STUTTGARTER BEITRÄGE ZUR PRODUKTIONSFORSCHUNG

Lorenz Halt

»Reglersynthese für aufgabenraumgesteuerte Industrieroboter«

Fraunhofer IPA

Universität Stuttgart

STUTTGARTER BEITRÄGE ZUR PRODUKTIONSFORSCHUNG BAND 136

Lorenz Halt

»Reglersynthese für aufgabenraumgesteuerte Industrieroboter«

Herausgeber

Univ.-Prof. Dr.-Ing. Thomas Bauernhansl[1,2]
Univ.-Prof. Dr.-Ing. Dipl.-Kfm. Alexander Sauer[1,3]
Univ.-Prof. Dr.-Ing. Kai Peter Birke[4]
Univ.-Prof. Dr.-Ing. Marco Huber[1,2]

[1]Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA, Stuttgart
[2]Institut für Industrielle Fertigung und Fabrikbetrieb (IFF) der Universität Stuttgart
[3]Institut für Energieeffizienz in der Produktion (EEP) der Universität Stuttgart
[4]Institut für Photovoltaik (*ipv*) der Universität Stuttgart

Kontaktadresse:
Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA
Nobelstr. 12
70569 Stuttgart
Telefon 0711 970-1101
info@ipa.fraunhofer.de
www.ipa.fraunhofer.de

Bibliographische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliographie; detaillierte bibliografische Daten sind im Internet über http://dnb.de abrufbar.

Zugl.: Stuttgart, Univ., Diss., 2022

D 93

2022

Druck und Weiterverarbeitung:
Fraunhofer Verlag, Mediendienstleistungen, Stuttgart, 2022
Für den Druck des Buches wurde chlor- und säurefreies Papier verwendet.

Reglersynthese für aufgabenraumgesteuerte Industrieroboter

Control synthesis for task-space controlled industrial robots

Von der Fakultät Konstruktions-, Produktions- und Fahrzeugtechnik
der Universität Stuttgart
zur Erlangung der Würde eines Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Abhandlung

Vorgelegt von
Lorenz Halt
geboren in Rüsselsheim

Hauptberichter:	Univ.-Prof. Dr.-Ing. Alexander Verl
Mitberichter:	Prof. Dr.-Ing. Rainer Müller
Tag der mündlichen Prüfung:	25. Februar 2022

Institut für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen (ISW)
der Universität Stuttgart

2022

Vorwort des Autors

Die vorliegende Arbeit entstand im Rahmen meiner Tätigkeit in der Gruppe Roboterprogrammierung und -regelung der Abteilung Roboter- und Assistenzsysteme am Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA. Eine Dissertation ist zu großen Teilen eine Einzelleistung, und doch ist sie ohne die mitunter vielfältige Unterstützung im beruflichen wie auch privaten Kontext nicht möglich. Demensprechend haben auch zu meiner Dissertation einige Menschen maßgeblich beigetragen, sodass ich sie erfolgreich abschließen konnte und nun diese Danksagung als Schlussakt formulieren darf.

Zunächst gilt mein besonderer Dank Univ.-Prof. Dr.-Ing. Alexander Verl für die Möglichkeit, diese Arbeit unter seiner Betreuung zu erstellen, und Prof. Dr.-Ing. Rainer Müller für die Übernahme des Korreferats.

Zusätzlich bedanke ich mich sehr herzlich bei PD Dr.-Ing. Andreas Pott, der ebenfalls lange Jahre in unserer Abteilung tätig war. Seine steten Bemühungen, meinen wissenschaftlichen Prozess zu begleiten und zu unterstützen, führten zunächst zu den Veröffentlichungen, die meiner Dissertation zugrunde liegen. Letztendlich hat seine Rolle ähnlich einem Mentor die Dissertation selbst erst möglich gemacht. Ich schätze seine Expertise, seine klaren Worte und unser freundschaftliches Verhältnis. Ich bin sehr froh, dass ich getreu dem Motto „zur rechten Zeit am rechten Ort“ seine Unterstützung gewonnen habe.

Dieses Motto hat überdies weitere besondere Zusammenarbeiten ermöglicht. Bedanken möchte ich mich deshalb auch bei meinen Kollegen aus unserer Forschungsgruppe, insbesondere bei Frank Nägele, Philipp Tenbrock und Philipp Gorczak, für ihre Unterstützung. Gemeinsam haben wir mit unseren Erfahrungen und unserem Engagement die Grundlage geschaffen, die das systematische Ansteuern eines Roboters für uns überhaupt erst möglich gemacht hat. Nur dank ihrer vorbehaltlosen Unterstützung und Motivation war es möglich, alle Rückschläge konstruktiv zu überwinden und dennoch ins Ziel zu kommen.

Besonderer Dank gilt nicht zuletzt meinen Eltern und Schwiegereltern, die mich auf meinem Weg stets unterstützt haben, und meiner Frau Anne, die mir den Freiraum geschaffen und manche Nacht- oder Wochenendschicht ermöglicht hat, um diese Arbeit zu vervollständigen. Sie hat niemals den Glauben an mich oder mein Ziel verloren, was mich sehr im Weitermachen

bestärkt hat. Ohne die familiäre Unterstützung hätte ich es nicht geschafft, die Dissertation fertigzustellen. Danke!

Stuttgart, im März 2022 Lorenz Halt

Kurzinhalt

Die Motivation der vorliegenden Dissertation ist es ein performantes sowie roboter- und kinematikneutrales Steuerungsrahmenwerk für roboterbasierte Montage zu schaffen. Dieses Rahmenwerk soll einfach zu programmieren sein und die Möglichkeit aufweisen Programme zwischen Robotern zu übertragen. Hierfür wurde eine *skillbasierte Programmierung* mit dem iTaSC Formalismus kombiniert. Darauf aufbauend wurden als Hauptteil dieser Arbeit Reglerstrukturen entwickelt, die sich ohne Zutun des Programmierers eigenständig parametrisieren und sich somit automatisch an den eingesetzten Roboteraufbau anpassen.

Für kontaktfreie Bewegungen des Roboters wurde ein modellbasierter Regelungsansatz ausgewählt. Zunächst wird ein lineares Modell angenähert und zur automatischen Synthese einer dynamischen Ausgangsrückführung eingesetzt. Das Verfahren ermöglicht nahezu zeitoptimales Verhalten unter Berücksichtigung von Stellgrößenbegrenzungen. Für die Kontakt- und Kraftregelung wurde ein *modellfreier* Ansatz verfolgt. Hierbei wird die Reglerverstärkung basierend auf den aktuellen Regelungsfehler so adaptiert, dass sich ein Regelfehlerverlauf innerhalb vorgegebener Performanzgrenzen ergibt.

Die Regelungsansätze wurden einzeln in Simulationen verifiziert, in das iTaSC basierte Rahmenwerk eingefügt und jeweils mit verschiedenen Szenarien und Robotern experimentell erprobt. Es ergeben sich sowohl neue Einblicke in die Verhalten der einzelnen Technologien, als auch in das Zusammenspiel der Komponenten des dargestellten Steuerungsrahmenwerks. Beide Regelungsansätze ermöglichen hohe Regelgüte und große Übertragbarkeit für komplexe Roboterbewegungen bei Montageaufgaben. Die Ansätze benötigen keine manuellen Anpassungen und ermöglichen so die Programmierung durch Prozessexperten ohne tiefere Kenntnisse der Regelungstechnik.

Short Summary

The motivation of this dissertation is to create a high-performance robotic and kinematic-neutral control framework for robot-based assembly. This framework should be easy to program and have the ability to transfer programs between robots. For this a *skill-based programming* was combined with the iTaSC formalism. On this basis, as the main part of this work, controllers were developed which parameterize independently without the intervention of the programmer and thus automatically adapt to the used robot.

For non-contact movements of the robot, a model-based control approach was selected. First, a linear model is approximated and used to automatically synthesize a dynamic output feedback. The method allows almost time-optimal behavior taking into account setpoint limitations. For contact and force control a *model-free* approach was followed. In this case the controller gains are adapted based on the current control error so that that given performance limits are met.

The control approaches were individually verified in simulations, inserted into the iTaSC-based framework and verified with different scenarios and robots. This results in new insights into the behavior of the individual technologies as well as in the interplay of the presented control framework. Both control approaches allow high control quality and high transferability for complex robot movements for assembly tasks. The approaches do not require manual adjustments and thus allow programming by process experts without deeper knowledge of control engineering.

Inhaltsverzeichnis

Abkürzungsverzeichnis

ARF	Ausgangsrückführung
CAD	engl.: computer aided design
CB	Codeblock
DCP	engl.: disciplied convex programming
DIN	Deutsche Industienorm
DSL	Spezifikationssprache (engl.: domain specific language)
FB	Funktionsblock
IEKF	engl: identifying extended Kalman filter
IIR	engl.: infinite impulse response: IIR
IIWA	Leichtbauroboter der Firma KUKA
KMS	Kraft-Momenten-Sensor
LMI	engl.: linear matrix inequalities
LQR	Linear-quadratische Regler
OOP	Objektorientierter Programmierung
PID	engl: proportional-integral-derivative controller
PPC	engl.: prescribed performance controller
PbD	engl.: programming by demonstration
QP	engl.: quadratic problem

RCC . . . . . engl.: remote center of compliance

ROS . . . . . engl.: robot operating system

SDP . . . . . Semidefinites Problem

Skill . . . . . Gekapselten Roboterfähigkeiten

TFF . . . . . engl.: task function formalism

VDI . . . . . Verein Deutscher Ingenieure

VKC . . . . . engl.: virtual kinematic chain

XML . . . . . engl.: extensible markup language

eTaSL . . . . . engl.: expressiongraph-based task specification language

iTaSC . . . . . engl.: instantaneous Task specification and control

pitasc . . . . . Beschriebenes Robotersteuerungsrahmenwerk

Symbolverzeichnis

Lateinische Symbole

Symbol	Einheit	Beschreibung
P	-	Performanzband, Performanzregion
$V(\boldsymbol{x})$	-	Ljapunov Funktionskandidat
$\boldsymbol{A}_K, \boldsymbol{B}_K, \boldsymbol{C}_K$	-	Matrixkoeffizienten der ARF
$\boldsymbol{C}_\mathrm{q}, \boldsymbol{C}_\mathrm{f}$	-	Auswahlmatrizen
$\boldsymbol{D}_K, \boldsymbol{E}_K, \boldsymbol{N}_K$	-	Matrixkoeffizienten der ARF
$\boldsymbol{J}_\mathrm{q}, \boldsymbol{J}_\mathrm{f}, \boldsymbol{J}_\mathrm{u}$	-	Jacobi-Matrizen
$\boldsymbol{C}_H, \boldsymbol{D}_H$	-	Matrixkoeffizienten der ungesättigten ARF
$\boldsymbol{A}, \boldsymbol{F}$	-	Systemmatrix
$\boldsymbol{B}$	-	Eingangsmatrix
$\boldsymbol{C}, \boldsymbol{H}$	-	Ausgangsmatrix
$\boldsymbol{D}$	-	Durchgangsmatrix
$\boldsymbol{I}$	-	Einheitsmatrix, ggf. Dimension tiefgestellt
$\boldsymbol{K}$	-	Kalman Verstärkung
$\boldsymbol{0}$	-	Nullmatrix, ggf. Dimension tiefgestellt
$\boldsymbol{P}$	-	Kovarianzmatrix der Zustandsschätzung
$\boldsymbol{Q}$	-	Kovarianzmatrix des Modells
$\boldsymbol{R}$	-	Kovarianzmatrix der Messung
$\boldsymbol{l}$	-	Geschlossene kinematische Schleife
$\boldsymbol{q}_d$	rad	Achswinkelführungsgröße
$\boldsymbol{q}$	rad	Achswinkeln des Roboters
$\boldsymbol{u}_H$	-	Steuerungsvektor unterhalb Sättigungsgrenzen
$\boldsymbol{u}_\mathrm{max}$	-	Zulässiger Maximalwert von $\boldsymbol{u}$
$\boldsymbol{u}$	-	Steuerungsvektor
$\boldsymbol{x}$	-	Zustandsvektor
$\boldsymbol{y}_d$	-	Führungsgröße
$\boldsymbol{y}$	-	Ausgangsvektor
$\boldsymbol{T}_\mathrm{w}^\mathrm{f}$	-	Transformation von f nach w

Symbol	Einheit	Beschreibung
f_x, f_y, f_z	N	Kraftkomponente
m_x, m_y, m_z	Nm	Drehmomentkomponente
$p(t)$	-	Performanzfunktion der PPC Regelung
x, y, z	m	Kartesische Koordinaten
$(\cdot)^+, (\cdot)^\#$	-	Generalisierte gewichtete Matrixinverse

Griechische Symbole

Symbol	Einheit	Beschreibung
α, β, γ	rad	Roll-Nick-Gier-Lagewinkel
$\boldsymbol{\chi}_\mathrm{u}$	-	Unsicherheitskoordinaten
$\boldsymbol{\chi}_\mathrm{f}$	-	Arbeitsraumkoordinaten
ω	-	Imaginärteil eines komplexen Pols
σ	-	Realteil eines komplexen Pols
ε	-	Fehlertransformation der PPC Regelung

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Ausgangssituation und Motivation

Industrieroboter für die Produktion werden vor allem in der Automobilbranche großflächig eingesetzt und können als Stand der Technik angesehen werden. Hierbei sind die mit Abstand am häufigsten durchgeführten Robotertätigkeiten Materialhandhabung oder automatisches Schweißen. In diesen Anwendungsfeldern wird die hohe Wiederholgenauigkeit der Roboter unter starker Reduktion von äußeren Unsicherheitseinflüssen ausgenutzt, um Produkte gleichbleibender Qualität herzustellen.

Für die Montage werden Industrieroboter deutlich weniger eingesetzt und rangieren in der Statistik nur an dritter Stelle. Nach der Internationalen Föderation der Robotik erreichte der Verkauf von Industrierobotern für die Montage im Jahr 2015 etwa 254 Tausend Einheiten und somit ein Wachstum um ca. 15% gegenüber 2014. Weiteres kontinuierliches Wachstum wird erwartet (IFR 2018).

Gründe hierfür finden sich in den von den Robotern durchgeführten Arbeiten und Abläufen. Während bei Anwendungen wie Handhabung oder Schweißen das Roboterwerkzeug wiederholt auf vordefinierten Bahnen verfahren wird, benötigen Montageanwendungen ein kompliziertes Zusammenspiel zwischen dem durch den Roboter geführten Werkstück und der Umgebung. Die besonderen Anforderungen in der Montage wie Bauteiltoleranzen, Genauigkeitsanforderungen, definierte Fügekräfte und großer Variantenreichtum bewirken, dass in heutigen Montagelinien noch ein sehr großes Potenzial für Rationalisierung durch Automatisierung liegt (Pott et al. 2019).

Zum Aufbringen von definierten Fügekräften werden im industriellen Einsatz hauptsächlich passive Mechanismen wie RCC-Module (Whitney 1982) eingesetzt oder direkt die Nachgiebigkeiten der Bauteile bzw. der Montagevorrichtungen ausgenutzt. Hierdurch wird die Wandlungsfähigkeit der Roboterzellen deutlich eingeschränkt. Trotz der Verfügbarkeit von Technologiepaketen zur aktiven Kraftregelung werden diese nur selten im produktiven Einsatz genutzt, und spielen deshalb nur eine untergeordnete Rolle (Winkler 2015).

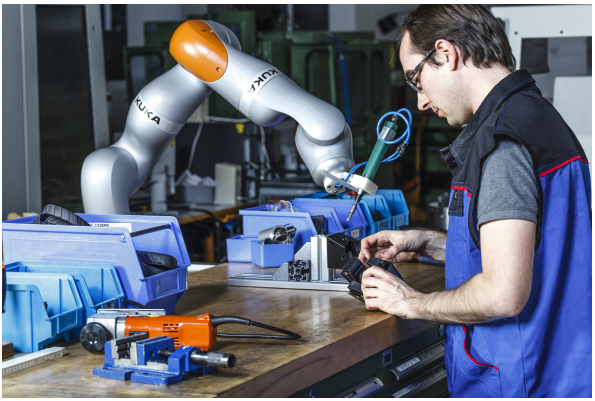


Abbildung 1.1: Der Leichtbauroboter *IIWA* der Firma KUKA im kollaborativen Betrieb zur Unterstützung von Montagetätigkeiten

Industrielle Produkte mit integrierter Kraftregelung wie der Leichtbauroboter *IIWA* der Firma KUKA (Bischoff et al. 2010) oder der Leichtbauroboter *Panda* der Firma Franka Emika (Franka Emika 2019a) ermöglichen es, den Roboter mit einer einstellbaren Nachgiebigkeit zu konfigurieren und so neue Prozesse zu automatisieren. Allerdings eignet sich die vergleichsweise komplizierte Programmierung des IIWAs kaum für einen Einstieg in die Montageautomatisierung bzw. führt zu einer Überschätzung der Komplexität der Programmierung.

Einfache und intuitive Programmierung ist ein wichtiger Schritt für die Verbreitung von Industrierobotern, vor allem in Bezug auf mittlere und kleine Produktionsstückzahlen. Aber auch für die Massenproduktion ist eine verbesserte Wandlungsfähigkeit durch effektives Umprogrammieren ein großer Anreiz. Dies spiegelt sich beispielsweise in dem Erfolg der jungen Firma *Artiminds* wider, welche sich auf einfache Programmierung mit anschließender automatischer Robotercode-Erstellung spezialisiert hat (Artiminds 2019).

Auch Roboterhersteller fokussieren sich auf einfache Bedienbarkeit, wie bspw. (Universal-Robots 2019) und (Franka Emika 2019a). Ebenso adressiert die europäische Forschungsförderung speziell die vereinfachte Bedienung und das Zusammenführen von Einzeltechnologien zu Gesamtsystemen (European Commission 2013). Die einzelnen wissenschaftlichen Fragestellungen für Montagerobotik, wie bspw. Kraftregelung oder Bewegungskontrolle von Robotern, scheinen in der Literatur erschöpfend behandelt und gelöst zu sein. Die ausbleibende Verbreitung des industriellen Einsatzes deutet jedoch auf offene Fragen und praktische Probleme hin.

In Abbildung 1.2 sind Hemmnisse für den Robotereinsatz bei kleinen und mittelständigen Unternehmen aus einer Umfrage des Fraunhofer ISI aufgetragen. Als wichtigste Gründe wurden

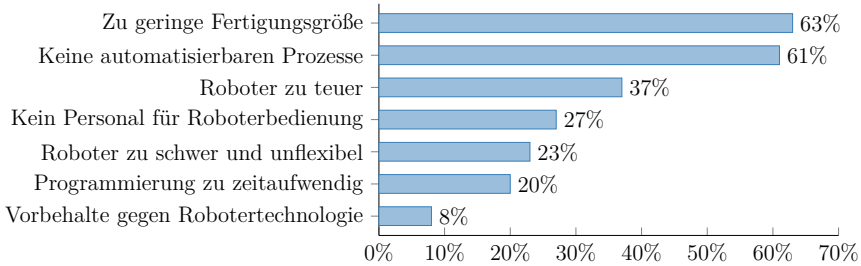


Abbildung 1.2: Hemmnisse für den Robotereinsatz bei kleinen und mittleren Unternehmen nach (Pott et al. 2019)

zu kleine Fertigungsgrößen (63%) und für Automatisierung ungeeignete Prozesse (61%) angegeben. Weitere Gründe liegen in fehlenden Fachkräften (27%) und zu hohem Zeitaufwand bei der Roboterprogrammierung (20%). Das Potenzial der Automatisierung und das Risiko kritische Prozesse automatisieren zu können wird häufig von den Entscheidungsträgern zu pessimistisch beurteilt (Pott et al. 2019).

