch aus Abschnitt 7.1, obwohl die Anpresskraft viel niedriger ist. Außerdem bleiben die niederfrequenten Schwingungen aus (im Vergleich dazu Abbildung 6.9). Daraus folgt, dass sich das System in den verschiedenen kartesischen Richtungen sehr in seinen Reaktionen und seiner Schwingungsanfälligkeit unterscheidet. Das erlaubt es, in der x-y-Ebene härtere Reglerparameter einzustellen und ergo Prozesse mit deutlich höherer Geschwindigkeit durchzuführen. Um das Verhalten des Systems bei höheren Geschwindigkeiten zu untersuchen, wird der Versuch mehrfach wiederholt, wobei die Prozesszeit der Kreisbahn immer weiter reduziert wird. Abbildung 7.9 stellt die Schleppfehler dar.

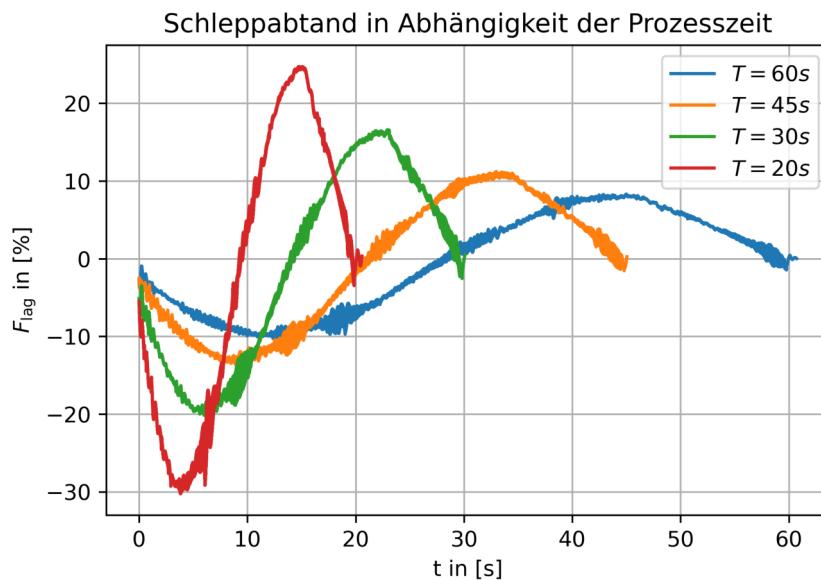


Abbildung 7.9.: Relative Kraftschleppfehler bei varierter Prozesszeit

Dabei ist zu erkennen, dass sich das Schwingungsverhalten nicht signifikant ändert, die Amplituden wachsen lediglich linear mit den Kraftschleppfehlern. Diese verhalten sich umgekehrt proportional zur Prozessgeschwindigkeit. Dieser Versuch stellt die Funktionsfähigkeit der Kraftregelung für gedrehte *Task Spaces* unter Beweis.

### 7.3.2. Kraftregelung in Normalenrichtung

Abschnitt 7.3.1 beschreibt die Kraftregelung in variabler Prozessrichtung und die dafür notwendige Rotation des *Task Spaces*. Die Prozessrichtung ist dort auf Basis der Position definiert. Dadurch kann ein stabiler Prozess gewährleistet werden. Die radiale Variante der Transformation ist jedoch nicht optimal, weil sie bei steigender Exzentrizität der Wegvorgabe zunehmend ungenau wird. Ohne Kenntnis des Objektes kann diese Art der Regelung gar nicht genutzt werden, weil kein Mittelpunkt zur Verfügung steht.

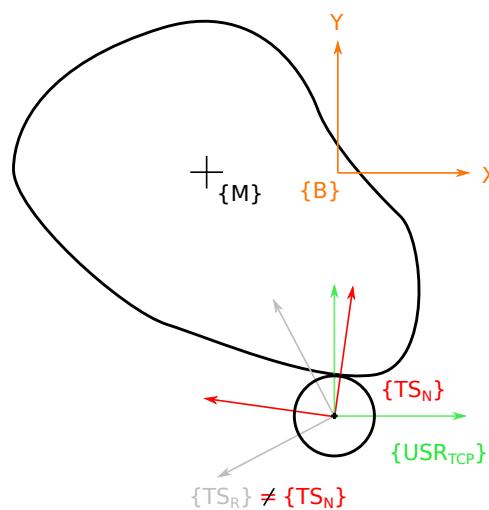


Abbildung 7.10.: Koordinatensysteme der Kraftregelung in Oberflächennormalenrichtung,  $TS_N$  steht für Task Space in Normalenrichtung und  $TS_R$  in Radialrichtung

Noch gravierender ist, dass die radiale Kraftregelung nur für radialsymmetrische Oberflächen präzise funktioniert. Das betrifft nicht nur die aktiv geregelte Kraft, sondern auch die Erzeugung von Querkräften. Um diesen Einfluss zu verdeutlichen, wird der Versuch aus Abschnitt 7.3.1 wiederholt, nur dass statt des Zylinders eine unregelmäßige Geometrie im Arbeitsraum befestigt ist.

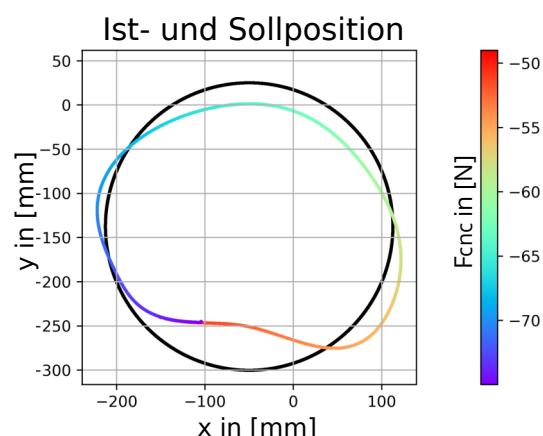
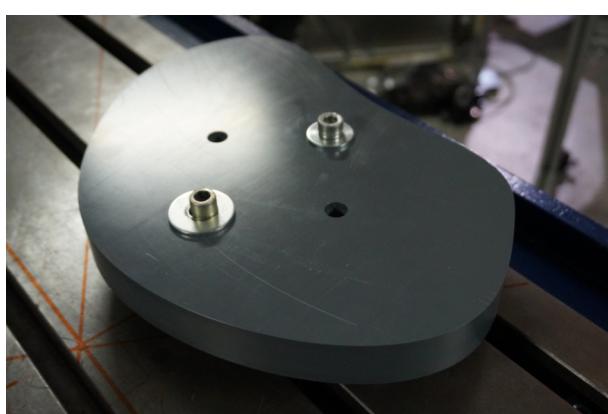