Durch diese Arbeit soll dazu beigetragen werden einen großen Anteil dieser Hemmnisse abzubauen. Dies wird durch eine vereinfachte Erstellung von Roboterprogrammen und eine Unterstützung zur idealen Parametrisierung der Programme erreicht. Dadurch wird die Automatisierung bei kleinen Fertigungsgrößen wirtschaftlich, die Automatisierung komplexer, bisher nicht-automatisierbarer Prozesse möglich und die Programmierung insgesamt weniger zeitaufwändig.

Zur Verbreitung von kraftgeregelten Montagerobotern im industriellen Einsatz muss die Einstiegshürde und das unternehmerische Risiko möglichst niedrig gehalten werden (Winkler 2015). Der Einstieg kann vereinfacht werden, wenn vorhandene Roboter weiterhin genutzt sowie bereits bekannte Produktionsmittel eingesetzt werden können. Darüber hinaus steigt die Bereitschaft Roboter zu nutzen, wenn die Programmierung als die Beschreibung der einzelnen Prozessschritte erfolgen kann und keine einschlägigen Kenntnisse über die interne Funktionsweise des Roboters vorausgesetzt werden müssen. Das sich aus veränderten Parametern ergebende Verhalten sollte jeweils verständlich und intuitiv nachvollziehbar sein. Ein solches Robotersystem kann nur durch einen Gesamtentwurf erreicht werden, in dem die weitestgehend bekannten wissenschaftlichen Einzeldisziplinen ganzheitlich betrachtet, miteinander verbunden und integriert werden (Kröger et al. 2010).

Daher wurde im Rahmen dieser Arbeit ein nachrüstbares Zusatzsystem für die übergeordnete Bewegungssteuerung und -koordination (*High Level Programmierung*) von komplexen Montagevorgängen entwickelt. Die hierfür notwendige Regelung baut auf verbreiteten tieferliegenden

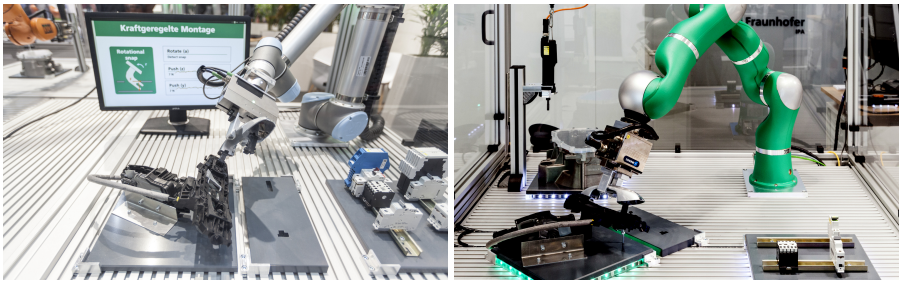


Abbildung 1.3: Die Leichtbauroboter Universal Robots UR5 und KUKA LBR4+ bei gleichen Montageaufgaben mit jeweils denselben Roboterprogrammen.

Schichten der Industrierobotersteuerung auf. Die hierbei befolgten Designprinzipien sind Angemessenheit der Programmierung (ISO/IEC 25010 2011), Modularisierung der Programmierung, Abstraktion und Modularisierung der Roboter und Endeffektoren sowie High Level Programmierung des Prozessablaufs. Angemessenheit der Programmierung bedeutet hierbei, dass der Unterschied an Komplexität der Beschreibung zwischen einfachen und komplexen Prozesse im gleichen Maß dem Unterschied zwischen den jeweils entsprechenden Programmen entspricht. Um die Modularisierung der Programmierung sowie eine Programmierung auf Prozessebene zur ermöglichen muss roboterspezifische Komplexität automatisch aufgelöst und weitestgehend vor dem Bediener verborgen werden, ohne diese für Experten unerreichbar zu machen. Die manuelle Parametrisierung bzw. die Synthese von Kraft- und Positionsreglern ist die zu vermeidende Komplexität, welche in dieser Arbeit vorrangig bearbeitet wird.

1.2 Stand der Technik

Das Forschungsfeld der automatischen Montage mit Robotern setzt sich aus der Vereinigung einer Vielzahl von Einzeldisziplinen zusammen. Es ist eine grobe Einteilung in die Felder der *Montage-Ablaufplanung* und der *Bewegungsplanung und -regelung* ist möglich (Bruyninckx et al. 2001). Das letztere Feld unterteilt sich für die *(nachgiebige) Bewegungsplanung bei Kontakt* weiter in die *Modellierung einer Kontaktsituation*, die *Planung der Bewegung*, die *Wahrnehmung und Schätzung eines Kontaktzustandes* und die *Kontaktkraftregelung*. Die Beherrschung der Vielfältigkeit selbst spiegelt sich in einer eigenen Disziplin der *Systemintegration* wieder (Lefebvre et al. 2012).

Bereits im Jahr 1989 wurden systematische, geometrische Beschreibungen von typischen Einfügeoperationen anhand des *peg-in-hole*-Problems beschrieben (Caine et al. 1989). Im folgenden

Jahr wurden durch den Verein Deutscher Ingenieure (VDI) Richtlinien zur einheitlichen und verständlichen Darstellung von Montageabläufen veröffentlicht (VDI 2860 1990). Kurz darauf folgten die ersten Beschreibungen *skillbasierter* Roboterprogrammierung (Hasegawa et al. 1992). Hiermit konnten typische Montageabläufe als Sequenz von Einzelanweisungen beschrieben werden.

Seit diesen grundlegenden Arbeiten kann das Feld der Montage-Ablaufplanung grob weiter unterteilt werden. Entweder wurden Verfahren entwickelt, um aus Bauplänen des zu montierenden Produkts, bspw. aus CAD-Daten, die Anweisungenssequenz automatisch herzuleiten oder es wurde der Bediener mit Hilfe von graphischen Repräsentationen und vereinfachter Sequenzerstellung bei der Erstellung unterstützt.

Einen Überblick über automatische Sequenzerstellung geben (Lefebvre et al. 2012) und (Jiménez 2013). In diesem Zusammenhang sei insbesondere auf die Arbeiten von Thomas et al. in bspw. (Thomas et al. 2011) hingewiesen. In dieser Arbeit wird auf Basis eines Kontaktformations-Graph einer CAD Konstruktion automatisch ein Netz aus *Manipulationsprimitiven* erstellt. Das so erstellte Netz dient als Schnittstelle zwischen Applikationsplanung und Bewegungskoordination und -ausführung. Die automatische Erstellung optimaler Roboterprogrammabläufe auf Basis simulierter Dekonstruktion von CAD Modellen wurde (Costa et al. 2018) mit Hilfe eines *Branch-and-Bound* Ansatz deutlich beschleunigt.

Neben einer automatischen Erstellung ist es möglich, Anweisungssequenzen durch die Eingabe eines Bedieners erstellen zu lassen. Dies kann bspw. durch die Nutzung einer hierfür ausgelegten Spezifikationssprache (engl. domain specific language: DSL) (Klotzbücher et al. 2011), (Thomas et al. 2013), (Vanthienen et al. 2013) oder mittels grafischer Symbole (Bischoff et al. 2002), (Müller et al. 2017) geschehen.

Die einzelne Roboterfähigkeiten, wie bspw. *Bewegen*, *Drehen* oder *Kontakt-herstellen* werden als *Skills* bezeichnet. Die Effektivität skillbasierter Programmierung wurde in den Arbeiten (Pedersen et al. 2016) und (Stenmark et al. 2017) mit Probandenstudien untersucht. Beide Studien kommen zu dem Schluss, dass eine Vielzahl an industriell relevanten Abläufen durch eine kleine Anzahl an Skills abgedeckt werden können. Die notwendige Programmierzeit fällt hierbei gravierend kürzer als bei konventioneller Programmierung aus. Die Studien beziehen sich auf Bediener ohne Erfahrung mit Roboterprogrammierung.

Weitere Möglichkeiten zur intuitiven Programmierung mittels, Sprach- und Gesteneingabe oder Demonstration durch den Bediener (engl.: programming by demonstration: PbD) werden bspw. in (Maeda et al. 2008), (Akan et al. 2011), (Makris et al. 2014), (Peternel et al. 2015) oder (Jäkel et al. 2010) beschrieben.

Abbildung 1.4: Programmierung einer Handhabungsaufgabe durch Demonstration des Bedieners. Durch eine Tiefenbildkamera werden die Positionen des handgeführten Greifers aufgezeichnet und in entsprechende Roboterarmbewegungen umgerechnet.

Basierend auf den auf (Samson et al. 1991) zurückgehenden *task function formalism* (TFF) und dem *compliance frame* (Mason 1981), spezifizierten Bruyninckx et al. eine bedingungsbasierte Beschreibung von kraftgeregelten Aktionen (Bruyninckx et al. 1996). In (Schutter et al. 2005) wurden diese Beschreibungen für sensorbasierte Roboterbewegungen weiterentwickelt.

Mit dem Formalismus iTaSC (Smits et al. 2008), (Smits 2010) wurde ein systematisches Vorgehen definiert um die Zwangsbedingungen im Aufgabenraum in Stellgrößen für die Antriebe des Roboters aufzulösen. Durch den ausschließlichen Einsatz von Bedingungsgleichungen konnte das Optimierungsproblem analytisch gelöst werden. Weiterhin kann eine Priorisierung von Bedingungen durch rekursive Nullraumprojektion erreicht werden (Vgl. Kapitel 2.3.1). Anwendungsbeispiele werden bspw. in (Smits et al. 2011), (Laet et al. 2012) oder (Borghesan et al. 2012) angeführt. Auf diesem Ansatz basiert diese Arbeit.

Die Weiterentwicklung der DSL (Vanthienen et al. 2013) und die Erweiterung um Bedingungsungleichungen (Decré et al. 2013) führten zu einer Neuauflage des Formalismus mit dem Namen eTaSL (Aertbeliën et al. 2014). Das Optimierungsproblem zur Auflösung der Roboterbewegung resultiert in ein *quadratisches Programm* (QP), welches in eTaSL durch einen entsprechenden Solver numerisch gelöst werden muss.

In (Mansard et al. 2009) wurde ein zu iTaSC vergleichbares Vorgehen namens *Stack-of-Tasks* (SoT) für die Anwendung bei humanoiden Robotern beschrieben. Weitere aktuelle Veröffentlichungen beschäftigen sich mit der Beschreibung und Lösung von Roboterbewegungen für spezi-

elle Aufgaben. In (Hoffman et al. 2018) wird ein priorisierter kartesischer Nachgiebigkeitsregler mit der Echtzeitlösung eines quadratischen Programms realisiert. Ebenfalls mittels Lösung eines QP Problems wird in (Maderna et al. 2018) eine Roboterarmbewegung so eingeregelt, dass das Überlaufen einer transportierten Flüssigkeit verhindert wird.

Die Ausführung zuvor definierter Abläufe basiert je nach unterliegendem System auf endlichen Zustandsautomaten (Klotzbücher et al. 2012) bzw. Zustandsnetzen (Thomas et al. 2013), *Behavior Trees* (Csiszar et al. 2017) oder Petri Netzen (Wang et al. 2015). In den Arbeiten (Lambrecht et al. 2011), (Wang et al. 2015), (Pedersen et al. 2016), (Müller et al. 2017), (Herrero et al. 2017), (Stenmark et al. 2017), werden die zugrundeliegenden Funktionen (*Primitive*) durch Experten im Vorfeld bereitgestellt. Primitive können typischerweise durch den Benutzer in Teilabläufe zusammengefasst werden (zusammengesetzte Skills) oder anhand eines CAD Modells gefolgert (Mosemann et al. 2001). Gesamtanwendungen können wiederum aus Skills und Primitiven zusammengesetzt werden.

Ein Sonderfall bildet das System von Finkemeyer. Mit Hilfe einer zeitvarianten Auswahlmatrix werden die Stellgrößenanteile verschiedener Roboterverhalten (bspw. durch Kraft- und Positionsregelung) reguliert. Somit können komplexe Roboterverhalten kombiniert werden (Finkemeyer 2004). Das System wurde auch für die Ausführung von Montageabläufe eingesetzt (Finkemeyer et al. 2005). Durch eine kontinuierliche Veränderung der Auswahlmatrix werden weiterhin glatte Transitionen zwischen den Einzelschritten eines Ablaufs ermöglicht (Kröger et al. 2011). Damit wird dem Bediener erlaubt, durch Experten bereitgestelltes Verhalten in geeigneter Form für die gegebene Aufgabe zu kombinieren.

Diverse industrielle Anwendungen wie die Montage eines Not-Stop Knopfs (Stolt et al. 2011a), von Flugzeugbaugruppen (Stolt et al. 2011b) oder eines Mobiltelefons (Stolt et al. 2012), wurden mit einem System basierend auf der Arbeit von (Stolt 2012) realisiert. Die Montageabläufe wurden durch eine übergeordnete Zustandsmaschine und einer iTaSC basierten Ausführungsschicht durchgeführt.

Im Bereich der Servicerobotik arbeitet Beetz et al. an einem cloudbasierten Robotersteuerungssystem. Einen Überblick gibt (Beetz et al. 2018). Die im Zentrum stehende symbolische Datenbank verknüpft frei verfügbares Wissen aus dem Internet, Eingaben durch Benutzer sowie Umgebungswissen der eingesetzten Robotersysteme. Innerhalb dieser angereicherten Wissensdatenbank existieren auch Ablaufbeschreibungen, welche zu Roboterbewegungen umgesetzt werden können (Kresse et al. 2012). Die interne Repräsentation von Prozessschritten basiert auf zu erfüllenden Randbedingungen (Tenorth et al. 2014), welche durch eine eTaSL Implementierung aufgelöst wird.

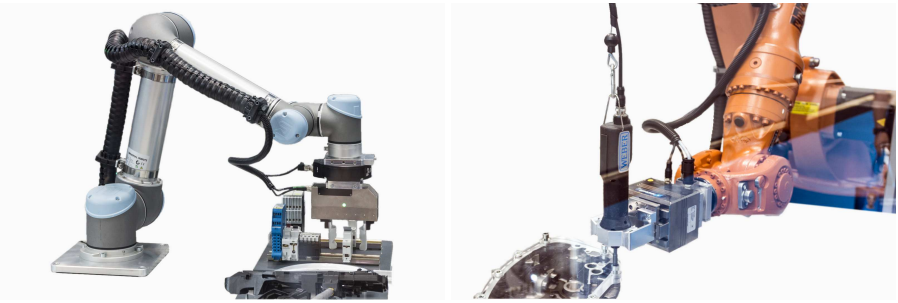


Abbildung 1.5: Links: Ein UR5 bei einer Anwendung aus der Schaltschrankmontage. Das elektronische Einbauelement wird auf eine Hutschiene eingeklipst. Rechts: Mithilfe eines elektrischen Handschraubendrehers verschraubt ein KUKA KR16 Industrieroboter einen Motorblock. Nachdem der Kontakt zwischen dem Schrauber und der Schraube hergestellt ist, muss der das Schraubwerkzeug dem Schraubenkopf während des Einschraubvorgangs folgen.

Ein intuitives Roboterprogrammiersystem auf Basis von geometrischen Beziehungen zwischen den zu montierenden Baugruppen und Aufgabenbeschreibungen durch den Benutzer wurde in (Perzylo et al. 2015), (Somani et al. 2015), (Perzylo et al. 2016) und (Somani et al. 2016) beschrieben. Die geometrischen Beziehungen wurden aus einem CAD Modell des Werkstücks entnommen und mittels einer bereitgestellten Taxonomie semantischen Prozessmodellen zugeordnet. Die Robotersysteme boten vollständig vorparametrisierte *engl.: low level implementation* an, welche die modellierten Prozesse realisierten.

Durch die vollständige Modellierung der Roboterumgebung, der Bauteile und der entsprechenden Nachgiebigkeiten wurden Roboterbewegungen zur Montage innerhalb einer physikalischen Simulation geplant (Schmitt et al. 2017), (Wirnshofer et al. 2018). Unter Berücksichtigung der aktiven Nachgiebigkeit des Roboters wurden die Roboterpfade erstellt, welche innerhalb der simulierten Unsicherheiten am erfolgreichsten sind. In (Malhan et al. 2018) wurden in einem vergleichbaren simulationsbasierten Ansatz optimale Parameter für die Nachgiebigkeitsregelung eines Roboters gesucht.

Die Regelungsansätze für Kraftregelung haben über eine lange Zeit eine ausreichende Reife entwickelt (Lefebvre et al. 2012). Eine erste Unterteilung erfolgt zwischen Methoden, die *direkt* die geforderte Kraft einstellen, und Methoden, welche die Kraft *indirekt* durch ein nachgiebiges Verhalten des Roboters erwirken[1]. Direkte Methoden untergliedern sich weiter in *hybride Kraft-/Positionsregelung* und *Parallele Kraftreglung*. Indirekte Methoden können in *passive* und *aktive* Ansätze unterteilt werden.

[1] Direkte und indirekte Methoden werden teilweise als *explizite* bzw. *implizite* Methoden bezeichnet.

Bei hybrider Kraft-/Positionsregelung wird jeder der sechs Bewegungsrichtungen entweder einer Bewegungs- oder einer Kraftregelung zugeordnet (Raibert et al. 1981). Die entsprechende Spezifikation wurde mit engem Bezug zu TFF und dem *compliance frame* entwickelt (Lefebvre et al. 2012). Kraftgeregelte Bewegungsrichtungen werden mittels einer "inversen Nachgiebigkeit" (Admittanz) in Bewegungsbefehle umgewandelt. Kraftabweichungen werden in Bewegungen umgesetzt, die den Fehler reduzieren.

Parallele Kraftregelung hingegen erlauben Kraft- und Positionsregelung in gleicher Richtung. Die Kraftregelung wird wie bei dem hybriden Ansatz umgeformt. Konkurrierende Stellgrößen der Kraft- und Positionsregelungen werden gewichtet zusammengefasst (Chiaverini et al. 1997), (Siciliano et al. 1999).

Die einfachste und verbreitetste Form einer indirekten Kraftregelung ist der Einsatz eines passiven Nachgiebigkeitselements (RCC) (Whitney 1982). Ein elastisch verformbares Element wird so an den Roboter angebracht, dass die Kraft durch dessen Verformung eingestellt wird.

Aktive Nachgiebigkeitsregelung reguliert nicht die Kräfte selbst, sondern regelt eine (virtuelle) Nachgiebigkeit des an sich steifen Roboters (Hogan 1984). Das nachgiebige Verhalten des Roboters wird mit einer Wegplanung überlagert. Somit ergibt sich die wirkende Kraft aus der Nachgiebigkeit des Roboters und wie tief der zu erreichende Wegpunkt in die Umgebung "eindringt".

Für die Einregelung von Kontaktkräften bei Kontaktaufbau in einer unbekannten Umgebung wurde in (Bechlioulis et al. 2008) ein aktiver Kraftregelungsansatz beschrieben, welcher anhand des aktuellen Regelfehlers die Regelungsverstärkungsfaktoren adaptiert. Die Verstärkungsfaktoren werden so gewählt, dass der Istwert-Verlauf innerhalb definierbarer Performanzgrenzen bleibt. Es wird von *Prescribed Performance Control* (PPC) gesprochen. Der Ansatz wurde in (Kanakis et al. 2017) auf die Regelung von Robotern angewendet und in (Kanakis et al. 2018) experimentell für vergleichbar hohe Kontaktgeschwindigkeiten untersucht. Dieses Verfahren wird in dieser Arbeit aufgegriffen, da es unabhängig des zu regelnden Systems eine definierte Performanz anstrebt und somit übertragbar bei gleichbleibender Performanz ist.