Abbildung 7.11.: Testgeometrie 2, Links: Photo, Rechts: Repräsentation aus Konturverfolgung

Die neue Form ist in Abbildung 7.11 dargestellt und wird im Folgenden *Testgeometrie 2* genannt. Die Differenz zwischen Wegvorgabe und wirklicher Geometrie ist signifikant größer als für den Zylinder. Abbildung 7.12 vergleicht die auftretenden Kräfte zwischen Testgeometrie 2 und Zylinder. Weil der Kraftregler für die Testgeometrie 2 signifikant mehr Weg zustellen muss, weist der Prozess einen höheren Schleppfehler auf. Das ist kein Versagen der radialen Regelstrategie. Problematisch sind jedoch die großen Querkräfte. Während sie für den Zylinder unter 10 % bleiben, liegen sie bei der Testgeometrie 2 fast in derselben Größenordnung wie die aktiv geregelte Kraft. Das Ziel dieses Abschnittes ist es, diese Querkräfte zu minimieren und die Regelung für eine breitere Vielzahl an Geometrien zu ertüchtigen. Dafür soll die Rotation des *Task Spaces* nicht auf Basis der Position, sondern auf Basis der Kontaktkräfte erfolgen. Abbildung 7.10 illustriert die Unterschiede der Task-Space-Berechnung im Vergleich zur radialen Regelung.

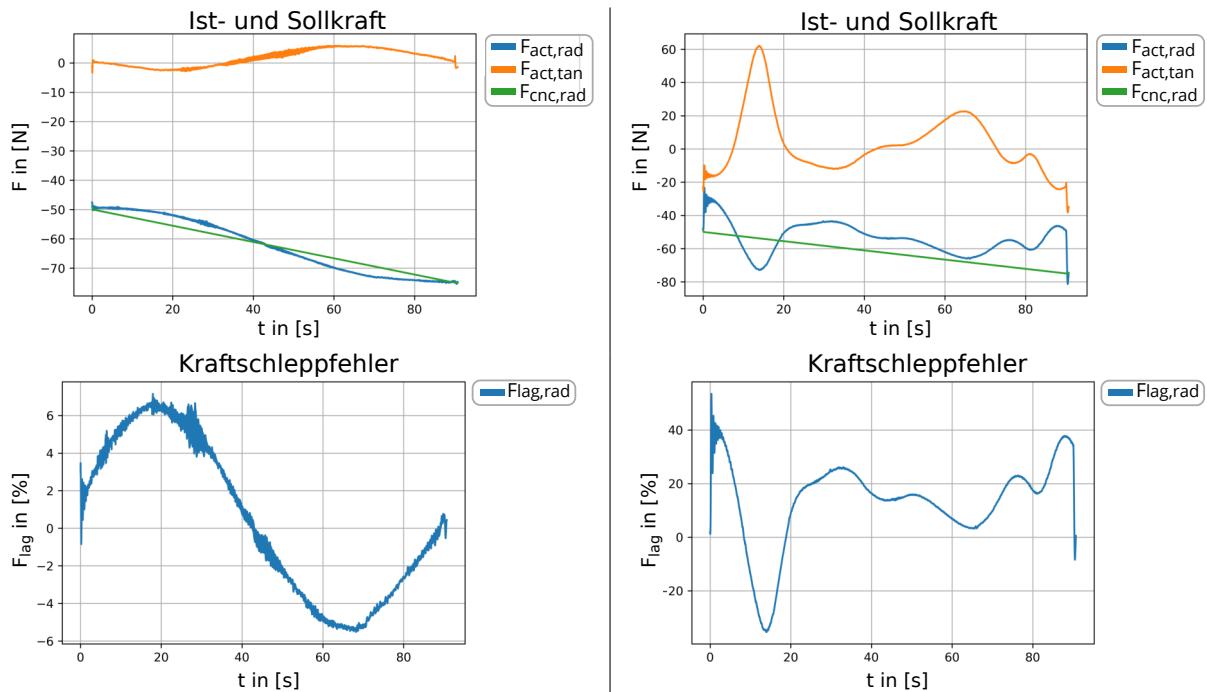


Abbildung 7.12.: Links: Zylinder,  $T = 90$  s, Rechts: Testgeometrie 2,  $T = 90$  s

Die Berechnung des Winkels  $\alpha$  erfolgt über diese Gleichung:

$$\alpha_{\text{norm}} = \text{atan2}(f_{\text{act},y}, f_{\text{act},x}) + \pi, \quad (7.2)$$

Die Berechnung des Winkels  $\alpha$  ist nur im Kontakt möglich. Weil die Kraftsensoren auch in der freien Fahrt rauschen, sind Kontaktenschwellen definiert. Unterhalb der Schwellkräfte wird alpha nach Gleichung 7.1 berechnet. Damit  $\alpha$  in der Nähe der Kontaktenschwelle nicht hin- und herspringt, ist der Umschaltmechanismus hysteresebasiert. Damit die Berechnung des Winkels  $\alpha$  im Kontakt auf Basis der gemessenen Kräfte stattfindet, muss das Verhalten im G-Code aktiviert sein. Das geschieht über M57, wie in Tabelle 4.1 aufgeführt. Dazu muss die Konturverfolgung mit M55 aktiviert sein. Weil das die einzige Änderung im Vergleich zu Quelltext 7.2 darstellt, wird der veränderte G-Code aus Platzgründen nicht noch einmal aufgeführt.

Mit dieser Regelung wird der selbe Prozess noch einmal durchgeführt. Die Ergebnisse sind in Abbildung 7.13 zu sehen.

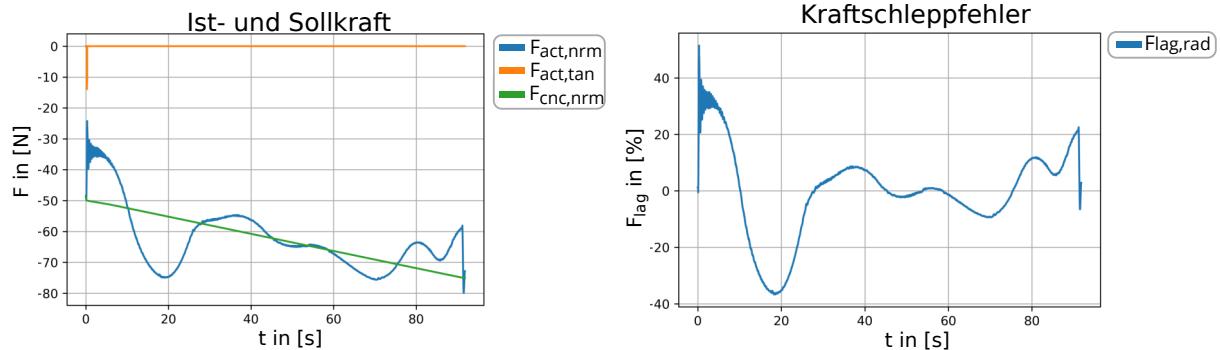


Abbildung 7.13.: Kraftregelung in Normalrichtung

Es fällt auf, dass in der passiven Richtung keine Kräfte mehr auftreten. Außerdem ist der Schleppfehler geringer als in der radialen Regelung. Abbildung 7.14 stellt die Gesamtkräfte in der x-y-Ebene und die Schleppfehler beider Regelungen zusammen dar, um die Vergleichbarkeit zu erhöhen.

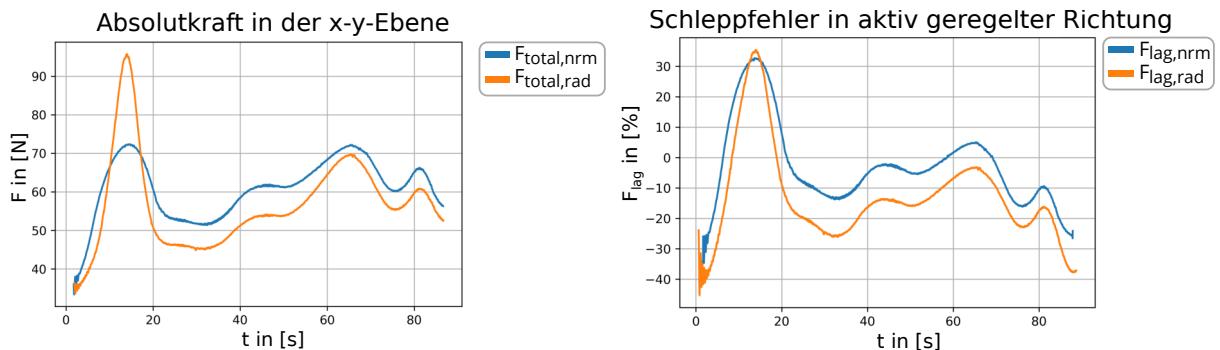


Abbildung 7.14.: Vergleich zwischen Kraftregelung in Radial- und Normalrichtung

Die Darstellung der Absolutkraft verdeutlicht besonders stark, welchen Einfluss die Richtung der aktiven Koordinatenachse auf die Gesamtkraft ausübt. Das Maximum liegt mit etwa 15 s genau an der Stelle, wo der Unterschied zwischen  $a_{norm}$  und  $a_{rad}$  besonders eklatant ist. Deswegen baut sich eine große Blindkraft auf, um die gewünschte Kraft in aktiver Richtung zu erreichen. Die Befürchtung, dass der Prozess durch die Verwendung der Kraftistwerte zur Berechnung von  $\alpha$  instabil wird, hat sich zumindest für niedrige Geschwindigkeiten nicht bewährt. Die Möglichkeit, Kräfte normal zur Reaktionskraft zu regeln, befähigt den Hexapod zu weitaus flexibleren Aufgaben.

## 8. Zusammenfassung

Obwohl das Thema der Kraftregelung die Forschung schon seit mehreren Jahrzehnten beschäftigt, sind kommerzielle Kraftregelungen auf Werkzeugmaschinen kaum verbreitet. Im Rahmen dieser Diplomarbeit fand die Entwicklung und Erprobung einer Kraftregelung am Hexapod *Felix* / der TU Dresden statt. Ziel war es, zu demonstrieren, dass praxisrelevante Prozesse auf einer Werkzeugmaschine kraftgeregt umsetzbar sind. Die Entwicklung der Aufgabenstellung fand unter der Prämisse statt, dass die hohen Sensorkosten, das Fehlen einer weitreichenden Integration einer Kraftregelung in kommerzielle Steuerungssysteme und die komplexe Aufgabenbeschreibung kraftgeregelter Prozesse maßgeblich für die zögerliche kommerzielle Integration von Kraftregelungen verantwortlich sind. Die im Rahmen dieser Arbeit erreichten Entwicklungen leisten zu allen drei Herausforderungen einen Beitrag. Die Priorität liegt dabei deutlich auf der Steuerungsintegration.

Die entwickelte Kraftregelungssoftware ist in das kommerzielle Steuerungssystem *TwinCAT* integriert. Dabei baut das Projekt auf vorangegangenen Arbeiten am IMD auf. Zu Beginn der Arbeit wurde die Infrastruktur des bestehenden Softwareprojekts innerhalb *TwinCATs* erweitert, um eine Vorgabe von Kraftsollwerten im G-Code zu ermöglichen. Dabei gelang es, die CNC zur parallelen Interpolation von Kraft- und Positionssollwerten zu verwenden. Dafür wurde die CNC zur Onlinetransformation der Positionswerte befähigt. Dem schloss sich die Umsetzung eines alternativen Konzepts zum Aufschalten eines Wegoffsets auf die Positionssollwerte der CNC an. Das legte den Grundstein für die eigentliche Entwicklung der Kraftregelung im kartesischen Raum, die komplett per G-Code steuerbar ist. Um eine sichere und transparente Bedienung der Kraftregelung zu gewährleisten, wurde die Regelung in eine Zustandsmaschine eingebettet, welche die Interaktion mit der Kraftregelung verwaltet. Parallel dazu fand die Konzeption und Implementierung eines virtuellen Steuerungs- und Prozessmodells statt, das zu Unit-, Integration- und System-Testing-Zwecken Verwendung findet. Dadurch konnte im Lauf der gesamten Entwicklung vor der realen Inbetriebnahme der Software ein virtueller Probelauf stattfinden. Das bedeutet eine große Steigerung der Flexibilität und Sicherheit der Entwicklung.

Nach erfolgreicher Inbetriebnahme an der realen Maschine fanden Untersuchungen zu Ma-

schinenverhalten und Regelbandbreite statt. Anschließend wurde ein Experiment konzipiert, das praxisrelevante Problematiken fertigungstechnischer Prozesse abbildet und dabei trotzdem über einen einfachen Aufbau verfügt. Die Wahl für das Experiment fiel auf einen Konturverfolgungsprozess, der zunächst im kartesischen Raum stattfand. Dabei erwies sich die strukturintegrierte Kraftmessung tauglich für die Kraftregelung.