Winkler et al. haben in diversen Veröffentlichungen einfache Strukturen von Kraftreglern untersucht und verglichen (Winkler et al. 2012), (Winkler et al. 2015), (Winkler et al. 2016). Ein Verfahren um die Leistungsfähigkeit von Kraftregelungsansätzen für Konturfolge-Aufgaben vergleichen zu können, wurde in (Behrens et al. 2018) präsentiert.

Um dateneffektives Parametrisieren von gegebenen Reglerstrukturen zu ermöglichen, wurde maschinelles Lernen vorgeschlagen, um optimale Parameter eines linear-quadratischen Reglers (LQR) (Marco et al. 2016) und einer multivarianten PID-Regler Struktur (Doerr et al. 2017b)

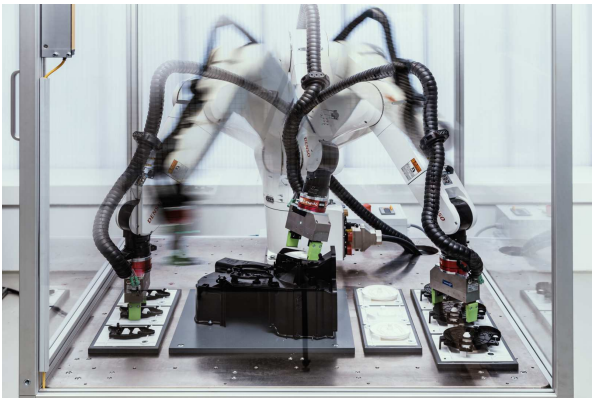


Abbildung 1.6: Fotomontage der verschiedenen Teilabläufe einer kraftgeregelten Montage von Kunststoffteilen zur robusten Produktion trotz Bauteiltoleranzen

für Gauß Prozesse zu explorieren. Bei diesen Ansätzen wurden iterativ Experimente zur Modellbildung möglichst effektiv durch das Identifikationssystem ausgewählt und durchgeführt. Eine Methode zur Verbesserung des Modelllernens wird in (Doerr et al. 2017a) vorgestellt.

Die Grundlagen für den in dieser Arbeit verwendeten Entwurf einer sättigenden dynamischen Ausgangsrückführung (ARF) beruhen auf (Lens 2010). Der Entwurf einer Regler-Beobachter-Struktur in (Lens et al. 2008) und (Lens 2009) kann in diesem Zusammenhang als Vorarbeit angesehen werden. Da die Einhaltung der Stellgrößenbegrenzung wie bspw. zulässige kartesische Verfahrgeschwindigkeiten wichtige Verfahrensparameter der Montage sind, eignet sich die ARF als Beispiel eines automatisch synthetisierten modellbasierten Regelungsentwurfs. Bereits zur Synthese wird die Stellgrößenbegrenzung berücksichtigt und eine optimale Lösung unter der Voraussetzung der Begrenzung gesucht.

In (Abid et al. 2013) wurde die Identifikation linearer dynamischer Modelle mittels Variationen von Kalman Filtern untersucht. Eine Systematik für ein *identifizierendes erweitertes Kalman Filter* (IEKF) wurde in (Best et al. 2007) präsentiert und in (Best et al. 2017) um die Möglichkeit erweitert, nicht-lineare Anteile mittels *Spline Interpolation* anzunähern und zu identifizieren. Das Entwurfsverfahren der ARF nutzt ein lineares Systemmodell, was die Identifikation mittels Kalman Filter nahe legt.

Die automatische Synthese basiert auf der Anwendung von konvexer Optimierung (Boyd et al. 2015). Kann ein Optimierungsproblem in konvexer Form ausgedrückt werden, stehen eine Vielzahl an effektiven Solvern zur Verfügung. Einen Überblick gibt bspw. (Mittelmann 2012). Der

entscheidende Vorteil eines konvexen Optimierungsproblems liegt darin, dass ein globales Optimum existiert. Die Lösung des Problems ist somit unabhängig von eingesetztem Solver und Initialwert. Die Klasse der *linearen Matrixungleichungen* (engl.: linear matrix inequalities: LMI) bildet eine Untergruppe der allgemeinen konvexen Optimierungsprobleme. LMIs finden sich häufig in regelungstechnischen Problemstellungen (Boyd et al. 1994). Für diese Arbeit bedeutet der Einsatz von LMIs daher den Rückgriff auf bewährte und bekannte Verfahren sowie Software-Werkzeugen.

Um einheitliche Schnittstellen bei der Vielzahl an verfügbaren Solvern bereitzustellen, wurden in MATLAB die Schnittstellen (engl. Interfaces) *YALMIP* (Löfberg 2004) oder *CVX* (Grant et al. 2014) entwickelt. Für die Sprache Python sind bspw. die Pakete *picos* (Sagnol 2012) und *cvxpy* (Diamond et al. 2016) verfügbar. Cvxpy garantiert hierbei, angelehnt an CVX, die Konvexität der gegebenen Probleme durch die Prüfung auf *disziplinierte konvexe Programmierung* (engl.: disciplied convex programming: DCP) (Grant et al. 2006).

Mit diesem Überblick über Programmierkonzepte, Regelungsstruktur- und Regelungsentwurfsansätzen zeigt sich, dass die Einzelfragestellungen weitestgehend beantwortet wurden. Es scheint allerdings noch Hemmnisse zu geben, die den Zugang zu den Technologien für die breite Anwendung verhindern. Im Folgenden soll die Integration einer Auswahl an repräsentativen Technologien in ein Gesamtsystem dargestellt und hiermit die Forschungsaufgabe spezifiziert werden.

1.3 Defizit in Bezug auf übertragbare Roboterprogramme und Forschungsaufgabe

Der Überblick über den Stand der Technik zeigt, dass umfassende Ansätze und Grundlagen für die Einzelfragestellungen vorliegen. Bereits 2001 betonten Bruyninckx et al. die Wichtigkeit durchgängiger Gesamtlösungen für "intelligente Robotersteuerungen" (Bruyninckx et al. 2001). Vergleichbare Anmerkungen und Aufrufe an die Entwicklergemeinde sind auch in (Kröger et al. 2010) und (Lefebvre et al. 2012) zu finden.

In der Literatur finden sich verschiedene Ansätze für durchgängige Gesamtlösungen. Der Ansatz dieser Arbeit namens *pitasc* wird mit drei repräsentativen Ansätzen (*Funktionsblock-basiert*, *Code Generator* und *Auswahlmatrix*) verglichen um die jeweiligen Beziehungen oder Abgrenzungen darzustellen.

Die auf Primitiven bzw. Funktionsblöcken (kurz: FB) basierenden Systeme fokussieren sich typischerweise auf eine intuitive Bedienung und Programmierung von einfachen Roboteraufgaben. Eine Zustandsmaschine ruft hierbei vorgefertigte Funktionen des Robotersystems auf.

Diese Systeme eignen sich, um effektiv repetitive manuelle Arbeiten zu automatisieren und so die Einstiegshürde zur Automatisierung zu verringern, wie auch die empirischen Arbeiten von (Pedersen et al. 2016) und (Stenmark et al. 2017) bestätigen.

Werden die Funktionsblöcke als aktive Komponenten erstellt, besteht eine laufzeitkritische Verbindung zwischen der koordinierenden Zustandsmaschine und den FBs. Werden die auszuführenden Funktionen als Codeblöcke (kurz: CB) in einer Grammatik für die jeweiligen spezifischen Roboterprogrammiersprache bereitgestellt und zugeordnet, ist es möglich den kompletten Ablauf in roboterspezifischen Code zu übersetzen. In diesem Fall ist zur Laufzeit ausschließlich die Robotersteuerung aktiv. Die durch den Bediener spezifizierte Zustandsmaschine wird hierbei vollständig in Robotercode überführt und übertragen. Komplexe Fügebewegungen oder grundlegende "neue Skills" können durch einen Bediener nur schwer basierend auf FBs oder CBs erstellt werden. Hierfür sind weiterhin Roboterexperten notwendig, die die Funktionen oder Implementierungen zur Verfügung stellen. Zudem sind Skills mit Spezialverhalten, bspw. unter Einsatz von nachgiebigem Verhalten des Roboters, nicht automatisch auf andere Roboter übertragbar. Daher eignet sich dieses Verfahren nicht für die komplexe und variantenreiche Programmierung von Montagerobotern.

Das System von Finkemeyer erlaubt dem Bediener neue Verhalten durch geeignete Kombination von voreingestellten Verhalten mit einer Auswahlmatrix zu erstellen. Reglerschleifen zur Realisierung einzelner Verhalten wie bspw. Geschwindigkeits-, Positions- oder Kraftregelung wurden einzeln für das Robotersystem vorgefertigt und mit einer Auswahlmatrix zusammengeführt. Die Zustandsmaschine erlaubt eine Manipulation der Auswahlmatrix und somit eine kontinuierliche und zustandsbezogene Kombination der Verhaltensweisen. Da die praktischen Auswirkungen der Verhaltenskombinationen von der spezifischen Realisierung der Einzelverhalten abhängig sind, bleibt es unklar ob eine Applikationsbeschreibung aus Auswahlmatrixkonfigurationen roboterunabhängig ist. Eine direkte Übertragung von Roboterprogrammen bzw. roboterunabhängige Programmierung wird nicht fokussiert. Außerdem liegt im Fall eines nachrüstbaren Steuerungssystems kein direkter Zugriff auf die internen Regelantworten des Roboters vor. Dies macht eine Durchführung durch die Auswahlmatrix unmöglich, wenn die Regelschleifen nicht jeweils außerhalb der Robotersteuerung nachgebildet werden.

Das hier eingesetzte System *pitasc* erweitert das Vorgehen von Stolt bzw. von Smits. Die auszuführenden Prozesse werden im Aufgabenraum bspw. als Kontaktkraftanforderung zwischen dem Roboter und einem Bauteil beschrieben. Mit dem iTaSC Formalismus werden diese Prozesse in roboterabhängige Achsgeschwindigkeitsvorgaben gewandelt. Der Bediener spezifiziert eine Zustandsmaschine mit semantischen Zuständen, welche im Hintergrund zu entsprechenden iTaSC und Regler-Konfigurationen umgesetzt werden. Um die Nachbildung von bereits durch den Roboter gegebenen Fähigkeiten zu vermeiden, kann alternativ zu der äußeren Regelung

mittels iTaSC auch auf eine FB-basierte Ausführung der vorhandenen Funktionen umgeschaltet werden. Somit kann der Bediener eine Roboteranwendung erstellen, die sich aus einer beliebigen Kombination der robotereigenen und der durch das System erweiterten Fähigkeiten zusammensetzt.

In Tabelle 1.1 sind die prinzipiellen Abläufe der verschiedenen Ansätze vereinfacht dargestellt. Die grobe Grenze, unter der die Komponenten roboterspezifische Details enthalten, ist durch einen horizontalen doppelten Stich angedeutet. Weiter ist der Bereich der laufzeitaktiven Komponenten durch einen vertikalen Balken markiert. Graue Pfeile deuten eine Parameterübertragung an, gepunktete Pfeile symbolisieren softwareinterne Verknüpfungen und gestrichelte Pfeile stehen für serviceorientierte Funktionsaufrufe.

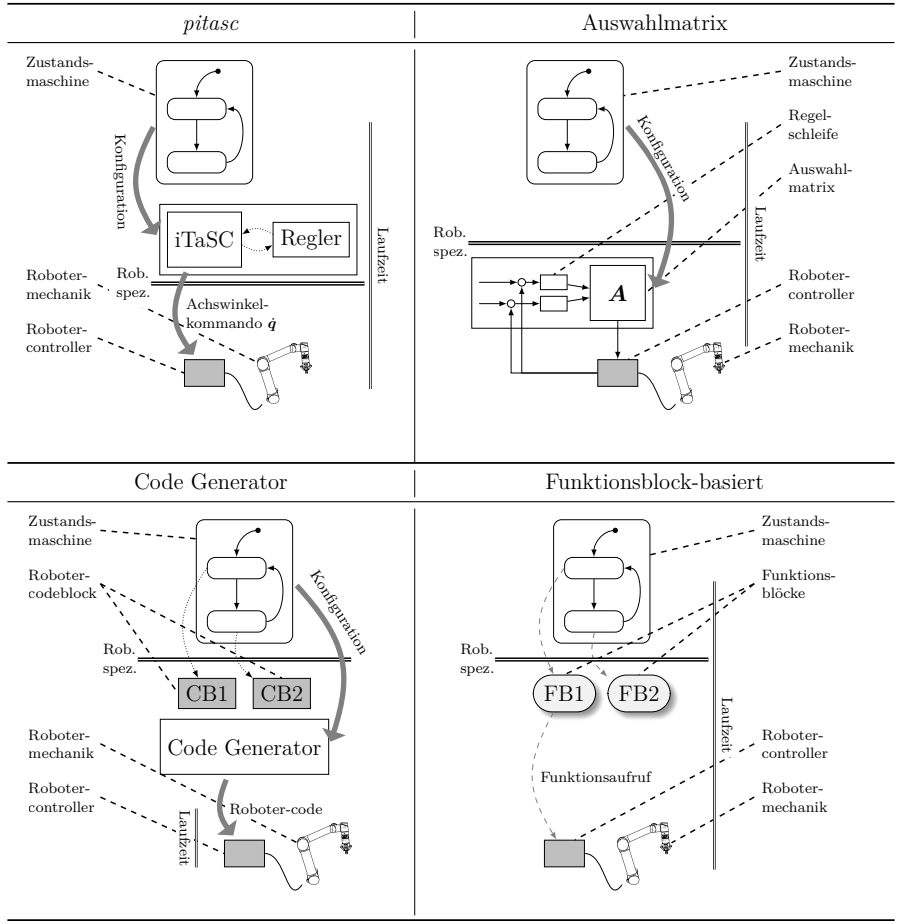
Im Zuge dieser Arbeit wurde das Robotersteuerungsrahmenwerk *pitasc* geschaffen. Die jeweiligen Designentscheidungen wurden im Hinblick auf die Erweiterbarkeit der Roboterprogramme, die Wiederverwendbarkeit von Teilprogrammen und damit deren Roboterunabhängigkeit entschieden. Die Auswahl und Verzahnung der Software-Schichten wird näher in Kapitel 2 beschrieben. Da das so entstandene System allerdings keine Kenntnis der dynamischen Eigenschaften der Einsatzumgebung besitzt, muss die Einstellung der Regler bislang durch den Bediener manuell vorgenommen werden. Hierbei neigen Standardeinstellungen oder Regelungsansätze mit wenigen Einstellmöglichkeiten zu konservativer Reglerperformanz.

Diese Arbeit beschäftigt sich mit der Fragestellung, geeignete Regelungsansätze einzupassen und zu erproben, sodass das System mit Regelungsansätzen mithalten kann, welche volle Kenntnis über das geregelte System voraussetzen. Darüber hinaus sollen die Regelungsansätze die Wiederverwendbarkeit und Übertragbarkeit von Programmen zwischen verschiedenen Robotern und Anwendungen unterstützen.

Hierfür wird auf eine Methode zur online Identifikation eines linearen mathematischen Modells mittels Kalman Filter eingegangen. Mit Hilfe des mathematischen Modells wird ein automatischer Entwurf einer dynamischen Ausgangsrückführung (ARF) vorgestellt. Im Vergleich zu konventionellen Entwurfsverfahren (bspw. LQR Regler) wird in diesem Ansatz die Begrenzung der Stellgröße, d. h. die maximale Robotergeschwindigkeit, miteinbezogen.

Für den Einsatz zur direkten Kontaktkraftregelung ist die experimentelle Ermittlung eines mathematischen Modells zu riskant. Treten in diesen Experimenten betragsmäßig zu große Kräfte auf, so besteht die Gefahr, dass der Roboter oder die Arbeitsmittel beschädigt werden. Für diesen Fall wird ein Kraftregelungsansatz aufgegriffen und eingearbeitet, welcher ohne die Voraussetzung eines im Vorfeld bekannten mathematischen Systems auskommt.

Tabelle 1.1: Darstellung der prinzipiellen Abläufe verschiedener Robotersteuerungsansätze. Horizontalen doppelten Stich deutet eine Einschätzung der Grenze an, unter der die Komponenten roboterspezifische Details enthalten. Weiter ist der Bereich der laufzeitaktiven Komponenten durch einen vertikalen Balken markiert. Graue Pfeile deuten eine Parameterübertragung an, gepunktete Pfeile symbolisieren softwareinterne Verknüpfungen und gestrichelte Pfeile stehen für Funktionsaufrufe.



Die zuvor aufgeführten Einzelansätze werden in das Rahmenwerk eingepasst und sukzessiv miteinander in Verbindung gesetzt. Hierdurch ergeben sich sowohl isolierte wissenschaftliche Beiträge bei den Einzelansätzen als auch die exemplarische Darstellung einer Gesamtintegration.

Diese Arbeit behandelt daher im Einzelnen:

- Identifizierendes erweitertes Kalman Filter
  - Konkrete Implementierung einer symbolisch-numerischen Lösung
  - Untersuchung der Konvergenzgeschwindigkeiten unter dem Einfluss von Abtastung
  - Entwicklung eines Indikators für die Konvergenz
- Dynamische Ausgangsrückführung
  - Untersuchung der Robustheit gegenüber Zeitverzug und Abtastung
  - Berechnung eines Vorfilters und Diskretisierung der synthetisierten Ausgangsrückführung
  - Untersuchung des Einflusses der Begrenzung der platzierten Eigenwerte
  - Experimentelle Erprobung des Ansatzes in Verbindung mit einem zuvor automatisch identifizierten Modells
- Regler mit vorgegebenem Performanzband
  - Entwicklung einer Reglerstruktur im Aufgabenraum und deren Stabilitätsnachweis
  - Entwicklung von Maßnahmen gegen die Verletzung der Stabilitätsvoraussetzungen unter realen Bedingungen
  - Experimentelle Erprobung des Ansatzes in Simulation und mit verschiedenen Robotertypen und Umgebungsnachgiebigkeiten
- Integration in *pitasc*
  - Durchführung einer systematischen Reglersynthese im Aufgabenraum
  - Experimentelle Erprobung von Programmübertragungen zwischen verschiedenen Robotertypen
  - Beschreibung einer automatischen Reglersynthese ohne a priori Kenntnis der zugrundeliegenden Systemdynamik

Die Arbeit ist wie folgt gegliedert: In Kapitel 2 wird zunächst auf das zugrundeliegende Robotersteuerungsrahmenwerk *pitasc* eingegangen. Hierbei werden die einzelnen Softwareschichten, deren Aufgaben und Interaktion beschrieben. Das skillbasierte Programmierkonzept wird motiviert und deren Bezug zu dem Formalismus iTaSC beschrieben. Auf den Formalismus selbst und dessen Realisierung wird im Folgenden eingegangen. Das Kapitel befasst sich weiter mit den Kommunikationsmechanismen im *Robot Operating System* ROS, mit Grundlagen für die Einzeltechnologien des identifizierendem Kalman Filters IEKF und der Synthese der dynamischen Ausgangsrückführung ARF. Hierfür wird zunächst die Funktionsweise und die mathematischen Formulierungen eines Kalman Filters dargestellt, auf die Form und wichtige Eigenschaften von *linearen Matrizenungleichungen* LMIs und auf die Stabilitätsanalyse mittels direkter Methode von Ljapunov eingegangen.

Kapitel 3 beschreibt in seinen drei Unterkapiteln jeweils zunächst eine Einzeltechnologie, deren Implementierung sowie deren Erprobung anhand von simulierten Umgebungen: Modellidentifikation durch ein erweitertes Kalman Filter, modellbasierte Synthese einer dynamischen Ausgangsrückführung sowie modellfreie Regelung mit vorgegebenem Performanzband.
Ein Konvergenzmaß für die Identifikation eines linearen Systemmodells mittels erweiterten Kalman Filters wird aus dem Modellfehler abgeleitet. Um sicherzustellen, dass sich das Modell nicht im stationären Zustand befindet und deshalb der Modellfehler abklingt, wird zusätzlich die Frobeniusnorm der Kovarianzmatrix betrachtet.
Weiter wird das LMI Optimierungsproblem der Synthese einer ARF hergeleitet, ein Vorfilter berechnet und das synthetisierte Zustandsraumsystem diskretisiert. Mit Simulationen wird die Robustheit der ARF auf unmodellierte Verzugszeiten und Reglertaktung, sowie der Einfluss der Begrenzung der resultierenden Systemdynamik untersucht.
Als dritte Einzeltechnologie wird der modellfreie Regelungsansatz mit vorgegebenem Reglerperformanzband PPC angeführt. Die Struktur des Reglers im Aufgabenraum des iTaSC Formalismus wird dargestellt und dessen Stabilität nachgewiesen. Um den PPC robust gegenüber ungünstigen Einstellung des Performanzbands oder überschätzter Roboterdynamik zu gestalten, werden Maßnahmen vorgenommen, die einen fortlaufenden Betrieb des Reglers selbst bei Verletzen des vorgegeben Bandes erlauben und ggf. zu schwach gewählte Verstärkungsfaktoren vergrößern. Der Regelungsansatz wird ebenfalls anhand von Simulationen auf die Robustheit gegenüber verrauschten Signalen und Reglertaktung untersucht.

In Kapitel 4 werden an realen Robotern die im vorherigen Kapitel beschriebenen Szenarien anhand von Versuchen verifiziert. Zuerst wird voneinander isoliert die Positionsregelung mittels ARF und die Modellidentifikation mittels IEKF für verschiedene Roboter erprobt. In einem weiteren Schritt werden beide Technologien verbunden und die Synthese einer ARF auf Basis eines zuvor durch einen IEKF identifizierten Modells gezeigt. Diese werden mit der entspre-

chenden Sprungantwort des robotereigenen internen Reglers verglichen.
Zur Untersuchung der Kontaktkraftregelung mittels PPC wurde die Regelung auf drei Robotern verschiedener Dynamik mit fünf verschiedenen Umgebungsnachgiebigkeiten angewendet und verglichen. Beim Vergleich der Dauer der Stabilisierungszeiten über die verschiedenen Roboter und Umgebungsnachgiebigkeiten zeigt der PPC sich deutlich einheitlicher und somit besser übertragbar als der in der Literatur verbreitete Impedanzregleransatz. Das Experimentalkapitel wird mit der Diskussion geschlossen, wie die Ergebnisse der Experimente auf praktische Anwendungen übertragen werden können.

In Kapitel 5 wird die Arbeit zusammengefasst und ein Ausblick auf mögliche weiterführende Entwicklungen gegeben. Hierbei wird der bislang fehlende Stabilitätsnachweis einer zeitdiskreten Regelung mit vorgegebenem Performanzband sowie die Möglichkeit die Robustheit der ARF durch weitere Randbedingungen zu erhöhen adressiert.

2 Grundlagen

Dieses Kapitel beschreibt den Aufbau des Gesamtsystems, die Realisierung der einzelnen Systemschichten und deren Anbindung an die Hardware. Das aus verschiedenen Komponenten bestehende Gesamtsystem wird mit dem Namen *pitasc* bezeichnet. Mit Hinblick auf die systematische Trennung von funktionellen Schichten in der Steuerungsentwicklung für Roboter (Dominick et al. 2014) wurde eine Konfigurationsschicht zur Aufgabenbeschreibung mit einer iTaSC-basierten Ausführungsschicht angesetzt.

Es wird ein Verständnis für die verschiedenen Aufgaben und die Trennung der *Konfigurationsphase* und *Ausführungsphase* und deren zeitliche Abfolge aufgebaut. Da der Fokus dieser Arbeit auf der Ausführungszeit liegt, wird auf die entsprechende tiefere Unterteilung eingegangen und mit anderen Systemansätzen verglichen. Abschließend werden mathematische und systemtheoretische Grundlagen dargestellt, welche für die angewendeten Ansätze vorausgesetzt werden.

2.1 Ein System Überblick

Der typische Arbeitsablauf einer Roboterprogrammierung bei der Umsetzung einer Produktionsaufgabe gliedert sich in zwei Phasen. Zuerst wird ein Ablauf strukturiert eingegeben. Ist die Struktur eines (Teil-)Ablaufes festgelegt, wird mit der Konfiguration von Parametern fortgefahren. Dieser Zeitabschnitt wird hier als *Konfigurationsphase* bezeichnet.

In der Konfigurationsphase bewegt sich der Roboter typischerweise nicht selbsttätig. Je nach Programmierkonzept wird ggf. der Roboter in einen aktiven Modus versetzt und unter besonderer Sorgfalt des Bedieners bewegt, um die Konfiguration eines zuvor ausgewählten (Teil-)Ablaufs zu gestalten. Wurde der Roboter so in eine gewünschte Zielpose bewegt, können die entsprechenden Positionsdaten (Achsstellungen oder kartesische Zielpose) in das Programm als Zieleinstellung einer Bewegung übernommen werden. Die aktuelle Stellung des Roboters gibt dem Bediener einen direkten Eindruck über die gewählten Einstellungen. Danach fährt der Bediener mit der Wahl eines nächsten (Teil-)Ablaufs fort. Diese Art der Programmierung wird

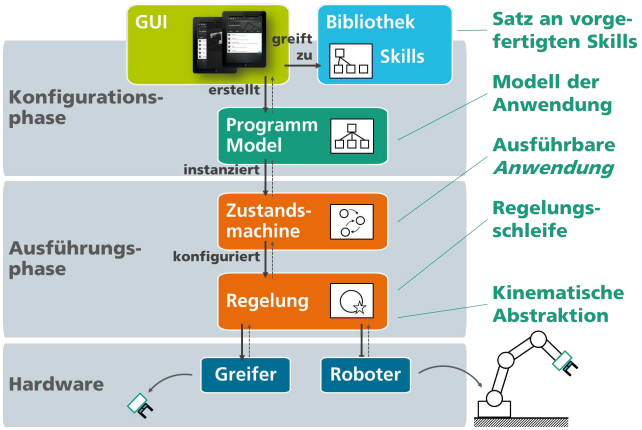


Abbildung 2.1: Systemüberblick der *pitasc* Softwareschichten und der Einsatzphasen (Halt et al. 2018b).

als Online-Programmierung bezeichnet. Der Vorgang den Roboter in die gewünschte Pose zu verfahren und dort die Koordinaten abzuspeichern trägt die Bezeichnung *Teach-In*.

Mit der Ausführung eines konfigurierten Ablaufs beginnt der eigentliche produktive Betrieb. Während der *Ausführungsphase* kann der Bediener zwar den Ablauf stoppen, nimmt aber typischerweise keine aktive Rolle mehr in der Ablaufsteuerung ein. Zu Beginn der Ausführungszeit wird die vollständige Parametrisierung des Ablaufes überprüft und die Ausführbarkeit wird sichergestellt. Wurde der Ablauf beendet oder durch den Bediener unterbrochen, kann in die Konfigurationsphase zurückgekehrt werden. Der Vorgang einer typischen Roboterprogrammierung enthält meist viele Sprünge zwischen den Phasen.

Hierzu im Gegensatz sind die Phasen bei der Offline-Programmierung am stärksten getrennt, da meist ein komplettes Programm ohne aktive Ausführung der wirklichen Roboteranlage erstellt wird. Eine Vermischung findet hingegen bei der Online-Programmierung eines Roboterarms statt, da jede Eingabe direkt ausgeführt wird. Der Bediener wählt die Programmstruktur im Vorfeld und gibt diese über das Eingabegerät ein.

In dem hier verfolgten Systemansatz wird auf die starke Trennung der Phasen geachtet. Zum einen kann so die Nachvollziehbarkeit eines ausgeführten Programms erhöht werden, da ohne aktive Rekonfigurationsphase dasselbe Programm immer denselben Ablauf verfolgt, zum anderen kann ein Programm während der Konfigurationsphase als weitestgehend roboterneutral angesehen werden und erst mit Beginn der Ausführungsphase konkretisiert werden. Die Phasentrennung bietet also eine geeignete Schnittstelle für übertragbare Roboterprogramme.

Die Ausführung eines vollständig parametrisierten Roboterprogramms besteht wiederum aus zwei Hauptkomponenten. Eine Zustandsmaschine realisiert den Programmablauf und überwacht Übergangs- und Abbruchbedingungen. Auf die Eigenschaften der eingesetzten Zustandsmaschine und dem Bezug zu dem skillbasierten Programmierkonzept wird in 2.2 eingegangen. Eine Regelungskomponente realisiert das dem aktuellen Zustand entsprechende Roboterverhalten. Durch den Einsatz der iTaSC Formalismus wird eine Regelungskomponente ermöglicht, die keine roboterspezifischen Implementierungen beinhaltet. Lediglich die kinematische Beschreibung muss zur Ausführung vorliegen.

Die Komponenten der Ausführungszeit sind weiter mit den zu steuernden Geräten bzw. den Sensoren verbunden. Die hierbei eingesetzte Hardwareabstraktion beruht für die experimentelle Untersuchung auf ROS Kommunikationsmechanismen. Somit werden Befehle der Regelungskomponente an ROS Knoten weitergegeben, die als gerätespezifische Treiber den Befehl ausführen. Prinzipiell kann aber auch jegliche andere Kommunikationsinfrastruktur angesetzt werden. Ein Überblick über den groben Systemaufbau wird in Abbildung 2.1 gegeben.

2.2 Skillbasierte Programmierung

Skillbasierte Programmierung wird eingesetzt, um eine Roboteraufgabe in einer verallgemeinerbaren Form beschreiben zu können. Die Bezeichnung *Skill* bezieht sich auf eine Fähigkeit des Roboters und nicht auf spezifische Befehle. Somit dient die skillbasierte Programmierung als Brücke zwischen der Roboteraufgabe und der Roboterprogrammiersprache. Durch die abstrakte Darstellung ergeben sich intuitive Möglichkeiten, dass ein Benutzer Roboterverhalten, d. h. Skills, kombiniert und wiederverwendet, ohne dass auf die roboterspezifischen Eigenheiten eingegangen werden muss.

Der klassische Ansatz zur Programmierung von Industrierobotern bedient sich einer auf die Roboterbewegungsbefehle ausgerichteten prozeduralen Programmiersprache. Typischerweise können in einer dafür ausgelegten Programmieroberfläche Sequenzen von Bewegungsbefehle wie bspw. Achswinkelbewegungen (PTP) und Linearbewegungen (LIN) erstellt werden. Durch direkte Auswahl auf dem Eingabegerät können die meisten Befehle ohne Texteingabe spezifiziert werden. Um die Robotersteuerung ebenfalls zur Kontrolle weiterer Geräte einsetzen zu können, sind auch Befehle verfügbar, welche keine direkte Roboterbewegung erzeugen. Beispiele hierfür sind das Schalten von digitalen Ein- und Ausgängen (IO) zur Steuerung von externen Werkzeugen oder Lichtampelanlagen.

Mit zunehmender Komplexität der Steuerungsprogramme werden auch Kontrollstrukturen wie *Schleifen* oder *bedingte Verzweigungen* oder teilweise selbst Nebenläufigkeiten durch bspw. *Mul-*

tithreading von Teilprogrammen nötig. Um die Komplexität weiterhin kontrollieren zu können und so die langfristige Wartbarkeit der Programme zu garantieren, bildete sich der Wunsch nach einem moderneren Programmierparadigma, welches auf die grundlegende Funktion eines Industrieroboters angepasst ist. Angelehnt an dem Konzept der *Objektorientierter Programmierung* OOP entstand der Ansatz der sogenannten *skillbasierten* Roboterprogrammierung. Eine einheitliche Definition von *Skills* fehlt bislang in der Literatur.

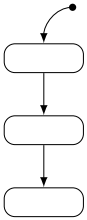
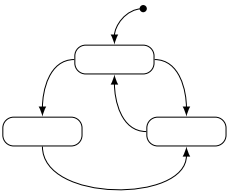
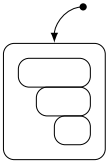
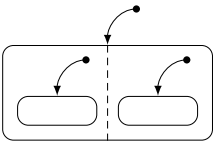
In dieser Arbeit bezeichnet ein *Skill* eine in sich vollständig gekapselte zielgerichtete Fähigkeit oder Funktionalität des Roboters. Wird ein Skill ausgeführt, so verhält sich der Roboter in einer geeigneten Weise, um das Ziel unter Einsatz der Fähigkeit zu erreichen. Beispielsweise beschreibt der Skill `Push` die Fähigkeit des Roboters eine definierte Kraft aufzubringen und zu stabilisieren. Mit den Zielparametern des Kraftwertes, -richtung und dem Werkzeugkoordinatensystem veranlasst die Ausführung des Skill `Push` das Roboterwerkzeug in Richtung des Kraftvektors so zu verfahren, dass sich der gegebene Kraftwert einstellt. Ein Skill ist nur genau dann ausführbar, wenn alle Parameter gegeben sind.

Konkret bedeutet diese Skilldefinition für das *pitasc* Rahmenwerk, dass ein Skill alle nötigen Angaben zur Erstellung einer iTaSC Konfiguration und alle Zielvorgaben in Form von Aufgabenraumzwangsbedingungen enthalten müssen, um ausführbar zu sein. Zusätzlich können einem Skill Methoden zur Überwachung von internen oder externen Signalen (sog. *Monitore*) und auf monitorausgabenbedingte Transitionen zu anderen Skills (sog. *Transitionen*) angefügt werden. Weiter beinhaltet jeder Skill einen einzusetzenden Regler mit ggf. weiteren Parametern. In trivialen Fällen, in denen keine Regelschleife geschlossen werden muss, wird das Regelungsglied als Durchgriff realisiert.

Zusammensetzte Skills ergeben sich als gekapselte Zustandsmaschinenstrukturen, wobei jeder Skill einem Zustand entspricht. Bei der Ausführung eines aus Skills aufgebauten Programms wird die endliche Zustandsmaschine instanziiert, wobei die von den Skills ausgelösten Transitionen durch einen Zustandsmaschinenkoordinator aufgegriffen und ausgeführt werden. Es ergibt sich ein Roboterprogramm als eine hierarchische Zustandsmaschine, aufgebaut aus ggf. wiederum zusammengesetzten Skills.

Eine Besonderheit des in dieser Arbeit angesetzten Skillkonzepts sind die Vereinigungsformen, in denen zusammengesetzte Skills auftreten können. Wie bereits erwähnt bestehen Skills aus gekapselten Zustandsmaschinen. Darin sind einzelne Skills, lineare Sequenzen oder komplexe (verschachtelte) Zustandsübergangsdiagramme inbegriffen. Des Weiteren können mehrere Skills gleichzeitig aktiv sein. Gleichzeitige Skills müssen immer gegeneinander priorisiert werden, um Konflikte durch gleichzeitigen Zugriff auf geteilte Freiheitsgrade des Roboters auflösen zu können. Besteht kein Konflikt, spielt die Priorisierung keine Rolle.

Tabelle 2.1: Visualisierung von möglichen Strukturen zusammengesetzter Skills

Einzel	Sequenz	Gruppe	Hierarchie	Parallel
				

Die Motivation der priorisierten Zusammensetzung wurde in (Halt et al. 2018b) erarbeitet. Als Inspiration wird die VDI Richtlinie 2860 “Montage- und Handhabungstechnik, [...]; Begriffe, Definitionen, Symbole” (VDI 2860 1990) herangezogen. Hierbei wird eine Handhabungsaufgabe in eine Folge von Teilfunktionen sog. *Prozessen* aufgelöst. Bei feinster Detaillierung wird von *elementaren Prozessen* gesprochen. Diese *Prozesse* werden zur Beschreibung der Handhabungsaufgabe zu gleichzeitig auszuführenden Prozessschritten und letztendlich zu kompletten Beschreibungen von Anlagenabläufen zusammengefügt.

In Anlehnung an die VDI Richtlinie 2860 definieren sich für die skillbasierte Roboterprogrammierung sechs elementare Prozesse, mit denen typische Montageaufgaben beschrieben werden können. Die Prozesse werden im Folgenden als Skills interpretiert und hierfür formal mit Zustandsüberwachung (*Monitoren*) und bedingte Zustandsübergangstransitionen (*Transitionen*) erweitert, da diese in der ursprünglichen Beschreibung der VDI Richtlinie 2860 nicht betont werden. Weiter werden jedem Prozess entsprechende Arbeitsraumtransformationen zugeordnet, welche die definierten Bedingungen in den Achswinkelraum des Roboters und die Messungen in den Arbeitsraum übersetzen.

Die symbolische Repräsentationen der Prozesse, sowie die Beschreibung der entsprechenden Verhalten sind in Tabelle 2.2 zusammengefasst. Die Arbeitsraumkoordinaten $\boldsymbol{\chi}_\mathrm{f}$ bezeichnen die Koordinaten, in dem die entsprechenden Bedingungen definiert werden. Dabei bezeichnen x, y, z kartesische Positionen und $\dot{x}, \dot{y}, \dot{z}$ die jeweiligen Richtungsgeschwindigkeiten. α, β, γ bezeichnen Eulerwinkel der Orientierung und $\dot{\alpha}, \dot{\beta}, \dot{\gamma}$ deren zeitliche Ableitung. Der Aufgabenraum der Kraftregelung wird mit den Kraftkoordinaten f_x, f_y, f_z und den Momenten m_x, m_y, m_z bezeichnet. Aus praktischen Gründen können durch den Einsatz von generalisierten Koordinaten paarweise auftretenden linearen und rotatorischen Skills zusammengefasst behandelt werden. Für den Skill Move to ergeben sich bspw. die Aufgabenraumkoordinaten $\boldsymbol{\chi}_\mathrm{f} = [x, y, z, \alpha, \beta, \gamma]$.

Tabelle 2.2: Symbolische Repräsentation und Beschreibung der elementaren Prozesse nach (Halt et al. 2018a)

	Symbol	Skill-Name	Aufgabenraumkoordinaten	Verhalten
(a)		Move	$\boldsymbol{\chi}_\mathrm{f} = [\dot{x}, \dot{y}, \dot{z}]^\mathsf{T}$	Lineare kartesische Bewegung mit gegebener Geschwindigkeit
(b)		Move (rotational)	$\boldsymbol{\chi}_\mathrm{f} = [\dot{\alpha}, \dot{\beta}, \dot{\gamma}]^\mathsf{T}$	Lineare kartesische Rotation mit gegebener Winkelgeschwindigkeit
(c)		Move to	$\boldsymbol{\chi}_\mathrm{f} = [x, y, z]^\mathsf{T}$	Lineare kartesische Bewegung zu einer gegebenen Zielposition
(d)		Move to (rotational)	$\boldsymbol{\chi}_\mathrm{f} = [\alpha, \beta, \gamma]^\mathsf{T}$	Lineare kartesische Drehung zu einer gegebenen Orientierung
(e)		Push	$\boldsymbol{\chi}_\mathrm{f} = [f_x, f_y, f_z]^\mathsf{T}$	Aufbringen einer gegebenen Kraft
(f)		Push (rotational)	$\boldsymbol{\chi}_\mathrm{f} = [m_x, m_y, m_z]^\mathsf{T}$	Aufbringen eines gegebenen Drehmomentes

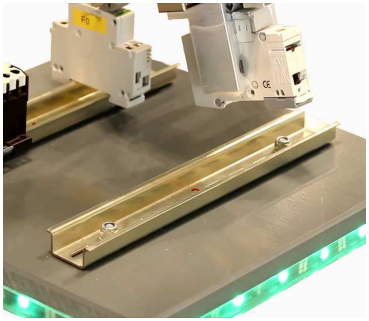
Zwei Montageanwendungen sind in Tabellen 2.3 und 2.4 dargestellt. Auf der jeweils rechten Seite des Fotos ist eine stichpunktartige verbale Beschreibung des Ablaufs gegeben. Darunter die entsprechende symbolische Repräsentation in Anlehnung an die VDI Richtlinie 2860 und er entsprechenden hierarchisch kombinierten Zustandsmaschine. Die Darstellung als Zustandsmaschine betont den prozeduralen Ablauf der Robotersteuerung, wogegen die Darstellung nach der VDI Richtlinie 2860 die Geräteunabhängigkeit hervorhebt. Trotzdem sind beide Darstellungen qualitativ gleichwertig und können ineinander überführt werden.