Anschließend wurde die Kraftregelung erweitert, um Kräfte in prozessabhängigen Koordinatensystemen zu regeln. Dazu fand eine Untersuchung radialer und normaler Kraftregelung statt. Dafür wurde das Konturverfolgungsexperiment erweitert. Es konnte gezeigt werden, dass kraftgeregelte Prozesse auch in variablen Aufgabenkoordinatensystemen stabil lauffähig sind.

Die entwickelte Lösung erfüllt die zwei wesentlichen Aspekte der Aufgabenstellung: Die parallele Interpolation von Kraft- und Sollwerten (Anforderung FA-1) funktioniert und die Dynamik der Kraftregelung konnte im Vergleich zur Vorgängerlösung um den Faktor 30 gesteigert werden (Anforderung LA-1), wie in Kapitel 7.1 erläutert.

In Abschnitt 6.2 stellte sich heraus, dass der kraftgeregelte Hexapod schwingungsanfällig ist. Durch eine starke Breitbandfilterung der Kraftwerte konnte die Schwingungen reduziert werden. Deren Wirksamkeit weist darauf hin, dass die Kraftregelung möglicherweise mechanische Eigenfrequenzen anregt. Aus der Filterung resultiert allerdings ein starker Regelbandbreitenverlust. Deswegen fand die Durchführung eines Experimentes zur Untersuchung des Maschinenverhaltens bei variiertem Schwingungsanregung statt. Die Ursache der Schwingungsanfälligkeit konnte damit nicht gefunden werden. Weil mit einer Schwingungsreduktion erhebliche Dynamikzugewinne einhergehen, wären weitere Forschungsarbeiten auf diesem Gebiet sinnvoll. Dafür böte sich eine Modellierung der Regelstrecke und des mechanischen Systems an, um den Auswirkungen der zahlreichen Einflussfaktoren effizient zu untersuchen. Mithilfe einer darauf aufbauenden Analyse der mechanischen und regelungstechnischen Komponenten könnten außerdem partiell wirksame Filter die Dynamik des Systems wesentlich verbessern. Das würde härtere Regelparameter ermöglichen, welche die passive Nachgiebigkeit verringern oder überflüssig machen könnten.

Weiteres Verbesserungspotential bietet die Aufgabendefinition. Im Moment ist es notwendig die Transformation des *Task Spaces* in C++ zu definieren. Der Nutzer kann sie im G-Code lediglich an- oder abwählen und parametrieren. Wünschenswert wäre es jedoch, die Definition flexibel im G-Code vorzunehmen.

Grundsätzlich sind alle Funktionen für die Momentenregelung vorhanden, müssten jedoch noch praktisch validiert werden.

Durch die Integration in *TwinCAT* konnte gezeigt werden, dass und wie die Umsetzung einer Kraftregelung auf einem kommerziellen Steuerungssystem möglich ist. Dadurch sinkt die Schwelle zur Verwendung in Werkzeugmaschinen erheblich. Besonders die in Anforderung FA-2 formulierte und erfolgreich implementierte Steuerung per G-Code sorgt dafür, dass die

Kraftregelung leicht in bestehende Arbeitsprozesse integriert werden kann. Weil die Regelung auch mit der strukturintegrierten Kraftmessung gute Ergebnisse erzielt, bestätigt das die Leistungsfähigkeit dieser Messeinrichtungsart. Die damit einhergehende Kostenreduktion macht die Kraftregelung attraktiver. Durch die Verwendung prozessspezifischer, mitgedrehter Koordinatensysteme nach Anforderungen FA-5 und FA-6 erleichtert sich die Definition von Kraftreglungsaufgaben in Oberflächennormalenrichtung.

Auch wenn die hier entwickelte Kraftregelung weit entfernt vom Einsatz in der Industrie ist, ebnet sie jedoch ein Stück des Weges zum Einsatz an der Werkzeugmaschine.

# A. Standardwerte der Kanalparameter

Tabelle A.1.: Standardwerte für die Kanalparameter V.G.KIN\_STEP[0].ID[trafold].PARAM[1-6] (Koordinatensystem Basis):

1	2	3	4	5	6
x	y	z	a	b	c
0	0	0	0	0	0

Tabelle A.2.: Standardwerte für die Kanalparameter V.G.KIN\_STEP[0].ID[trafold].PARAM[7-12] (Koordinatensystem Werkzeug):

7	8	9	10	11	12
x	y	z	a	b	c
0	0	0	0	0	0

Tabelle A.3.: Standardwerte für die Kanalparameter V.G.KIN\_STEP[0].ID[trafold].PARAM[13-18] (Kraftselektionsvektor):

13	14	15	16	17	18
x	y	z	a	b	c
0	0	0	0	0	0

Tabelle A.4.: Standardwerte für die Kanalparameter V.G.KIN\_STEP[0].ID[trafold].PARAM[19-24] (Untere Begrenzung des zur Kraftregelung freigegebenen Arbeitsraumes):

19	20	21	22	23	24
x	y	z	a	b	c
-400 mm	-400 mm	-295 mm	-20°	-20°	-180°

Tabelle A.5.: Standardwerte für die Kanalparameter V.G.KIN\_STEP[0].ID[trafold].PARAM[25-30] (Obere Begrenzung des zur Kraftregelung freigegebenen Arbeitsraumes):

25	26	27	28	29	30
x	y	z	a	b	c
400 mm	400 mm	200 mm	20°	20°	180°

Tabelle A.6.: Standardwerte für die Kanalparameter V.G.KIN\_STEP[0].ID[trafold].PARAM[31-36] (Weiche Kraftreglerparameter, Tn):

25	26	27	28	29	30
x	y	z	a	b	c
10s	10s	10s	1 001 s	1 001 s	1 001 s

Tabelle A.7.: Standardwerte für die Kanalparameter V.G.KIN\_STEP[0].ID[trafold].PARAM[37-42] (Weiche Kraftreglerparameter, Kp):

37	38	39	40	41	42
x	y	z	a	b	c
2 mm/N	2 mm/N	2 mm/N	1 001 rad/N	1 001 rad/N	1 001 rad/N

Tabelle A.8.: Standardwerte für die Kanalparameter V.G.KIN\_STEP[0].ID[trafold].PARAM[43-48] (Harte Kraftreglerparameter, Tn):

43	44	45	46	47	48
x	y	z	a	b	c
3 s	3 s	5 s	1 001 s	1 001 s	1 001 s

Tabelle A.9.: Standardwerte für die Kanalparameter V.G.KIN\_STEP[0].ID[trafold].PARAM[49-54]  
(Harte Kraftreglerparameter, Kp):

49	50	51	52	53	54
x	y	z	a	b	c
15 mm/N	15 mm/N	25 mm/N	1 001 rad/N	1 001 rad/N	1 001 rad/N

# Literaturverzeichnis

- [1] A. Winkler: *Ein Beitrag zur kraftbasierten Mensch-Roboter-Interaktion*. Dissertation, Technische Universität Chemnitz, 2006.
- [2] A. R. Mohanty: *Machinery Condition Monitoring: Principles and Practices*. CRC Press, 2014.
- [3] P. Fixell, T. Groth, M. Isaksson, H. Zhang, J. Wang, J. He, M. Kohlmaier, R. Krappinger und J. Ge: Roboter mit Fingerspitzengefühl. *ABB Technik*, 2007.
- [4] Beckhoff Automation GmbH & Co. KG: Infosys. url: [www.infosys.beckhoff.com](http://www.infosys.beckhoff.com) (besucht am 25. 02. 2021).
- [5] C. Friedrich, B. Kauschinger und S. Ihlenfeldt: Decentralized structure-integrated spatial force measurement in machine tools. *Mechatronics*, 2016.
- [6] R. Vogt: *Entwicklung und Umsetzung einer Kraftregelung mit strukturintegrierter Kraftsensorik an einem Hexapod*. Magisterarbeit, TU Dresden, 2017.
- [7] T. Wiese: Externe Sollwertaufschaltung für eine CNC-Steuerung, 2017.
- [8] B. Siciliano und O. Khatib: *Handbook of Robotics*. Springer, 2016.
- [9] K. Jenschek: Vorlesungsfolien der Veranstaltung "Steuerung von Robotersystemen", 2005.
- [10] S. Szatmari: *Kinematic Calibration of Parallel Kinematic Machines on the Example of the Hexapod of Simple Design*. Dissertation, TU Dresden, 2007.
- [11] J. Wollnack: Robotik (Analyse, Modellierung und Identifikation). url: <https://www.tuhh.de/ft2/wo/Scripts/Robotik/HomogeneTransform.pdf> (besucht am 10. 03. 2021).
- [12] K. Großmann: Die Entwicklung spanender Werkzeugmaschinen. K. Voge, Herausgeber. 2016. url: <http://voge-online.de/> (besucht am 04. 11. 2020).
- [13] Physik Instrumente (PI) GmbH & Co. KG: H-850 6-axis hexapod. url: <https://www.physikinstrumente.com/en/products/parallel-kinematic-hexapods/h-850-6-axis-hexapod-700800/> (besucht am 04. 11. 2020).
- [14] J. Tlusty, J. Ziegert und S. Ridgeway: Fundamental comparison of the use of serial and parallel kinematics for machines tools. *CIRP Annals*, 1999.

- [15] MAV Maschinen, Anlagen, Verfahren: Hilfe, die Hexapoden kommen! Technischer Bericht, 1997.
- [16] S. Bellakehal, N. Andreff, Y. Mezouar und M. Tadjine: Commande vision/force de robots parallèles. *Journal Européen des Systèmes Automatisés*, 2010.
- [17] R. Neugebauer: *Parallelkinematische Maschinen*. Springer Berlin Heidelberg, 2005.
- [18] J.-P. Merlet: *Parallel Robots*. Springer-Verlag GmbH, 2005.
- [19] B. Siciliano, L. Sciavicco, L. Villani und G. Oriolo: *Robotics*. Springer London Ltd, 2008.
- [20] Industrieanzeiger: High-speed durch hexapod-systeme. url: <https://industrieanzeiger.industrie.de/allgemein/high-speed-durch-hexapod-systeme/> (besucht am 05.11.2020).
- [21] Symetrie SARL. url: <https://symetrie.fr/en/> (besucht am 05.11.2020).
- [22] Cornell University: The amiba hexapod telescope mount. url: <https://arxiv.org/abs/0902.2335> (besucht am 05.11.2020).
- [23] B. Salah, M. Strehle und B. Noche: Entwicklung eines Konzepts zum Lagermanagement mit Optimierungsansätzen auf Ebene eines neuen Regalbediengerätes. In *Logistics Journal*, 2012.
- [24] Y. Huang, D. M. Pool, O. Stroosma, Q. P. Chu und M. Mulder: Modeling and simulation of hydraulic hexapod flight simulator motion systems. In *AIAA Modeling and Simulation Technologies Conference*, 2016.
- [25] G. Parma: Overview of the nasa docking system and the international docking system standard. In *AIAA Annual Technical Symposium*, 2011.
- [26] M. Eidelman und A. Katzman: Treatment of complex tibial fractures in children with the taylor spatial frame. *Orthopedics*, 2011.
- [27] D. Vollmer, U. Ehmer, C. Bourauel und G. Linß: Planung orthognather Chirurgie mit dem Hexapod - Planning of Orthognathic Surgery with the Hexapod System. *Biomedizinische Technik/Biomedical Engineering*, 2001.
- [28] F. Schlüter: *Entwurf und Umsetzung erweiterter Positions- und Kraftreglerprinzipien im CNC-Kern für eine Zweiarms-Kinematik*. Studienarbeit, TU Dresden, 2020.
- [29] M. Weck und C. Brecher.: *Werkzeugmaschinen 4: Automatisierung von Maschinen und Anlagen*. Springer, 2006.
- [30] T. Yoshikawa: Force control of robot manipulators. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation*.
- [31] V. Rusin: *Adaptive Regelung von Robotersystemen in Kontaktaufgaben*. Dissertation, Otto-von-Guericke-Universität Magdeburg, 2007.
- [32] D. H. Ballard: Task frames in robot manipulation. *AAAI-84*, 1984.
- [33] M. Mason: Robot motion: planning and control edited m.i.t. press. *Robotica*, 1983.