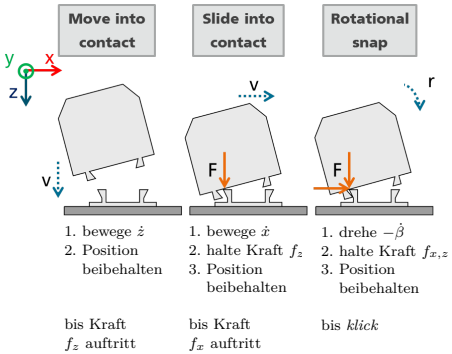
Beide Anwendungen dienen als repräsentative Beispiele für Kategorien des *Fügen durch Zusammensetzen* mit den Untergruppen *Einlegen*, *Ineinanderschieben* und *Einhängen* (DIN 8593-1 2003) sowie *Fügen durch An- und Einpressen* mit den Untergruppen *Schrauben*, *Verstiften*, *Verkeilen* und *Verspannen* (DIN 8593-3 2003). Somit wird ein Großteil der in DIN 8593 beschriebenen Vorgänge durch die beiden Beispiele abgedeckt.

Der Anwendungsfall *Hutschienenmontage* in Tabelle 2.3 beschreibt das Aufklipsen eines Hutschienenelementes bei der Schaltschrankmontage. Zunächst wird das Hutschienenelement der Hutschiene angenähert und ein Kontakt aufgebaut. Hierfür wird eine ausschließlich vertikale Bewegung auf die Richtung der Hutschiene ausgeführt. Ein entstandener Kontakt zwischen dem Hutschienenelement und der Hutschiene wird gehalten und gleichzeitig das Hutschienenelement so verschoben, dass der hintere Haken des Hutschienenelements in die Hutschiene greift. Der Kontakt des Hakens mit der Hutschiene wird nun gehalten und eine Drehung um die Achse des

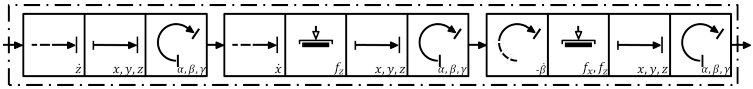
Tabelle 2.3: Beschreibung und symbolische Darstellung zusammengesetzter Roboterprogramme für das Beispiel einer Hutschienenmontage

Hutschienenmontage

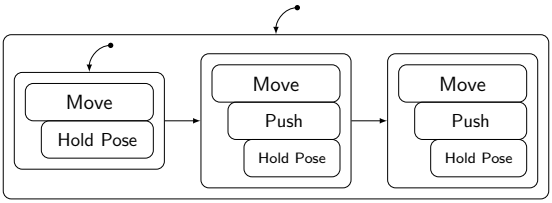




Symbolische Darstellung nach VDI Richtlinie 2860



Symbolische Darstellung der hierarchisch kombinierten Zustandsmaschine



Kontaktpunktes entlang der Hutschiene durchgeführt. Nachdem der vordere Haken eingerastet ist, wird der Montageprozess beendet.

Die symbolische Repräsentation des zuvor verbal beschriebenen Ablaufs gliedert sich ebenfalls in drei Abschnitte. Eine kartesische Geschwindigkeitsvorgabe in Z-Richtung des Skills Move bedingt die vertikale Absenkung des Hutschienenelements, während der niedrige priorisierte (linear und rotatorisch zusammengefasste) Skill Hold Pose eine Veränderung der kartesischen Ausrichtung verhindert. Durch die hierarchische Anordnung überschreibt die Geschwindigkeitsvorgabe des Skills Move die Bedingung des Skills Hold Pose in diesem Freiheitsgrad. Skill Hold Pose bezeichnet hierbei eine Erweiterung des Skills Move to um die Zielpose der bei Eintritt in den Skill eingenommenen Pose (Vgl. Tabelle A.1).

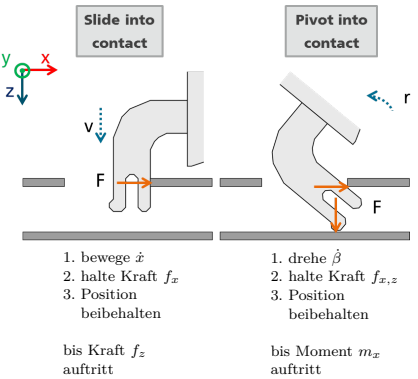
Der nächste Abschnitt zum Aufbau des Kontaktes zwischen hinteren Haken und Hutschiene besteht wiederum aus einer Kombination einer kartesischen Geschwindigkeitsvorgabe in X-Richtung mit einen Skill Push zur Stabilisierung des zuvor aufgebauten Kontakts. Wie zuvor wird eine Veränderung bisher nicht genutzter kartesischen Freiheitsgrade durch einen niederprioren Skill Hold Pose sichergestellt. Für den letzten Einrastschritt kombiniert sich eine kartesische rotatorische Geschwindigkeitsvorgabe um die Y-Achse mit der Stabilisierung der Kontaktkräfte sowohl in X als auch in Z-Richtung. Wieder wird jegliche Veränderung nicht genutzter kartesischer Freiheitsgrade durch einen niederprioren Skill Hold Pose verhindert.

Der Anwendungsfall *Türgriffmontage* in Tabelle 2.4 beschreibt einen Einfügevorgang zur Montage eines Autotürgriffs in den Schließmechanismus. Der Türgriff wird solange in den Schließmechanismus eingesenkt bis der innere Boden erreicht wurde, wobei stets der Kontakt zwischen dem Türgriff und der seitlichen Innenwand des Schließmechanismus gehalten wird. Im zweiten Schritt wird der Türgriff geschwenkt während das Fügeteil am inneren Boden entlanggleitet. Es entsteht eine Einführbewegung des Türgriffs in den Schließmechanismus. Die komplexe Bewegung aus der Drehung um einen sich bewegenden Drehpunkt ist vergleichbar mit einer an einer Wand stehenden herabgleitenden Leiter. Mit jeder Drehung kann der Türgriff tiefer in den Schließmechanismus eindringen, was wiederum weitere Drehung erlaubt. Die Bewegung endet, wenn keine Drehung mehr durchgeführt werden kann und ein Moment auftritt.

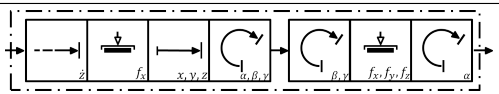
Die symbolische Beschreibung der Türgriffmontage gliedert sich in zwei Abschnitte. Die erste Absenkung in den Schließmechanismus ist strukturell gleich mit der Absenkung des Hutschienenelements auf die Hutschiene. Eine dominante Bewegung im zweiten Schritt ist die Drehung um die Y-Richtung. Gleichzeitig werden die Kontakte in X und Z-Richtung stabilisiert. Um ein seitliches Verhaken zu verhindern wird ebenfalls die Kontaktkraft in Y-Richtung zu Null reguliert, also ein starker Kontakt vermieden. Bei diesem Ablauf ist die Änderung der kartesischen Koordinaten erwünscht, obwohl diese nicht explizit angegeben wurden. Lediglich eine

Tabelle 2.4: Beschreibung und symbolische Darstellung zusammengesetzter Roboterprogramme für das Beispiel einer Türgriffmontage

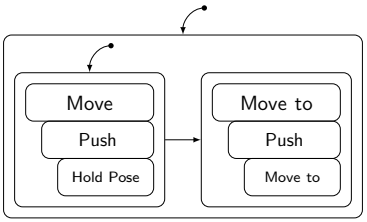
Türgriffmontage



Symbolische Darstellung nach VDI Richtlinie 2860



Symbolische Darstellung der hierarchisch kombinierten Zustandsmaschine



ungewollte Drehung des bisher ungenutzten Freiheitsgrads der Rotation um X wird mit einem niederprioren Skill Hold Pose vermieden.

Die in den Anwendungen zusammengesetzten Skills können zur Wiederverwendung in anderen Anwendungsfällen zugänglich gemacht werden. Es bildet sich eine zunehmend anwendungsspezialisierte Bibliothek an Skills heraus (Nägele et al. 2019). In Tabelle A.1 ist ein Überblick mit Verhaltensbeschreibung der in dieser Arbeit eingesetzten Skills aufgeführt.

Die Spezifikationssprache (DSL) zur textuellen Beschreibung einer Roboterapplikation wurde in (Nägele et al. 2018) präsentiert. Ziel der DSL Entwicklung war es eine roboterunabhängige Modellierung des Programms mit einem möglichst hohen Grad an Wiederverwendung, Individualisierbarkeit und inkrementeller Spezifizierung von Programmteilen zu ermöglichen.

Hierfür wurden zunächst die essentiellen modellierbaren Bestandteile komplexer Montageskills als *kinematischen Elemente* und *Ablaufbeschreibungselemente* identifiziert. Diese Elemente folgen formal einer systematischen Komposition zur Zusammenführung von bspw. ggf. unvollständigen Ablaufbeschreibungselementen und sich gegenseitig ergänzenden weiteren Elementen. Eine so entstehende Modellierung des gesamten Programmablaufs wird in Konfigurationen der iTaSC Implementierung gewandelt und den Zuständen der auszuführenden Zustandsmaschine zugeordnet.

Durch die Kapselung der zusammengeführten Elemente mit Standardwerten und -parametern entstehen erweiterbare Skills, die mit zunehmender Komplexität eine immer spezialisiertere Bibliothek an vorgefertigten Skills bildet. Auf die sich damit ergebende konkrete Bibliothek an Skills und deren Komposition aus aufeinander aufbauenden Skills wird in (Nägele et al. 2019) eingegangen. Die in der DSL eingesetzte Form des OOP Paradigmas wird als *prototypenbasierte Programmierung* bezeichnet. Diese, auch als *klassenlose Objektorientierung* bezeichnete Form birgt den konkreten Vorteil, dass keine Unterscheidung zwischen einer Klassendefinition (Interface) und einer Instanz beachtet werden muss, da vollständige Kopien (sog. Klone) Instanziierungen ersetzen. Dadurch entfällt der doppelte Wartungsaufwand einer *Interface-Instanz* Kombination und die Weitergabe von voreingestellten Standardwerten durch einen zu erweiternden Skill wird vereinfacht.

2.3 Steuerungstechnische Umsetzung

Die meisten skillbasierten Roboterprogrammierkonzepte brechen das auszuführende Programm auf einen finiten Satz an Funktionsblöcken herunter, wenden dieses auf die jeweilige roboterspezifische Implementierungen an und führen diese koordiniert aus. In dieser Arbeit hingegen wird

das Verhalten der Skills während der Laufzeit in entsprechende Achsgeschwindigkeitsvorgaben umgewandelt. Um einen Roboter entsprechend der Skills zu verfahren wird somit keine besondere Anforderung an die Robotersteuerung erhoben, sondern nur die (triviale) Anforderung einer möglichen Achsbewegung vorgegeben.

Die Roboterskills bzw. -aufgaben werden durch den *Aufgabenraumansatz* (engl.: Task Frame Formalism TFF) iTaSC (engl.: instantaneous Task specification and control) zur Ausführungszeit in Steuersignale umgewandelt. Somit ist nur die tieferliegende Kommunikationsschicht zwischen Steuerungscomputer und Achsreglern roboterspezifisch.

Neben der Verallgemeinerung der Roboterprogramme und die somit höhere Wiederverwendbarkeit mit verschiedenen Robotern und Kinematiken, ergibt sich der Vorteil komplexe Bewegungen durch die Kombination von Skills zu erreichen, ohne auf roboterspezifischen Funktionen eingehen zu müssen. Dadurch ergeben sich elegante und kompakte Formulierungen, die besonders für die automatisierte Montage geeignet sind.

Im Folgenden wird die Funktionsweise und mathematische Formulierung des iTaSC Formalismus eingeführt. Zusätzlich wird auf die hier angewendete Realisierung eingegangen.

2.3.1 Einführung des Aufgabenraumansatzes iTaSC

Der iTaSC Formalismus dient als allgemeine Transformation zwischen einem *Aufgabenraum* (engl.: task space) und dem *Achswinkelraum*[2] des Roboters (engl.: joint space). Die Ausformulierung des Aufgabenraums ist hierbei offen gehalten. Im gewählten Aufgabenraum werden *Bedingungen* (engl.: constraints) so gewählt, dass sich das gewünschte Roboterverhalten ergibt.

iTaSC modelliert das Optimierungsproblem der Aufgabenraumtransformation unter Einsatz von *Schleifen* (engl. loops), d. h. geschlossenen Transformationsketten. Mit sogenannten *virtuellen kinematischer Ketten* (engl.: virtual kinematic chain: VKC) werden die geometrischen Beziehungen zwischen *Objektkoordinaten* (engl.: object frames) beschrieben. Objektkoordinaten sind aufgabenrelevante Koordinatensysteme welche bspw. an das kontrollierbare Roboterwerkzeug oder das Werkstück gebunden sind und deren geometrische Beziehung beschreiben. Die VKCs definieren die den Objektkoordinaten entsprechenden *Aufgabenraumkoordinaten* (engl.: feature coordinates).

Zu betonen sei, dass sich jedes Koordinatensystem notwendigerweise auf ein Basiskoordinatensystem beziehen muss. Somit wird ein Koordinatensystem stets als Transformation aus dem

[2] häufig auch *Konfigurationsraum* genannt

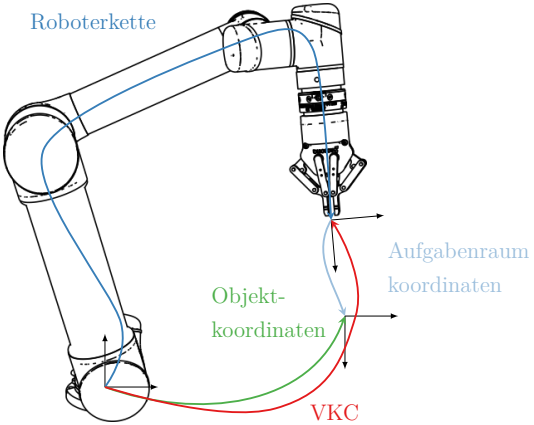


Abbildung 2.2: Schematische Darstellung einer virtuellen kinematischen Kette, einer Objekt- und Aufgabenraumtransformation

entsprechenden Basiskoordinatensystem angesehen. Zur verbalen Verdeutlichung dieser Transformation wird von *kinematischen Ketten* (engl.: kinematic chain) gesprochen. Eine besondere Bezeichnung verdienen die Objektkoordinaten der Roboterkinematik. Offensichtlich ist die Koordinatentransformation von der Roboterbasis zum Roboterwerkzeug von dem Vektor der Roboterachsstellungen abhängig. Diese Objektkoordinaten werden als *Roboterkette* bezeichnet.

Die Aufgabenraumkoordinaten sollten so gewählt werden, dass die zu spezifizierenden Bedingungen möglichst direkt beschrieben werden können. So können beispielsweise Abstände zwischen dem Roboterwerkzeug und dem Werkstück anwendungsspezifisch in dem kartesischen-, Sphärischen- oder Zylinderraum gewählt werden. Die meist eingesetzten Arbeitsräume sind der kartesische Raum, der Kraftraum und der Achswinkelraum. Bedingungen im kartesischem Raum bestimmen kartesische Abstände zwischen kontrollierbaren Koordinaten, bspw. eines Roboterwerkzeugs und einer Zielkoordinate. Die hierfür notwendige Transformation entspricht der klassischen Vorwärts- und Rückwärtstransformation. Da in vielen Fällen der Abstand zwischen den Koordinaten minimiert werden soll (in anderen Worten: die Zielpose soll möglichst genau erreicht werden), kann auch von einer kartesischen Abstandstransformation gesprochen werden.

Eine einfache Möglichkeit einen Abstand der kartesischen Orientierung zu bilden ist die Betragsnorm der Differenz der jeweiligen Eulerwinkelrepräsentation des *Roll-, Nick- und Gierwinkel*. Dadurch kann die Abstandsberechnung der Orientierungen identisch zu der Berechnung von kartesischen Positionen durchgeführt werden. Allerdings ergeben sich durch die Eulerwinkelre-

präsentation für die Bewegung drei linear voneinander unabhängige Drehrichtungen, welche sich gegenseitig stören können. Außerdem unterliegt die Eulerwinkelrepräsentation dem sog. *Gimbal Lock*, dem Verlust eines Freiheitsgrads in bestimmten Konfigurationen. Um diese Probleme zu vermeiden, jedoch dem Benutzer den intuitiveren Zugang zu Roll-, Nick- und Gierwinkeln zu erhalten, wurde eine interne Repräsentation der Orientierung mit Einheitsquaternionen mit angepasster Jacobi-Matrix in (Stolt 2012) vorgestellt.

Im Kraftraum werden zu wirkende Kräfte[3] beschrieben. Die Kraftbedingungen ergeben sich aus dem Betrag der Kräfte und der Wirkungsrichtung in einem gegebenen kartesischen Koordinatensystem.

Zuletzt sei noch der Achswinkelraum selbst erwähnt. Für direkte Achswinkelbedingungen, d. h. eine PTP Fahrt des Roboters, ist keine direkte Transformation nötig. Allerdings sind die (physikalischen) Auswirkungen von Achswinkelzielvorgaben gravierend von der spezifischen Roboterkinematik abhängig. Obwohl Vorgaben im Achswinkelraum prinzipiell übertragen werden können, sollten Skills mit direkten Achswinkelbedingungen als roboterspezifisch betrachtet werden. Werden die Achswinkelzielvorgaben während der Laufzeit von kartesischen Zielposen abgeleitet, können übertragbare PTP Bewegungen realisiert werden. Durch die hierbei erforderliche Lösung der Rückwärtskinematik sind allerdings die sich ergebenden Achswinkelzielvorgaben nicht eindeutig.

Istwerte in den entsprechenden Aufgabenraumkoordinaten müssen während der Laufzeit bekannt bzw. messbar sein. iTaSC führt des Weiteren *Unsicherheitskoordinaten* (engl.: uncertainty coordinates) $\boldsymbol{\chi}_\mathrm{u}$ ein. Durch diese Koordinaten wird es möglich, Transformationen mit einzubeziehen, die unsicheren geometrischen Parametern unterliegen. Diese Parameter können während der Laufzeit durch geeignete Schätzer (bspw. Kalman Filter (Stolt 2012)) angenähert werden. Daher unterliegen sie prinzipiell einem zeitlichen Verlauf und müssen bei der zeitlichen Ableitung einbezogen werden. In dieser Arbeit werden geometrische Unsicherheiten vernachlässigt, wodurch dieser Berechnungspfad praktisch entfällt.