- [34] T. Reisinger: *Kontaktregelung von Parallelrobotern auf der Basis von Aktionsprimitiven*. Dissertation, Technische Universität Carolo-Wilhelmina zu Braunschweig, 2008.
- [35] A. Winkler: *Sensorgeführte Bewegungen stationärer Roboter*. Habilitation, Technische Universität Chemnitz, 2016.
- [36] C. G. Liang und G. M. Lance: A differentiable null space method for constrained dynamic analysis. *Journal of Mechanisms, Transmissions, and Automation in Design*, 1987.
- [37] A. Spiller: *Unterstützung der Werkstückhandhabung kooperierender Industrieroboter durch Kraftregelung*. Dissertation, Universität Stuttgart, 2014.
- [38] W. Commons: File:dehnungsmessstreifen.svg — wikimedia commons, the free media repository, 2020. url: <https://commons.wikimedia.org/w/index.php?title=File:Dehnungsmessstreifen.svg&oldid=510208534> (besucht am 10.11.2020).
- [39] H. Schaumburg: *Sensoranwendungen*. Vieweg+Teubner Verlag, Wiesbaden, 1995.
- [40] P. Roberts, D. D. Damian, W. Shan, T. Lu und C. Majidi: Soft-matter capacitive sensor for measuring shear and pressure deformation. In *2013 IEEE International Conference on Robotics and Automation*, 2013.
- [41] piezosystem jena GmbH: Piezoline. url: <https://www.piezosystem.de/piezopedia/piezotheorie/> (besucht am 10.11.2020).
- [42] Kistler Instrumente AG: 6-Achsen Piezo Kraft-/Momenten Sensor. url: <https://www.kistler.com/de/produkt/type-9306a31/> (besucht am 10.11.2020).
- [43] T. Kleckers: Piezoelektrische Kraftaufnehmer: Das Prinzip ist einfach – die Möglichkeiten riesig. HBM. url: <https://www.hbm.com/de/7318/wie-funktioniert-eigentlich-ein-piezo-kraftaufnehmer/> (besucht am 30.11.2020).
- [44] ME-Meßsysteme GmbH: Kd80s. url: <https://www.me-systeme.de/shop/de/sensoren/kraftsensoren/kds/kd80s12> (besucht am 10.11.2020).
- [45] C. Friedrich und K. Großmann: Strukturintegrierte Kraftmessung. *ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb*, 2016.
- [46] M. Callegari und A. Suardi: On the force-controlled assembly operations of a new parallel kinematics manipulator. In *6th IEEE Mediterranean Conf. on Control and Automation*, 2003.
- [47] J. le Flohic: *Vers une commande basée modèle des machines complexes : application aux machines-outils et machines d'essais mécaniques*. Dissertation, Université Blaise Pascal - Clermont II, 2015.
- [48] P. Gebhard: *Dynamisches Verhalten von Werkzeugmaschinen bei Anwendung für das Rührreibschweißen*. Dissertation, TU München, 2010.
- [49] ABB Ltd: Robotware assembly fc. url: <https://new.abb.com/products/robotics/application-software/assembly/robotware-assembly-fc>.
- [50] N. Zemiti: *Commande en Effort des Systèmes Robotiques pour la Chirurgie Mini-Invasive*. Dissertation, UNIVERSITE PARIS 6, 2005.

- [51] Q. V. Dang: *Conception et commande d'une interface haptique à retour d'effort pour la CAO*. Dissertation, Université De Valenciennes et du Hainaut-cambresis, 2013.
- [52] J. N. Howell, R. R. Conatser, R. L. Williams, J. M. Burns und D. C. Eland: The virtual haptic back: a simulation for training in palpitory diagnosis. *BMC Medical Education*, 2008.
- [53] B. Kauschinger: *Verbesserung der Bewegungsgenauigkeit an einem Hexapod einfacher Bauart*. Inst. für Werkzeugmaschinen und Steuerungstechnik, Lehrstuhl für Werkzeugmaschinen, 2006.
- [54] C. Friedrich, B. Kauschinger und S. Ihlenfeldt: Stiffness evaluation of a hexapod machine tool with integrated force sensors. *Journal of Machine Engineering*, 2020.
- [55] ALTHEN Sensors & Controls Inc: Cylindrical force sensor alf256. url: <https://www.althensensors.com/sensors/force-sensors/cylindrical-force-sensors/4512/alf256-cylindrical-force-sensor/> (besucht am 13.11.2020).
- [56] ATI Industrial Automation: 6d kraftsensor omega 190. url: [https://www.ati-ia.com/products/ft/ft\\_models.aspx?ID=Omega190](https://www.ati-ia.com/products/ft/ft_models.aspx?ID=Omega190) (besucht am 13.11.2020).
- [57] R. Barth: Echtzeitverarbeitung – die Basis für PC-Control. Technischer Bericht, 2011. url: <https://www.elektrotechnik.vogel.de/echtzeitverarbeitung-die-basis-fuer-pc-control-a-335589> (besucht am 19.02.2021).
- [58] Industrielle Steuerungstechnik GmbH: Mccom dokumentation trafo anbindung einer kinematischen transformation. url: <https://www.isg-stuttgart.de/> (besucht am 31.01.2021).
- [59] Industrielle Steuerungstechnik GmbH: Handbuch Kanalparameter. url: <https://www.isg-stuttgart.de/> (besucht am 31.01.2021).
- [60] Industrielle Steuerungstechnik GmbH: Handbuch Achsparameter. url: <https://www.isg-stuttgart.de/> (besucht am 31.01.2021).
- [61] C. Friedrich, B. Kauschinger und S. Ihlenfeldt: Spatial force measurement using a rigid hexapod-based end-effector with structure-integrated force sensors in a hexapod machine tool. *Measurement*, 2019.
- [62] A. Burri: Tinyfsm. Digital Integrity GmbH. url: <https://github.com/digint/tinyfsm> (besucht am 17.12.2020).
- [63] R. Martin: *Clean coder Verhaltensregeln für professionelle Programmierer*. Addison-Wesley, 2011.
- [64] T. Linz: *Testing in Scrum*. Rocky Nook, 2014.
- [65] J. G. Ziegler und N. B. Nichols: Optimum settings for automatic controllers. *Journal of Dynamic Systems Measurement and Control-transactions of The Asme*, 1942.