Es folgt die mathematische Herleitung des iTaSC Formalismus. Für die Ausgangsgleichung wird allgemein eine Übertragungsfunktion $\boldsymbol{f}$ mit

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{q}, \boldsymbol{\chi}_\mathrm{f}) \ . \tag{2.1}$$

angenommen. $\boldsymbol{f}$ ist allgemein abhängig von den Achswinkeln $\boldsymbol{q}$ des Roboters und den Aufgabenraumkoordinaten $\boldsymbol{\chi}_\mathrm{f}$.

[3] In dieser Arbeit wird von generalisierten Kräften ausgegangen. Generalisierte Kräfte beschreiben allgemein die Wechselwirkung durch Krafteinwirkung und beinhalten sowohl Kräfte als auch Momente.

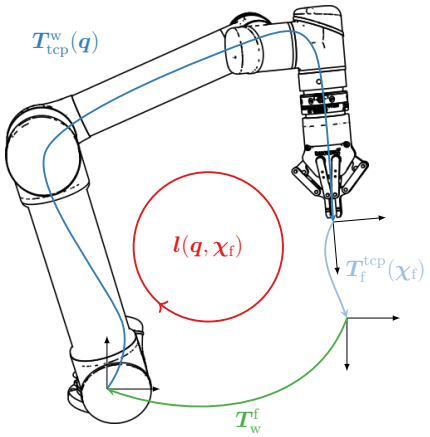


Abbildung 2.3: Schematische Darstellung einer geschlossenen kinematischen Schleife

Die Ausgangsgleichung (2.1) wird zeitlich abgeleitet und ergibt die Ausgangsgleichung auf Geschwindigkeitsbasis zu

$$\dot{\boldsymbol{y}} = \boldsymbol{C}_\mathrm{q}\dot{\boldsymbol{q}} + \boldsymbol{C}_\mathrm{f}\dot{\boldsymbol{\chi}}_\mathrm{f} \tag{2.2}$$

mit $\boldsymbol{C}_\mathrm{q} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}}$ und $\boldsymbol{C}_\mathrm{f} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\chi}_\mathrm{f}}$. Die Jacobi-Matrizen $\boldsymbol{C}_\mathrm{q}$ und $\boldsymbol{C}_\mathrm{f}$ werden als Auswahlmatrizen bezeichnet. Diese wählen oder linearkombinieren die Aufgabenraumkoordinaten. Bei geeigneter Wahl der Aufgabenraumkoordinaten sind die Auswahlmatrizen eine Permutation der Einheitsmatrix.

Die kinematische Schleife $\boldsymbol{l}$ ergibt sich aus der geschlossenen Verkettung der VKC und der Robotertransformation. Für die dargestellte Schleife in Abbildung 2.3 verläuft von der Roboterachswinkel $\boldsymbol{q}$ abhängigen Robotertransformation $\boldsymbol{T}^{\mathrm{w}}_{\mathrm{tcp}}(\boldsymbol{q})$ von der Roboterbasis bis zum Roboterwerkzeug, über die von den Aufgabenraumkoordinaten $\boldsymbol{\chi}_\mathrm{f}$ abhängigen Aufgabenraumtransformation $\boldsymbol{T}^{\mathrm{tcp}}_{\mathrm{f}}(\boldsymbol{\chi}_\mathrm{f})$ von dem Roboterwerkzeug bis zu den Werkstückvariablen und der Objekttransformation $\boldsymbol{T}^{\mathrm{f}}_{\mathrm{w}}$.

In einer geschlossenen Schleife gilt für die Verkettung von homogenen Transformationen

$$\tilde{\boldsymbol{l}}(\boldsymbol{q}, \boldsymbol{\chi}_\mathrm{f}, \boldsymbol{\chi}_\mathrm{u}) = \boldsymbol{I} \tag{2.3}$$

mit $\boldsymbol{I}$ als Einheitsmatrix oder für eine kompaktere Darstellung mit dem Nullvektor $\mathbf{0}$, beispielsweise durch generalisierte Koordinaten,

$$\boldsymbol{l}(\boldsymbol{q}, \boldsymbol{\chi}_\mathrm{f}, \boldsymbol{\chi}_\mathrm{u}) = \mathbf{0} \ . \tag{2.4}$$

Eine geschlossen Schleife ist allgemein abhängig von der Roboterachsstellung $\boldsymbol{q}$, dem Zustand der Aufgabenraumkoordinaten $\boldsymbol{\chi}_\mathrm{f}$ und der aktuellen Schätzung der Unsicherheitskoordinaten $\boldsymbol{\chi}_\mathrm{u}$.

Die zeitliche Ableitung von (2.3) oder (2.4) führt zu der geschwindigkeitsbasierten Gleichung

$$\boldsymbol{J}_\mathrm{q}\dot{\boldsymbol{q}} + \boldsymbol{J}_\mathrm{f}\dot{\boldsymbol{\chi}}_\mathrm{f} + \boldsymbol{J}_\mathrm{u}\dot{\boldsymbol{\chi}}_\mathrm{u} = \mathbf{0} \tag{2.5}$$

mit den Jacobi-Matrizen $\boldsymbol{J}_\mathrm{q} = \frac{\partial \boldsymbol{l}}{\partial \boldsymbol{q}}$, $\boldsymbol{J}_\mathrm{f} = \frac{\partial \boldsymbol{l}}{\partial \boldsymbol{\chi}_\mathrm{f}}$ und $\boldsymbol{J}_\mathrm{u} = \frac{\partial \boldsymbol{l}}{\partial \boldsymbol{\chi}_\mathrm{u}}$.

Für die analytische Lösung von (2.5) nach $\dot{\boldsymbol{\chi}}_\mathrm{f}$ ergibt sich

$$\dot{\boldsymbol{\chi}}_\mathrm{f} = -\boldsymbol{J}_\mathrm{f}^{+}(\boldsymbol{J}_\mathrm{q}\dot{\boldsymbol{q}} + \boldsymbol{J}_\mathrm{u}\dot{\boldsymbol{\chi}}_\mathrm{u}) \tag{2.6}$$

mit $\boldsymbol{J}_\mathrm{f}^{+}$ als generalisierte Inverse von $\boldsymbol{J}_\mathrm{f}$.

Wird Gleichung (2.6) in (2.2) eingesetzt und umgeformt, ergibt sich die modifizierte Ausgangsgleichung

$$\boldsymbol{\Psi}\dot{\boldsymbol{q}} = \dot{\boldsymbol{y}} + \boldsymbol{B}\dot{\boldsymbol{\chi}}_\mathrm{u} \tag{2.7}$$

mit $\boldsymbol{\Psi} = \boldsymbol{C}_\mathrm{q} - \boldsymbol{C}_\mathrm{f}\boldsymbol{J}_\mathrm{f}^{+}\boldsymbol{J}_\mathrm{q}$ und $\boldsymbol{B} = \boldsymbol{C}_\mathrm{f}\boldsymbol{J}_\mathrm{f}^{+}\boldsymbol{J}_\mathrm{u}$.

Da Gleichung (2.2) geschwindigkeitsbasiert ist, muss die (positionsbasierte) Regelung

$$\dot{\boldsymbol{y}} = \dot{\boldsymbol{y}}_d^{\circ} = f_\mathrm{Regelung}(\dot{\boldsymbol{y}}_d, \boldsymbol{y}_d, \boldsymbol{y}_m) \tag{2.8}$$

eingeführt werden, um Fehler bei der Integration, Modellfehler und Störungen ausgleichen zu können.

Unter Vernachlässigung der Roboterdynamik kann das zu steuernde System als ein ideales Geschwindigkeits-Übertragungsglied

$$\dot{\boldsymbol{q}} = \boldsymbol{u} = \dot{\boldsymbol{q}}_d \tag{2.9}$$

angesehen werden, bei dem die Stellgröße $\boldsymbol{u}$ den gewünschten Achsgeschwindigkeiten $\dot{\boldsymbol{q}}_d$ entspricht.

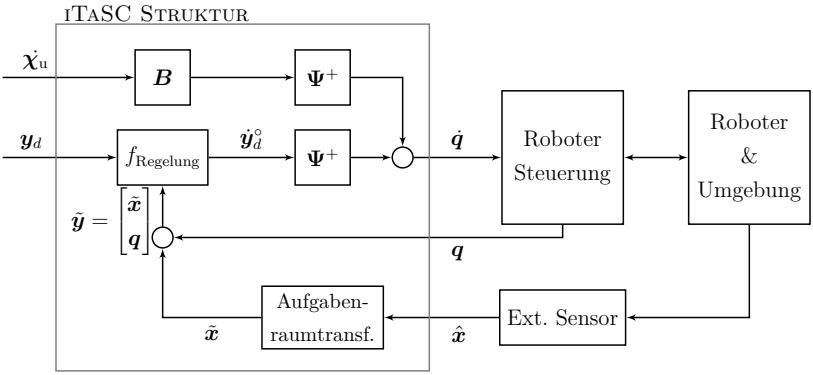


Abbildung 2.4: Blockdiagramm der iTaSC Struktur zur Aufgabenraumtransformation

Die gewünschten Soll-Achsgeschwindigkeiten $\dot{\boldsymbol{q}}_d$ können mit den Gleichungen (2.7), (2.8) und (2.9) zu

$$\dot{\boldsymbol{q}}_d = \boldsymbol{\Psi}^+(\dot{\boldsymbol{y}}_d^\circ + \boldsymbol{B}\dot{\boldsymbol{\chi}}_\mathrm{u}) \tag{2.10}$$

bestimmt werden. $\boldsymbol{\Psi}^+$ beschreibt hierbei eine geeignete Inverse der Matrix $\boldsymbol{\Psi}$, beispielsweise eine gewichteten generalisierte Inverse (Doty et al. 1993). Die Blockdarstellung der iTaSC Struktur ist in Abbildung 2.4 dargestellt.

Um gleichzeitige, priorisierte Erfüllung verschiedener Bedingungen zu ermöglichen kann die Invertierung der Matrix $\boldsymbol{\Psi}$ durch die *task priority strategy* (Nakamura et al. 1987) vorgenommen werden.

Unter der Vernachlässigung von geometrischen Unsicherheiten $\boldsymbol{\chi}_\mathrm{u} = \dot{\boldsymbol{\chi}}_\mathrm{u} = \boldsymbol{0}$ existieren für die Gleichung (2.10) die Lösung

$$\dot{\boldsymbol{q}}_d = \boldsymbol{\Psi}^+\dot{\boldsymbol{y}}_d^\circ + (\boldsymbol{I} - \boldsymbol{\Psi}^+\boldsymbol{\Psi})\boldsymbol{p} \tag{2.11}$$

mit einem beliebigen Vektor $\boldsymbol{p}$. Der zweite Summand der Lösung (2.11) projiziert den Vektor $\boldsymbol{p}$ in den Nullraum der Matrix $\boldsymbol{\Psi}$.

Mit den nach N Prioritäten sortierte Vektoren

$$\boldsymbol{\Psi} = [\boldsymbol{\Psi}_1, \boldsymbol{\Psi}_2, \ldots, \boldsymbol{\Psi}_N]^\intercal \text{ und} \tag{2.12}$$

$$\dot{\boldsymbol{y}}_d^\circ = [\dot{\boldsymbol{y}}_{d,1}^\circ, \dot{\boldsymbol{y}}_{d,2}^\circ, \ldots, \dot{\boldsymbol{y}}_{d,N}^\circ]^\intercal \tag{2.13}$$

werden für jede Priorität i die jeweilige Projektionsmatrix

$$\boldsymbol{P}_i = \boldsymbol{I} - \boldsymbol{\Psi}_{1,i}^{+}\boldsymbol{\Psi}_{1,i} \tag{2.14}$$

bestimmt, wobei die akkumulierte Matrix $\boldsymbol{\Psi}_{1,i}$ für alle höheren Prioritäten bis zur Priorität i zu

$$\boldsymbol{\Psi}_{1,i} = (\boldsymbol{\Psi}_1, \boldsymbol{\Psi}_2, \ldots, \boldsymbol{\Psi}_i)^{\mathsf{T}} \tag{2.15}$$

zusammenfasst.

Es ergibt sich die rekursive Lösungsstrategie für alle Bedingung der Priorität i

$$\dot{\boldsymbol{q}}_{d,1} = \boldsymbol{\Psi}^{\#}\dot{\boldsymbol{y}}_{d,1}^{\circ} \ , \tag{2.16}$$

$$\dot{\boldsymbol{q}}_{d,i} = \dot{\boldsymbol{q}}_{d,i-1} + (\boldsymbol{\Psi}_i\boldsymbol{P}_{i-1})^{\#}(\dot{\boldsymbol{y}}_{d,i}^{\circ} - \boldsymbol{\Psi}_i\dot{\boldsymbol{q}}_{d,i-1}) \tag{2.17}$$

mit $^{+}$ und $^{\#}$ als entsprechende, ggf. unterschiedlich gewichtete, generalisierte Inversen.

Hierbei werden nieder priorisierte Bedingungen i im ungenutzten Nullraum der höher priorisierten Bedingungen $i-1$ angewendet. Bedingungen mit höheren Prioritäten werden in der Anwendung somit nicht beeinflusst. Allerdings können niedriger priorisierte Bedingungen im übrigbleibenden Nullraum im Allgemeinen nicht vollständig erfüllt werden.

2.3.2 Praktische Anwendung und Implementierung

Der in Kapitel 2.3.1 beschriebene iTaSC Formalismus wird mit dem skillbasierten Ansatz aus Kapitel 2.2 eingesetzt. Eine durch die Anordnung der Skills definierte Zustandsmaschine konfiguriert durch den jeweils aktuellen Zustand eine auf iTaSC basierende Regelschleife. Bei einer Zustandsänderung werden die aktiven iTaSC Bedingungen umkonfiguriert und das Roboterverhalten ändert sich. Das Gesamtsystem *pitasc* besteht aus einer Spezifikationssprache, einer koordinierenden Zustandsmaschine, einer Regelung sowie Kommunikationsmechanismen zu Treibern oder weiteren externen Komponenten.

Im Folgenden sollen die Anbindung des *pitasc*-Systems (kurz: *pitasc*) an eine vorhandene Robotersteuerung und die Nutzung sowie Einschränkungen des Systems erörtert werden. Vorrangiges Ziel von *pitasc* ist ein Erweiterungssystem zu der bestehenden Robotersteuerung zu bilden. Der Anwendungsfokus liegt auf komplexen Bewegungsbahnen unter der gleichzeitigen Berücksichtigung von kraft-sensitiven Reaktionen.

Die *pitasc* Implementierung wird auf einem externen Steuerungs-PC ausgeführt. Wird durch die Robotersteuerung die Steuerung der Roboterachsen freigegeben, übernimmt *pitasc* die Regelungsaufgabe. Je nach Anwendung kann *pitasc* durch die bestehende Robotersteuerung aufgerufen werden und so als Zusatzfunktion des Roboters fungieren oder als übergeordnete Steuerung wahlweise robotereigene Funktionen und *pitasc* Regelungsaufgaben koordinieren.

Wird die Regelschleife durch eine Zustandsänderung umkonfiguriert, wird auch ein dem Skill zugehöriger Regelalgorithmus eingesetzt. Dadurch können jeweils passende Regelungsarchitekturen den Skills und somit den Roboterfähigkeiten zugeordnet werden.

Die Aufgabe des Bedieners, d. h. des Roboterprogrammierers, ist die Erstellung einer Abfolge von geeigneten Skills. Jeder Skill kann sich aus wiederum priorisierten Skills zusammensetzen. Jeder Skill trägt die jeweiligen Bedingungen zu dem multikriteriellen Optimierungsproblem bei, dessen Lösung die gewünschte Bewegung des Roboters ist.

Im direkten Vergleich zwischen auf Ausführungsprimitiven beruhenden skillbasierten Programmierumgebungen und *pitasc* können folgende Unterschiede aufgezeigt werden (Vgl. Tabelle 1.1):

- *pitasc* bedingt eine Roboterbewegung durch systematische reaktive Verbindung verschiedener Bewegungsbedingungen (Skills), die jeweils in verschiedene Bezugskoordinaten gegeben werden können. Dadurch können komplexe Montagebewegungen aus voneinander unabhängigen Fügebedingungen zusammengesetzt werden, wie bspw. die Fügebewegung unter Einhalten des Mehrpunktkontakts der *Türgriffmontage*.
- Die Beschreibung der Roboteraufgaben als Bedingungen in verschiedenen Aufgabenraumkoordinaten machen die Problembeschreibungen kompakter und leichter zugänglich für Prozessexperten ohne Robotik Hintergrund.
- Die implizite Programmierung mit Bedingungen ist unabhängig des einzusetzenden Roboters und ähnelt meist der textuellen Aufgabenbeschreibung oder dem Montageplan.
- Die resultierende Bewegung des Roboters resultiert als kontinuierliche Reaktion auf wahrgenommene Sensoreingänge und kann zwischen verschiedenen Roboter übertragen werden.

2.3.3 Einbindung anwendungsspezifischer Regler

Bei einer Zustandsänderung werden nicht nur die dem Skill eigenen Bedingungen, kinematischen Schleifen und Ketten umkonfiguriert, sondern auch jegliche zugehörigen Parameter. Ein

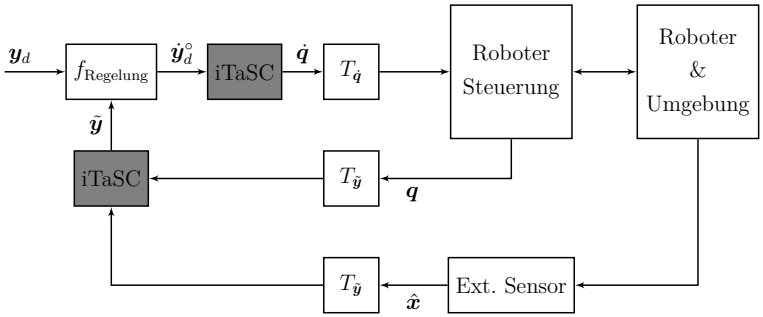


Abbildung 2.5: Das Blockdiagramm der Regelstrecke aus Sicht des Reglers. Die Aufgabenraumtransformation durch iTaSC ist dem Regler unbekannt und wirkt sich lediglich als Totzeit aus. Zur besseren Einordnung wurden die Hin- und Rücktransformationen als graue Boxen angedeutet.

parametrisierter Regler gehört zu den Skillparametern. Daher ergibt sich der Freiraum für den Bediener entweder den voreingestellten Regler oder einen alternativen Regler einzusetzen. Die Reglerimplementationen sind als Komponenten in der *pitasc* Bibliothek verfügbar und werden den entsprechenden Skills zugeordnet.

Aus Sicht des Reglers bildet sich die Regelstrecke durch die Roboterdynamik im Zusammenspiel mit der Umgebung. Die Aufgabenraumtransformation führt lediglich zu einer Totzeit aufgrund der Berechnungen. Dieser Verzug addiert sich zu möglichen kommunikationsbedingten Verzugszeiten T.

Der Regler f_{Regelung} hat generell keine Kenntnis über den Aufgabenraum, in dem er betrieben wird. Über die Anbindung an aufgabenspezifische Skills können allerdings spezialisierte, für die jeweilige Aufgabe ausgelegte Regler eingesetzt werden und so bessere Ergebnisse erzielt werden. Das Blockdiagramm in Abbildung 2.5 illustriert das Gesamtsystem. Die den Regler unbekannte Aufgabenraumtransformationen sind grau dargestellt.

2.4 Das Robot Operating System ROS

Die Realisierung des beschriebenen Ansatzes bedient sich Funktionen des *Robot Operating System (ROS)* (Quigley et al. 2009). Entgegen der Namensgebung handelt es sich bei ROS nicht um ein Betriebssystem. ROS bietet eine Softwarepaketorganisation mit der standardisierten Kompilierumgebung *catkin*. Hierdurch wird es möglich, mit Hilfe von automatisch erstellten Kommunikationsprotokollen zwischen verteilten Programmen zu kommunizieren.

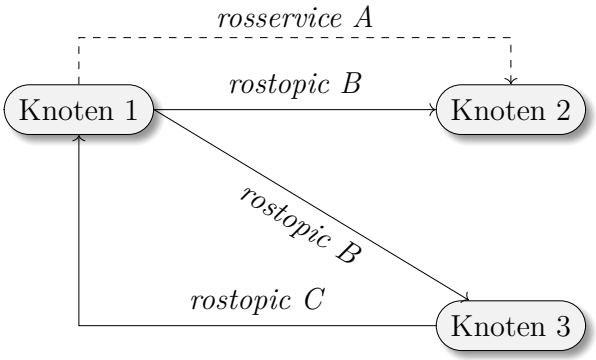


Abbildung 2.6: Beispiel einer ROS Kommunikation dreier ROS Knoten. *rostopics* werden mit durchgezogenen und *rosservices* mit gestrichelten Pfeilen dargestellt, wobei der Pfeil vom Sender zum Empfänger zeigt. Knoten 1 bietet und empfängt je einen Datenstrom per *rostopic*. Der ausgehende Datenstrom wird von Knoten 2 und Knoten 3 empfangen. Knoten 3 sendet einen anderen Datenstrom aus, welcher durch Knoten 1 empfangen wird. Zusätzlich bietet Knoten 2 einen *rosservice* an, welcher durch Knoten 1 angefordert wird.

Hierfür meldet sich ein Programm zur Laufzeit bei einem Vermittlerknoten (sog. `roscore`) an. Ein angemeldetes Programm wird als *ROS Knoten* bezeichnet. Der `roscore` ermöglicht dem Knoten auf eine allgemein zugängliche Parameterablage zuzugreifen und koordiniert die Kommunikation zwischen Knoten.

Prinzipiell sind zwei Kommunikationsmechanismen vorgesehen: Erstens eine TCP/IP basierte ungerichtete Übertragung (sog. *rostopics*), und zweitens eine XML-RPC basierte Übertragung von prozeduralen Aufrufen (sog. *rosservices*).

rostopics erlauben eine n-zu-n Kommunikation für kontinuierliche Datenströme und *rosservices* für Ereignis-basierte Einzelaufrufe. Eine beispielhafte Topologie mit drei Knoten ist in Abbildung 2.6 dargestellt. Eine verbreitete Erweiterung bildet die hybride Variante der `action_lib`. Hiermit werden ebenfalls Ereignis-basierte Einzelaufrufe jedoch mit kontinuierlicher Rückkommunikation aus verschiedenen *rostopic* Verbindungen realisiert.

Durch die Vereinheitlichung der Interprogrammkommunikation haben sich Standards in den übertragenen Datenstrukturen herausgebildet, welche es ermöglichen, einen eigenen Knoten mit einer großen Anzahl an frei verfügbaren Knoten zu verbinden und deren Funktionen in ein Gesamtsystem einzubinden. Hierdurch wird ein modularer Entwurf des Robotersystems möglich.

Es sind ROS Versionen für verschiedene Betriebssysteme verfügbar. Das in dieser Arbeit eingesetzte Betriebssystem ist *Ubuntu 16.04* mit der ROS Version *Kinetic Kame*.

ROS wird zur Verknüpfung des *pitasc* Gesamtsystem mit verschiedenen Geräten eingesetzt. Die gerätespezifische Kommunikation, bspw. zu der Robotersteuerung oder dem Kraft-Momenten-Sensoren, wird in einem ROS Knoten realisiert und die Daten über *rostopics* weitergegeben.

pitasc tritt in dem Zusammenhang der Bewegungsregelung als ein einzelner ROS Knoten auf. Einzelne Skills können ebenfalls weitere ROS Kommunikation aufbauen. So werden bspw. Transitionsbedingungen basierend auf externen Eingängen realisiert oder es können externe Dienste mittels *rosservices* angefordert werden. Letzteres wird eingesetzt um die Parametersynthese aus Kapitel 3.2 zu einem gegebenen Zeitpunkt während der Ausführung anzufordern.

2.5 Das Kalman Filter

Da in dieser Arbeit eine Variation eines erweiterten Kalman Filters als Modellschätzer eingesetzt wird, soll an dieser Stelle in kompakter Form auf die Funktionsweise eines Kalman Filters eingegangen werden. Auf probabilistische, sowie systemtheoretische Untersuchungen und Eigenschaften wird weitestgehend verzichtet. Der geneigte Leser sei bspw. auf (Thrun et al. 2006) verwiesen. Der hier dargestellte Ansatz richtet sich frei nach (Labbe 2014).

2.5.1 Die Funktionsweise

Ziel eines jeden Schätzers ist es, einen Zustand (bspw. eine Position) aus verrauschten Messdaten zu gewinnen. Hierzu stehen Sensoren zur Verfügung, mit denen sich Informationen über den Zustand ermitteln lassen. Allerdings muss davon ausgegangen werden, dass die wahrgenommenen Messdaten mit Rauschen oder Fehlmessungen überlagert sind. Es wird angenommen, dass ein mathematisches Systemmodell vorliegt, welches die Messungen in die Zustandsvariablen überführt. Auch dieses Modell beinhaltet Fehler, beispielsweise durch eine vereinfachte Modellannahme.

Es wird angenommen, dass sich das beobachtete System kausal verhält und ausreichend durch das Modell angenähert werden kann. Daher ist intuitiv einsichtig, dass nicht mit dem Modell vereinbare gemessene Zustände wahrscheinlicher als fehlerhaft angenommen werden sollten, als solche die einer Vorhersage durch das Modell entsprechen. Der Abstand zwischen der Vorhersage und der Messung gibt ein Maß der Unsicherheit. Im Gegenzug gelten Messungen, die mehr den Erwartungen entsprechen als vertrauenswürdiger.

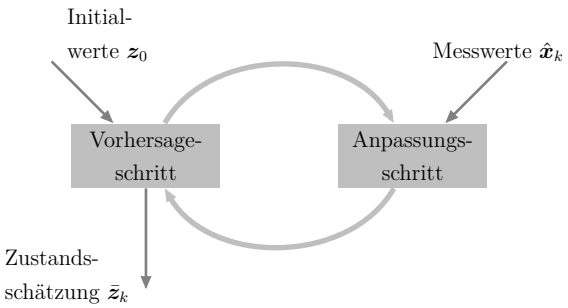


Abbildung 2.7: Interne Ablaufschritte eines g-h Filters nach (Labbe 2014)

Trotzdem müssen ebenfalls Voraussagefehler des Modells in Betracht gezogen werden. Selbst unter der Annahme eines Modells, welches die Systemveränderungen in jedem Zeitschritt perfekt beschreibt, beruht die Vorhersage auf eine Historie von vergangenen Zuständen. Letztendlich ist es unmöglich, die exakten Anfangszustände des Systems zu bestimmen, um das Modell perfekt zu initialisieren. In Anbetracht der anzunehmenden Modellungenauigkeiten muss Vorhersagen in der näheren Zukunft größere Bedeutung zugemessen werden, als solche die durch einen größeren zeitlichen Abstand einem größeren Schätzungsfehler unterliegen können.

Formuliert man aus den vorigen Überlegungen eine Funktionsweise, ergibt sich ein zweischrittiges Vorgehen:

1. Auf Basis der letzten bekannten Messung (oder eines Initalwerts) und unter Einbeziehung der Modellunsicherheiten wird ein zu erwartender Zustand vorhergesagt (Vorhersage).

2. Mit aktuellen Sensordaten wird ein gemessener Zustand ermittelt. Der geschätzte Zustand wird zwischen dem gemessenen und dem vorhergesagten Zustand vermutet. Der Wert des geschätzten Zustands ergibt sich aus der Abwägung der Mess- gegenüber den Modellvorhersageunsicherheiten (Anpassung).

Diese Schritte wiederholen sich jeden Zeitschritt k. Zur besseren Lesbarkeit wird im Folgenden das tiefgestellte k ausgelassen.

Der zuvor beschriebene Ablauf ist auch unter der Bezeichnung *Alpha-Beta Filter* oder *g-h Filter* bekannt. Das Kalman Filter ist eine Spezialisierung des zuvor beschrieben Ablaufs in einer strengeren und formaleren Form.

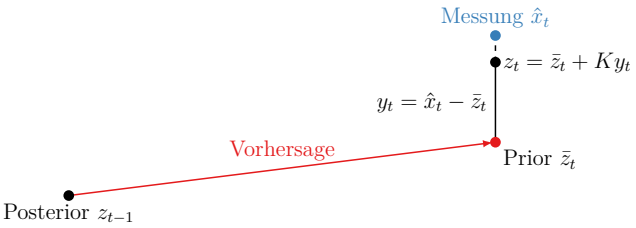


Abbildung 2.8: Schematische Darstellung der Ergebnisse der Schritte für ein univariantes Kalman Filter

2.5.2 Kalman Formulierung und Erweiterungen

Für die kompakte Formulierung des Kalman Filters wird die Annahme getroffen, dass das Messrauschen und somit das Rauschen des aus der Messung gewonnen Zustandsvorhersage und des Modellfehlers mit einer Gaußsche Normalverteilung $\mathcal{N}(\mu, \sigma^2)$ angenommen werden kann. Jede Gaußsche Normalverteilung kann vollständige durch den Mittelwert μ und Kovarianz σ^2 beschrieben werden. Diese starke Annahme einer Gaußsche Verteilung ist ein Hauptkritikpunkt an der konventionellen Kalman Filter Formulierung und wurde in Zuge von Weiterentwicklungen fallen gelassen.

Somit wird ein gesamt zu schätzenden Zustandsvektor probabilistisch durch die Kovarianzmatrix $\boldsymbol{R}$, die Zustandsvorhersage durch die Kovarianzmatrix $\boldsymbol{P}$ und der Modellfehler durch die Kovarianzmatrix $\boldsymbol{Q}$ beschrieben.

Wird das Systemmodel durch das Übertragungssystem

$$\begin{aligned} \bar{\boldsymbol{z}} &= \boldsymbol{F}\boldsymbol{z} + \boldsymbol{B}\boldsymbol{u} \\ \bar{\boldsymbol{y}} &= \boldsymbol{H}\boldsymbol{z} \end{aligned} \tag{2.18}$$

beschrieben, so kann eine Vorhersage $\bar{\boldsymbol{z}}$ (Prädiktion) durch Anwendung der Gleichung (2.18) mit dem vorherigen Zustand $\boldsymbol{z}$ (Posterior) und dem bekannten Eingangssignal $\boldsymbol{u}$ gebildet werden. Das Ergebnis $\bar{\boldsymbol{z}}$ wird als Prior bezeichnet. Die Matrix $\boldsymbol{F}$ beschreibt die Zustandsentwicklung und die Matrix $\boldsymbol{B}$ beschreibt den Einfluss von Stellgrößen auf das System. Die Matrix $\boldsymbol{H}$ liefert den messbaren Ausgang des Systems in Abhängigkeit des internen Zustands $\boldsymbol{z}$.

Die eigentliche Schätzung des Zustands (Modellanpassung) ist dreistufig:

1. Der Abstand $\boldsymbol{y}$ (Residual) zwischen der aktuellen Messung $\hat{\boldsymbol{x}}$ und der Vorhersage $\bar{\boldsymbol{z}}$ wird gebildet. Der Eingang $\boldsymbol{u}$ wird als bekannt angenommen.

2. Basierend auf den Kovarianzmatrix des Zustands $\boldsymbol{P}$ und der Messung $\boldsymbol{R}$, wird ein Faktor $\boldsymbol{K}$ bestimmt. $\boldsymbol{K}$ wird Kalman Verstärkung genannt. $\boldsymbol{K}$ bestimmt, ob bei der Zustandsschätzung die Vorhersage $\bar{\boldsymbol{z}}$ oder der gemessene Zustand $\hat{\boldsymbol{x}}$ einen stärkeren Anteil annimmt. Das Ergebnis bildet das Posterior für den nächsten Zeitschritt.

3. Mit der Kalman Verstärkung $\boldsymbol{K}$ werden sowohl der aktuelle Schätzwert $\boldsymbol{z}$ als auch die Kovarianz $\boldsymbol{P}$ für den nächsten Zeitschritt angepasst. $\boldsymbol{P}$ kann als ein Maß der Sicherheit der Zustandsschätzung ausgelegt werden.

Die Stufen sind in Abbildung 2.8 für eine univariante Schätzung nach (Labbe 2014) dargestellt. Bei einer univarianten Formulierung wird durch die Addition der aktuellen Kovarianz des Zustands P mit der Modellunsicherheit Q der Vorhersagefehler $\bar{P}$ mit berücksichtigt. Für die multivariante Formulierung sei auf Tabelle 2.5 verwiesen.

Um nichtlineare Modelle mit beliebigen Übertragungsfunktionen $\bar{\boldsymbol{z}} = \boldsymbol{f}(\boldsymbol{z})$ und $\bar{\boldsymbol{y}} = \boldsymbol{h}(\boldsymbol{z})$ behandeln zu können (im Gegensatz zu (2.18)), wurde das Kalman Filter um die Bildung von partiellen Ableitungen erweitert. Ein *Erweitertes Kalman Filter (EKF)* bildet in jedem Zeitschritt partielle Ableitungen der Übertragungsfunktionen

$$\boldsymbol{F} = \frac{\partial \boldsymbol{f}(\dot{\boldsymbol{z}}, \boldsymbol{u})}{\partial \boldsymbol{z}}\bigg|_{\boldsymbol{z}=\boldsymbol{z}_t, \boldsymbol{u}=\boldsymbol{u}_t} \quad \text{und} \tag{2.19}$$

$$\boldsymbol{H} = \frac{\partial \boldsymbol{h}(\bar{\boldsymbol{z}})}{\partial \bar{\boldsymbol{z}}}\bigg|_{\bar{\boldsymbol{z}}=\bar{\boldsymbol{z}}_t} . \tag{2.20}$$

Die Ableitungen können für jeden Zeitschritt numerisch angenähert oder analytisch gelöst werden.

Die formalen Beschreibungen sind in Tabelle 2.5 zusammengetragen. Hierbei sind die einzelnen Schritte des univarianten, multivarianten und erweiterten Kalman Filter nebeneinander gestellt.

Bei dem, in dieser Arbeit eingesetzten, erweiterten Kalman Filter zur Modellidentifikation IEKF werden analytische Funktionen $\boldsymbol{f}$ und $\boldsymbol{h}$ angesetzt und somit ist es möglich eine geschlossene analytische Lösung der jeweiligen Ableitung $\boldsymbol{F}$ und $\boldsymbol{H}$ zu Beginn der Modellbildung zu finden und dann jeweils in jedem Zeitschritt auszuwerten. Mit diesem festgelegten Ablauf des Kalman Filters reduziert sich der Entwurf des IEKF auf eine Festlegung der Funktionen $\boldsymbol{f}$ und $\boldsymbol{h}$, der Fehlerkovarianzmatrix $\boldsymbol{R}$ und der Modellunsicherheitsmatrix $\boldsymbol{Q}$.

Tabelle 2.5: Kalman Formulierungen

Schritt	Univariantes Kalman Filter	Multivariantes Kalman Filter	Erweitertes Kalman Filter
Vorhersage			$F = \frac{\partial \boldsymbol{f}(\boldsymbol{z},\boldsymbol{u})}{\partial \boldsymbol{z}}\Big\vert_{\boldsymbol{z}=\boldsymbol{z}_t,\boldsymbol{u}=\boldsymbol{u}_t}$
	$\bar{z} = z + \Delta z$	$\bar{\boldsymbol{z}} = \boldsymbol{F}\boldsymbol{z} + \boldsymbol{B}\boldsymbol{u}$	$\bar{\boldsymbol{z}} = \boldsymbol{f}(\boldsymbol{z},\boldsymbol{u})$
	$\bar{P} = P + Q$	$\bar{\boldsymbol{P}} = \boldsymbol{F}\boldsymbol{P}\boldsymbol{F}^\intercal + \boldsymbol{Q}$	$\bar{\boldsymbol{P}} = \boldsymbol{F}\boldsymbol{P}\boldsymbol{F}^\intercal + \boldsymbol{Q}$
Anpassung			$\boldsymbol{H} = \frac{\partial \boldsymbol{h}(\bar{\boldsymbol{z}})}{\partial \bar{\boldsymbol{z}}}\Big\vert_{\bar{\boldsymbol{z}}=\bar{\boldsymbol{z}}_t}$
	$y = \hat{x} - H\bar{z}$	$\boldsymbol{y} = \hat{\boldsymbol{x}} - \boldsymbol{H}\bar{\boldsymbol{z}}$	$\boldsymbol{y} = \hat{\boldsymbol{x}} - \boldsymbol{h}(\bar{\boldsymbol{z}})$
	$K = \frac{\bar{P}}{\bar{P}+R}$	$\boldsymbol{K} = \bar{\boldsymbol{P}}\boldsymbol{H}^\intercal(\boldsymbol{H}\bar{\boldsymbol{P}}\boldsymbol{H}^\intercal + \boldsymbol{R})^{-1}$	$\boldsymbol{K} = \bar{\boldsymbol{P}}\boldsymbol{H}^\intercal(\boldsymbol{H}\bar{\boldsymbol{P}}\boldsymbol{H}^\intercal + \boldsymbol{R})^{-1}$
	$z = \bar{z} + Ky$	$\boldsymbol{z} = \bar{\boldsymbol{z}} + \boldsymbol{K}\boldsymbol{y}$	$\boldsymbol{z} = \bar{\boldsymbol{z}} + \boldsymbol{K}\boldsymbol{y}$
	$P = (1-K)\bar{P}$	$\boldsymbol{P} = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\bar{\boldsymbol{P}}$	$\boldsymbol{P} = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\bar{\boldsymbol{P}}$

2.6 Konvexe Optimierung von linearen Matrixungleichungen

Als weitere mathematische Grundlage wird im folgenden Kapitel auf die Formulierung und die wichtigsten Eigenschaften von *linearen Matrixungleichungen* (engl.: linear matrix inequalities: LMI) sowie der Formulierung eines LMI-Optimierungsproblems eingegangen. Für den Entwurf der in Kapitel 3.2 vorgestellten Ausgangsrückführung ARF wird das zu lösende Problem aus einer Verkettung von LMIs formuliert. Diese Formulierung ist vorteilhaft, da sich Verkettung von LMI einfach realisieren lassen und LMI-Optimierungsprobleme eine spezielle Klasse an Optimierungsproblemen bilden, die zwar nur selten analytisch, jedoch sehr effektiv mittels numerischer Methoden gelöst werden können.

Eine lineare Matrizenungleichung kann allgemein in der Form

$$\boldsymbol{Y}(\boldsymbol{x}) = \boldsymbol{Y}_0 + \sum_{i=1}^{m} x_i \boldsymbol{Y}_i \succ 0 \tag{2.21}$$

dargestellt werden. Hierbei sind die Matrizen $\boldsymbol{Y}_0$ und $\boldsymbol{Y}_i$ konstant und symmetrisch. x_i beschreiben die Komponenten des gesuchten Vektors $\boldsymbol{x} \in \mathbb{R}^m$, der sogenannten LMI-Variable. Gesucht wird eine Lösung für den Vektor $\boldsymbol{x}$, sodass die aus linearen Gleichungen zusammengesetzte Matrix $\boldsymbol{Y}(\boldsymbol{x})$ positiv definit ist.