# Formelzeichen und Abkürzungen

## Formelzeichen

### Skalare Größen

$M$	$Nm$	Moment
$T$	$s$	Periodendauer
$f$	$f$	Frequenz

### Vektorielle Größen

$A$	-	poseabhängige Konfigurationsmatrix
$f_{act}$	$N$ bzw. Nm	Istkraft
$f_{nom}$	$N$ bzw. Nm	Nutzerdefinierte Sollkraft
$f_{cnc}$	$N$ bzw. Nm	Durch die CNC interpolierte Sollkraft
$J_{kin}$	-	poseabhängige kinetische Jacobimatrix
$J_{geo}$	-	poseabhängige geometrische Jacobimatrix
$O$	mm	Ursprung eines Koordinatensystems
$P$	mm	Koordinaten eines Punktes
$q$	mm	Koordinaten im Gelenakraum
$R$	-	Rotationsmatrix
$S$	-	Kraftselektionsmatrix
${}^{USR}T_{TS}$	mm bzw. rad	Posetransformation vom Nutzerkoordinatensystem zum Aufgaben-koordinatensystem

---

${}^B T_{USR}$	mm bzw. rad	Posetransformation vom globalen Maschinenkoordinatensystem zum Nutzerkoordinatensystem
${}^{USR} T_{TS}^f$	mm bzw. rad	Kraftransformation vom Nutzerkoordinatensystem zum Aufgaben- koordinatensystem
${}^B T_{USR}^f$	mm bzw. rad	Kraftransformation vom globalen Maschinenkoordinatensystem zum Nutzerkoordinatensystem
$x$	-	Koordinaten im kartesischen Raum
$x_{act}$	mm bzw. rad	kartesische Istpose
$x_{nom}$	mm bzw. deg	kartesische Nutzerdefinierte Sollpose
$x_{lag}$	mm bzw. deg	kartesischer Wegschleppfehler
$x_{cnc}$	mm bzw. rad	kartesische Durch die CNC interpolierte Sollpose
$\Delta x$	mm bzw. rad	kartesischer Poseoffset

## Abkürzungen

CNC	Computerized Numerical Control
SPS	SpericherProgrammierbare Steuerung
TwinCAT	Steuerungssystems der Beckhoff Automation GmbH & Co. KG
TCP	Tool Center Point
DIN	Deutsche IndustrieNorm
DK	Direkte Kinematik
IK	Inverse Kinematik
IMD	Institut für Mechatronischen Maschinenbau
IEC	International Electrotechnical Commission
ADS	Automation Device Specification der Beckhoff Automation GmbH & Co. KG
CAN-Bus	Controller Area Network-Bussystem von Robert Bosch GmbH und Intel Corporation
PID-Regler	Proportional-Integrational-Derivative-Regler
CAD	Computer Aided Design
IO-mapping	Input/Output-mapping
NC	Numerical Control
AR	Arbeitsraum
GR	Gelenkraum
KIN-ID	Kinematische Identifikationsnummer

TcCOM	TwinCAT Component Object Model der Beckhoff Automation GmbH & Co. KG
HLI	HighLevelInterface der Beckhoff Automation GmbH & Co. KG

# Abbildungsverzeichnis

2.1. Veranschaulichung 3D-Pose . . . . .	5
2.2. modifizierte Eulerwinkel B,L & D [10]. . . . .	6
2.3. Links: Entkoppelte, serielle Kinematik einer konventionellen Werkzeugmaschine, Freiheitsgrade in rot [12], rechts: Parallelkinematik Hexapod [13] . . . . .	7
2.4. Aufgabenteilung zwischen CNC und SPS . . . . .	10
2.5. Dehnmessstreifen [38] . . . . .	12
2.6. Aufbau eines kapazitiven Kraftsensors . . . . .	13
2.7. Piezoelektrischer Effekt . . . . .	13
2.8. Links: 6D-Variante von Kistler [42], rechts: 1D-Variante von ME-Systeme [44] . .	14
2.9. Objekt mit im Punkt $M$ angreifender Kraft $F_M$ und Moment $M_M$ . . . . .	14
2.10. direkte, parallel-hybride Kraftregelung im kartesischen Raum mittels invertierter Jacobimatrix[28] . . . . .	16
2.11. direkte, parallel-hybride Kraftregelung im kartesischen Raum mittels invertierter Jacobimatrix[28] . . . . .	16
2.12. direkte, parallel-hybride Kraftregelung im kartesischen Raum mittels vollständig inverser Kinematik[28] . . . . .	17
2.13. CAD-Darstellung des Hexapods <i>Felix I</i> der TU Dresden [53] . . . . .	19
2.14. Minimaldarstellung der kinematischen Kette als Basis für die Transformationen [53] . . . . .	20
2.15. Kraftsensoren des Hexapods <i>Felix I</i> [45] . . . . .	21
2.16. Am Hexapod verwendete Kraftmessdosen, Links: 1D-Kraftsensor <i>ALF256</i> [55] für Variante (1) und (2), rechts: 6D-Kraftsensor <i>Omega190</i> [56] für Variante (R) . . . . .	21
2.17. Tisch im Arbeitsraum des Hexapods . . . . .	22
2.18. Schematische Darstellung der Modulkommunikation in <i>TwinCAT</i> . . . . .	23
2.19. Abarbeitung der Module in einem Zyklus in <i>TwinCAT</i> nach [4] . . . . .	23
2.20. Signalflussplan des Ausgangszustandes [6] . . . . .	26
4.1. Koordinatensysteme am Hexapod mit dazugehörigen Transformationen. Be- nutzerdefinierte Koordinatensysteme sind mit dem Index $usr$ gekennzeichnet . .	34

5.1.	Signalflussdiagramm des Gesamtsystems, die Klasse <i>ForceManager</i> implementiert die Kraftregelung . . . . .	36
5.2.	Signalflussdiagramm der Kraft- und Positionsdaten . . . . .	37
5.3.	Aktivieren und Deaktivieren der Trafo per G-Code [58] . . . . .	39
5.4.	UML-Klassendiagramm der Klasse <i>ForceManager</i> , zur besseren Übersichtlichkeit leicht gekürzt . . . . .	42
5.5.	UML-Aktivitätsdiagramm der Funktion <i>CycleUpdate</i> , zur besseren Übersichtlichkeit leicht gekürzt . . . . .	43
5.6.	UML-Sequenzdiagramm für die Interaktion innerhalb der Klasse <i>ForceManager</i> , zur besseren Übersichtlichkeit leicht gekürzt . . . . .	44
5.7.	Zustände der Klasse <i>StateMachine</i> , Eingangssignale in Orange . . . . .	46
5.8.	Zur besseren Übersichtlichkeit leicht gekürztes UML-Sequenzdiagramm der Kontaktsuche . . . . .	47
5.9.	Zur besseren Übersichtlichkeit leicht gekürztes UML-Sequenzdiagramm der Kraftregelung . . . . .	48
5.10.	Zur besseren Übersichtlichkeit leicht gekürztes UML-Sequenzdiagramm des Zustandes <i>Offset herausfahren</i> . . . . .	49
5.11.	Transformationen für das Aufschalten des Poseoffsets der Kraftregelung auf die Positionssollwerte . . . . .	52
5.12.	Signalfluss und Transformationen im <i>ForceController</i> . . . . .	55
6.1.	Virtuelles Prozess- und Steuerungsmodell . . . . .	58
6.2.	Virtuelles Prozess- und Steuerungsmodell . . . . .	59
6.3.	Beispiel zur Erzeugung einer simulierten Kraft in der x-y-Ebene . . . . .	60
6.4.	virtuelles Experiment, Links: Kraftvorgabe und -istwert, Rechts: relative Regelabweichung . . . . .	62
6.5.	Links: Versuchsaufbau der realen Inbetriebnahme, rechts: Endeffektor . . . . .	63
6.6.	Umbau zur Erhöhung der Nachgiebigkeit . . . . .	64
6.7.	Eine Periode der Sollkraftänderung für die Reglereinstellung,Links: Kraftvorgabe und -istwert, Rechts: relative Regelabweichung . . . . .	65
6.8.	Kraftverläufe im Störbetrieb, Links: Kraftvorgabe und -istwert, Rechts: relative Regelabweichung . . . . .	66
6.9.	Links: Hystereseeffekt in der Istkraft des Versuchs zum Führungsverhalten, rechts: Schwingungen bei Kräften unterhalb von 100 N . . . . .	66
6.10.	Bodediagramm des Schwingungsversuchs . . . . .	67
7.1.	Vertikale Testgeometrie, Links: Halterung, Rechts: Versuchsaufbau . . . . .	70
7.2.	Konturverfolgung der Testgeometrie 1 ohne Vorsteuerung . . . . .	72
7.3.	Konturverfolgung der Testgeometrie 1 mit Vorsteuerung . . . . .	72
7.4.	Kraftverläufe der Konturverfolgung in z-Richtung mit konventionellem Kraftsensor ohne Vorsteuerung, Links: Kraftvorgabe und -istwert, Rechts: relative Regelabweichung . . . . .	73

7.5. Links: Querkraft während der Konturverfolgung durch Kraftregelung in z-Richtung aus Abschnitt 7.1, Rechts: Aus der Querkraft resultierende Verkipfung ohne Führung . . . . .	74
7.6. Links: Koordinatensysteme der radialen Kraftregelung, $TS_N$ steht für Task Space in Oberflächennormalenrichtung und $TS_R$ in Radialrichtung, Rechts: eingespannter flacher Zylinder für das Experiment der Konturverfolgung . . . . .	76
7.7. Schwarz: Sollposition, Bunt: Istposition mit Kraftvorgabe an jedem Punkt . . . . .	77
7.8. Infolge der Exzentrizität schwankende Kraftschleppfehler . . . . .	78
7.9. Relative Kraftschleppfehler bei variiertem Prozesszeit . . . . .	78
7.10. Koordinatensysteme der Kraftregelung in Oberflächennormalenrichtung, $TS_N$ steht für Task Space in Normalenrichtung und $TS_R$ in Radialrichtung . . . . .	79
7.11. Testgeometrie 2, Links: Photo, Rechts: Repräsentation aus Konturverfolgung . .	79
7.12. Links: Zylinder, $T = 90\text{ s}$ , Rechts: Testgeometrie 2, $T = 90\text{ s}$ . . . . .	80
7.13. Kraftregelung in Normalrichtung . . . . .	81
7.14. Vergleich zwischen Kraftregelung in Radial- und Normalrichtung . . . . .	81

# Tabellenverzeichnis

2.1. Module der Steuerung im Ausgangszustand . . . . .	24
4.1. verfügbare Befehle zur Steuerung der Kraftregelung . . . . .	33
5.1. Übersicht über die im Rahmen der <i>Man-in-the-middle-Aktion</i> eingelesenen Daten	52
5.2. Übersicht über die im Rahmen der <i>Man-in-the-middle-Aktion</i> geschriebenen Daten	53
5.3. Standardwerte der Schwellkräfte und -momente für die Kontaktsuche . . . . .	54
5.4. Standardwerte der Kraft- und Momentenlimits . . . . .	54
7.1. Vergleich der zwei Varianten . . . . .	73
A.1. Standardwerte für die Kanalparameter V.G.KIN_STEP[0].ID[trafold].PARAM[1-6] (Koordinatensystem Basis): . . . . .	85
A.2. Standardwerte für die Kanalparameter V.G.KIN_STEP[0].ID[trafold].PARAM[7-12] (Koordinatensystem Werkzeug): . . . . .	85
A.3. Standardwerte für die Kanalparameter V.G.KIN_STEP[0].ID[trafold].PARAM[13- 18] (Kraftselektionsvektor): . . . . .	85
A.4. Standardwerte für die Kanalparameter V.G.KIN_STEP[0].ID[trafold].PARAM[19- 24] (Untere Begrenzung des zur Kraftregelung freigegebenen Arbeitsraumes): .	86
A.5. Standardwerte für die Kanalparameter V.G.KIN_STEP[0].ID[trafold].PARAM[25- 30] (Obere Begrenzung des zur Kraftregelung freigegebenen Arbeitsraumes): .	86
A.6. Standardwerte für die Kanalparameter V.G.KIN_STEP[0].ID[trafold].PARAM[31- 36] (Weiche Kraftreglerparameter, Tn): . . . . .	86
A.7. Standardwerte für die Kanalparameter V.G.KIN_STEP[0].ID[trafold].PARAM[37- 42] (Weiche Kraftreglerparameter, Kp): . . . . .	86
A.8. Standardwerte für die Kanalparameter V.G.KIN_STEP[0].ID[trafold].PARAM[43- 48] (Harte Kraftreglerparameter, Tn): . . . . .	86
A.9. Standardwerte für die Kanalparameter V.G.KIN_STEP[0].ID[trafold].PARAM[49- 54] (Harte Kraftreglerparameter, Kp): . . . . .	87

## EIDESSTATTLICHE ERKLÄRUNG

Hiermit versichere ich, Frederik Schlüter, die vorliegende Arbeit selbstständig, ohne fremde Hilfe und ohne Benutzung anderer als der von mir angegebenen Quellen angefertigt zu haben. Alle aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche gekennzeichnet.

Die Arbeit wurde noch keiner Prüfungsbehörde in gleicher oder ähnlicher Form vorgelegt.

Ort, Datum

Unterschrift