Die Notation $\boldsymbol{Y}_1 \succ 0$ bedeutet, dass $\boldsymbol{Y}_1$ positiv definit sei, und somit alle Eigenwerte der Matrix $\boldsymbol{Y}_1$ größer Null sind. In dieser Schreibweise bedeutet $\boldsymbol{Y}_1 \succ \boldsymbol{Y}_2$, dass $\boldsymbol{Y}_1 - \boldsymbol{Y}_2 \succ 0$ gilt. Negative Definitheit der Matrix $\boldsymbol{Y}_1$ wird analog mit $\boldsymbol{Y}_1 \prec 0$ dargestellt.

LMIs sind für die Verwendung in Optimierungsproblemen vorteilhaft, da die Menge der Lösungen $\boldsymbol{x} : \boldsymbol{Y}(\boldsymbol{x}) \succ 0$ konvex ist. Eine Menge $\mathcal{C}$ heißt konvex, wenn die Verbindungslinie zwischen zwei beliebigen Punkten aus $\mathcal{C}$ vollständig in $\mathcal{C}$ liegt. Diese Eigenschaft erlaubt einer numerischen Optimierung, vereinfacht betrachtet, den Verbindungslinien zwischen gültigen Lösungen zu folgen, um das globale Optimum zu finden, ohne die zulässige Lösungsmenge zu verlassen.

Mehrere LMIs können zusammengefasst werden. Beispielsweise entsprechen die LMIs

$$\boldsymbol{Y}_1 \succ 0\,,\ \boldsymbol{Y}_2 \succ 0 \tag{2.22}$$

der LMI mit der Blockdiagonalmatrix

$$\boldsymbol{Y}_3 = \begin{bmatrix} \boldsymbol{Y}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Y}_2 \end{bmatrix} \succ 0\ . \tag{2.23}$$

Da die Eigenwerte der Matrix $\boldsymbol{Y}_3$ mit den Eigenwerten der Matrizen $\boldsymbol{Y}_1$ und $\boldsymbol{Y}_2$ übereinstimmen, stimmt auch die Definitheit überein. Wegen der Zusammenfassbarkeit ist eine gut lesbare und erweiterbare Darstellung als Liste der einzelnen LMI-Bedingung möglich.

Werden sowohl die Nebenbedingungen in LMI-Form als auch die Gütefunktion $f(\boldsymbol{x}) = \boldsymbol{c}^\intercal \boldsymbol{x}$ linear dargestellt, liegt ein konvexes Optimierungsproblem vor. Dessen Lösbarkeit kann sicher nachgewiesen und dessen globales Optimum kann numerisch effizient bestimmt werden (Boyd et al. 2015). Der in der Informatik gängige Begriff für ein Minimierungsproblem der Form,

$$\min\ \ f(\boldsymbol{x})\,,\ \text{sodass}\ \ \boldsymbol{Y}(\boldsymbol{x}) \succ 0\ , \tag{2.24}$$

lautet *Semidefinites Problem* (SDP) und beschreibt eine Unterklasse der konvexen Optimierungsprobleme.

Es sei ergänzend erwähnt, dass das Maximierungsproblem,

$$\max\ \ -f(\boldsymbol{x})\,,\ \text{sodass}\ \ \boldsymbol{Y}(\boldsymbol{x}) \succ 0\ , \tag{2.25}$$

dem Minimierungsproblem (2.24) entspricht.

Wird lediglich eine gültige Lösung gesucht, so kann auf eine Gütefunktion verzichtet werden. In diesem Fall wird von einem Validierungsproblem gesprochen.

Es gibt keine einheitliche Darstellungsform eines Optimierungsproblems. Sogenannte *Interfaces* akzeptieren kanonische Darstellungen von Problemen und übernehmen die Umformung in die dem ausgewählten Solver entsprechende Form.

Das in dieser Arbeit eingesetzte Interface *cvxpy* (Diamond et al. 2016) übernimmt neben der zuvor erwähnten Umformung ebenfalls die Sicherstellung der Konvexität des gegebenen Problems. Das einzuhaltende Regelwerk der *disziplinierten konvexen Optimierung* (DCP) (Grant et al. 2006) definiert affine Operatoren, deren ausschließliche Anwendung die Konvexität der Lösung sicherstellen. Nicht affine Operatoren, welche sich durch geeignete Umformung affin ausdrücken lassen werden durch sogenannte *Atome* bereitgestellt. Zur Vertiefung der Eigenschaften von LMIs sowie Lemma und Hilfsätze, welche für die notwendigen Umformungen zur Erstellung von Atomen angewandt werden, sei auf das Standardwerk (Boyd et al. 2015) verwiesen.

2.7 Stabilitätsanalyse mittels direkter Methode von Ljapunov

Eine verbreitete Methode zum Stabilitätsnachweis von dynamischen Systemen ist die 1883 entwickelte *direkte Methode von Ljapunov*. Auf dieser Methode basierenden Stabilitätssätze sind praktisch in allen Lehrbüchern der *Nichtlinearen Regelung* zu finden, z. B. (Föllinger 1998) oder (Khalil 2002).

Anhand der Eigenschaften einer gesuchten Funktion $V(\boldsymbol{x}(t))$ eines dynamischen Systems der Form $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t))$ können die Stabilitätseigenschaften abgeleitet werden. Diese Methode wird in der Herleitung der Stabilität der dynamischen Ausgangsrückführung ARF in (Lens 2010) und in dieser Arbeit bei dem Stabilitätsnachweis der Aufgabenraumstruktur des Reglers mit vorgegebenem Performanzbands PPC eingesetzt. Dieses Kapitel soll einen Überblick über die *direkte Methode von Ljapunov*, bzw. den Stabilitätssatz von Ljapunov geben.

Betrachtet wird die Differenzialgleichung erster Ordnung des Zustandsvektors[4] $\boldsymbol{x}$ des dynamischen Systems der Form $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ mit der Ruhelage in $\boldsymbol{x} = \boldsymbol{0}$. Vorausgesetzt wird, dass für jeden Anfangswert innerhalb einer Umgebung $\mathcal{U}_1$ um den Ursprung eine stetige und eindeutige Lösung existiert.

[4] Zur besseren Lesbarkeit wird im Folgenden die Abhängigkeit des Zustandsvektors $\boldsymbol{x}$ von der Zeit t nicht explizit dargestellt.

Existiert eine Funktion $V(\boldsymbol{x})$ für das Vektorfeld $\boldsymbol{f}$ mit

$$V(\mathbf{0}) = 0\,, \tag{2.26}$$

welche in der Umgebung $\mathcal{U}_0 \subseteq \mathcal{U}_1$ stetig und stetig partiell differenzierbar ist und mit der Ausnahme von $\boldsymbol{x} = \mathbf{0}$ die Bedingungen,

$$V(\boldsymbol{x}) > 0 \tag{2.27}$$

$$\dot{V}(\boldsymbol{x}) \leq 0 \tag{2.28}$$

erfüllt, so ist die Ruhelage $\boldsymbol{x} = \mathbf{0}$ stabil im Sinne von Ljapunov.

Die Funktion $V(\boldsymbol{x})$ wird als Ljapunov Funktion (für das Vektorfeld $\boldsymbol{f}$) bezeichnet. Die Lösung der impliziten Gleichung $\dot{V}(\boldsymbol{x}) = 0$ umschließt ein Gebiet, welches für die stabile Ruhelage nicht verlassen wird. Gilt für (2.28) die strengere Bedingung $\dot{V}(\boldsymbol{x}) < 0$ so streben alle Trajektorien in diesem Gebiet in den Ursprung. Es wird in diesem Fall von *asymptotischer Stabilität* gesprochen.

Gelten die Bedingungen (2.27) und (2.28) für den gesamten Zustandsraum und zusätzlich

$$V(\boldsymbol{x}) \to \infty \text{ für } |\boldsymbol{x}| \to \infty\ , \tag{2.29}$$

so ist die Ruhelage *global stabil nach Ljapunov* (bzw. global asymptotisch stabil).

Die größte Herausforderung in dieser Stabilitätsanalyse ist das Auffinden einer geeigneten Funktion $V(\boldsymbol{x})$. Es ist im Allgemeinen nicht klar, ob solche Funktionen überhaupt existieren. Potentielle Funktionen werden auch als Ljapunov Funktionskandidaten bezeichnet.

3 Modellbildung und Reglerentwurf

In diesem Kapitel werden die Grundlagen der einzelnen Regelungsansätze dargestellt, weiterentwickelt und anhand von Simulationen untersucht. Zunächst wird auf die Modellbildung durch ein Kalman Filter eingegangen, welches die Grundlage zur modellbasierten Synthese der dynamischen Ausgangsrückführung (ARF) bildet. Daraufhin wird die Synthese und Untersuchung dieser ARF beschrieben. Zuletzt wird auf einen weiteren Regleransatz eingegangen, der ohne eine Modellannahme betrieben werden kann. Dieser Ansatz bietet sich für Situationen an, in denen eine experimentelle Modellbildung nicht möglich oder ungünstig ist. In dieser Arbeit wird der modellfreie Ansatz für die Regelung von Kontaktsituationen eingesetzt.

3.1 Automatische Modellidentifikation mittels erweiterten Kalman Filter

Im Folgenden wird auf den Entwurf und die Untersuchung mittels Simulationen eines erweiterten Kalman Filters zur Modellidentifikation eingegangen. Durch den festgelegten Ablauf eines erweiterten Kalman Filters reduziert sich der Entwurf auf die Auslegung des Zustandsübergangsmodell $\boldsymbol{f}$ und dem Sensormodell $\boldsymbol{h}$, sowie der Fehlerkovarianzmatrix $\boldsymbol{R}$ und der Modellunsicherheitsmatrix $\boldsymbol{Q}$. Nach der Auslegung des Kalman Filters sind lediglich die Modelldimension und die Initialwerte der Modellsystempole durch den Benutzer zu wählen.

Die Voraussetzung zur Reglersynthese eines modellbasierten Ansatzes ist die Existenz eines mathematischen Modells. Für die Klasse von Ausgangsrückführungen, welche in Kapitel 3.2 behandelt werden, ist ein mathematisches Modell in der Form von der Zustandsraumdarstellung notwendig. Der vorgestellte Ansatz eignet sich in diesem Fall, da die Schätzung ein lineares zeitinvariantes Modell ergibt.

Das zu schätzende Modell wird durch den Zustandsvektor $\boldsymbol{x}$, dem Steuerungsvektor $\boldsymbol{u}$ und des Ausgangsvektors $\boldsymbol{y}$ in den Matrizengleichungen

$$\begin{cases} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} \end{cases} \tag{3.1}$$

beschrieben.

Ziel der Modellidentifikation ist die Bestimmung der numerischen Werte der Matrizen $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ und $\boldsymbol{D}$. Die Durchgangssmatrix $\boldsymbol{D}$ wird oftmals als Nullmatrix angenommen oder in der Identifikation werden vernachlässigbar kleine Werte gefordert.

Für die Modellannäherung mit der linearen Struktur (3.1) wird ein erweitertes Kalman Filter nach der Beschreibung (Best et al. 2017) eingesetzt. Die zu schätzenden Parameter der Systemmatrizen werden in einem erweiterten Zustandsvektor $\boldsymbol{z}$ des Kalman Filters mit den Zuständen $\boldsymbol{x}$ zusammengefasst und kontinuierlich geschätzt.

Wird der Zustandsvektor eines Erweiterten Kalman Filters (EKF) durch die Modellparameter Θ erweitert, ergeben sich die Zustandsübergangsgleichungen für den identifizierenden erweiterten Kalman Filter (IEKF) zu jeden Zeitschritt k zu

$$\dot{\boldsymbol{x}}_k = \boldsymbol{f}(\dot{\boldsymbol{x}}_k, \boldsymbol{u}_k, \Theta_k) + \boldsymbol{w}_k \ , \tag{3.2}$$

$$\dot{\boldsymbol{y}}_k = \boldsymbol{h}(\dot{\boldsymbol{x}}_k, \boldsymbol{u}_k, \Theta_k) + \boldsymbol{v}_k \tag{3.3}$$

mit $\boldsymbol{w}$ als Modellierungsfehler und $\boldsymbol{v}$ als Sensormodellfehler und -rauschen.

Des Weiteren sind die Jacobi-Matrizen zu jedem Zeitpunkt durch

$$\boldsymbol{F}_k = \left.\frac{\partial \boldsymbol{f}(\dot{\boldsymbol{x}}, \boldsymbol{u}, \Theta)}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}_k, \boldsymbol{u}=\boldsymbol{u}_k, \Theta=\Theta_k} \quad \text{und} \tag{3.4}$$

$$\boldsymbol{H}_k = \left.\frac{\partial \boldsymbol{h}(\dot{\boldsymbol{x}}, \boldsymbol{u}, \Theta)}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}_k, \boldsymbol{u}=\boldsymbol{u}_k, \Theta=\Theta_k} \tag{3.5}$$

definiert.

Um die Anzahl der gesuchten Parameter in der Matrix $\boldsymbol{A}$ für das Modell (3.1) zu reduzieren, wird eine kanonische Normalform angenommen. Best et al. schlagen die modale Normalform vor, um den Größenbereich der gesuchten Parameter einzuschränken und somit Vorteile für die Konditionierung der Matrizen zu erhalten.

Die modale Normalform definiert die Matrix $\boldsymbol{A}$ durch die Systemeigenwerte auf der Blockhauptdiagonalen. Eine entsprechende Struktur eines beispielhaften Zustandsraumsystems fünfter Ordnung lautet

$$\begin{aligned} \boldsymbol{A} &= \begin{bmatrix} \sigma_1 & \omega_1 & 0 & 0 & 0 \\ -\omega_1 & \sigma_1 & 0 & 0 & 0 \\ 0 & 0 & \sigma_2 & \omega_2 & 0 \\ 0 & 0 & -\omega_2 & \sigma_2 & 0 \\ 0 & 0 & 0 & 0 & \sigma_3 \end{bmatrix}, \ \boldsymbol{B} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}, \\ \boldsymbol{C} &= \begin{bmatrix} c_1 & c_2 & c_3 & c_4 & c_5 \end{bmatrix}, \ \boldsymbol{D} = \begin{bmatrix} d_1 \end{bmatrix} . \end{aligned} \tag{3.6}$$

In diesem Beispiel bezeichnet σ_i den Realwert des $i$ten Eigenwertes und ω_i den entsprechenden Imaginärwert. Bei rein realwertigen Eigenwerten, wie bei σ_3, reduziert sich die Blockdiagonalstruktur zu einer Hauptdiagonalstruktur. Generell wird die Annahme konjugiert-komplexer Eigenwerte bevorzugt, da sich bei nicht schwingungsfähigen Systemen die Schätzung des Imaginärwertes ω dem Wert Null annähert und somit vollständig abgebildet wird. Die Matrizenstruktur kann ggf. entsprechend verändert und reduziert werden.

Zur weitere Vereinfachung müssen nicht alle Parameter der Matrix $\boldsymbol{B}$ und alle Parameter der Matrix $\boldsymbol{C}$ geschätzt werden. In der allgemeinen Struktur in beispielsweise (3.6) wird jede Eingangsvariable durch einen Parameter b_i und jede Ausgangsvariable durch einen Parameter c_i skaliert. Somit kann jeweils einer der sich erweiternden Parameter frei gewählt werden. Da im Allgemeinen nicht jeder Eingang alle Zustandsvariablen anregen wird, müssen einige der Parameter b_i zu Null konvergieren können. Daher werden die Parameter der Matrix $\boldsymbol{C}$ frei gewählt. Für den Spezialfall (3.6) einer jeweils einzelnen skalaren Eingangs- und Ausgangsvariablen (engl.: single-input single-output: SISO), kann die konstante Matrix $\boldsymbol{C} = \begin{bmatrix} 1 & 1 & 1 & \dots \end{bmatrix}$ gewählt werden. Für den allgemeinen Fall können ebenfalls Variablenabhängigkeiten definiert und somit der Suchraum verkleinert werden. Hierfür sei auf (Best et al. 2017) verwiesen.

Der erweiterte Zustandsvektor des IEKF wird durch Zusammenfassen der eigentlichen Zustandsvariablen des Vektors $\boldsymbol{x}$ mit allen Modellparametern Θ gebildet. Für die zuvor angeführte Struktur (3.6) ergibt sich der Zustandsvektor der Länge m ($m = 16$ für (3.6)) zu

$$\begin{aligned} \boldsymbol{z} = [&x_1 \quad x_2 \quad \cdots \quad x_5 \quad \sigma_1 \quad \omega_1 \quad \sigma_2 \quad \omega_2 \quad \sigma_3 \quad \cdots \\ &b_1 \quad b_2 \quad \cdots \quad b_5 \quad c_1 \quad c_2 \quad \cdots \quad c_5 \quad d_1]^\intercal . \end{aligned} \tag{3.7}$$

Das Zustandsübergangsmodell des Kalman Filters $\boldsymbol{f}(\boldsymbol{z})$ aus Gleichung (3.2) existiert nur für die eigentlichen Modellzustände $\boldsymbol{x}$ während für die Modellparameter Θ die erwartete zeitliche Entwicklung zu Null gesetzt wird.

Somit ergibt sich für das Beispiel (3.6) die Gleichungen (3.2) und (3.3) zu

$$\boldsymbol{f}(\boldsymbol{z}) = \begin{cases} \dot{\boldsymbol{z}}_{1...5} = \boldsymbol{f}(z_{1...5}) = \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \\ \dot{\boldsymbol{z}}_{6...16} = \boldsymbol{f}(z_{6...16}) = \boldsymbol{f}(\Theta) = \boldsymbol{0} \end{cases} , \tag{3.8}$$

$$\boldsymbol{h}(\boldsymbol{z}) = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} \ . \tag{3.9}$$

Die Gleichungen (3.4) und (3.5) folgen durch die Bildung der Jacobi-Matrizen, d. h. der vollständigen Ableitungen der Gleichungen (3.8) und (3.9) nach dem Zustandsvektor $\boldsymbol{z}$ zu

$$\boldsymbol{F} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}} \ , \tag{3.10}$$

$$\boldsymbol{H} = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}} \ . \tag{3.11}$$

Best et al. empfehlen für die Bildung der Jacobi-Matrizen ein Computeralgebrasystem wie in MATLAB oder Maple. In der hier angeführten Implementierung wird die symbolische Umgebung Sympy (Meurer et al. 2017) zum Einsatz kommen. Die daraus folgenden Softwaretechnischen Vorteile, die zu modellflexiblen Schätzern führen werden näher in Kapitel 3.1.1 beschrieben.

Die Fehlerkovarianzmatrix $\boldsymbol{R} = \boldsymbol{I}$ wird mit einer Einheitsmatrizen initialisiert. Für die Modellunsicherheitsmatrix $\boldsymbol{Q}$ wird eine modifizierte $(m \times m)$ Diagonalmatrix angesetzt. Bei den Einträgen der eigentlichen Zustände $\boldsymbol{x}$ und den gewählten Modellparametern c_i werden die Werte der Diagonaleinträge zu Null gewählt. Bei allen anderen Einträgen wird ein Wert $\rho < 1$ gewählt. Mit dieser Wahl der Einträge werden die durch das Modell vorhergesagten Zustände $\boldsymbol{x}$ als fehlerfrei deklariert, was dazu führt, dass die Zustandsübergänge ausschließlich aus dem (geschätzten) Modell gefolgert werden. Ebenso wird eine Anpassung der gewählten Parameter c_i verhindert.

Für das Beispiel (3.6) ergibt sich

$$\boldsymbol{Q} = \begin{bmatrix} 0 & & & & & & & & & & \\ & \ddots & & & & & & & & & \\ & & 0 & & & & & & & & \\ & & & \rho & & & & & & & \\ & & & & \ddots & & & & & & \\ & & & & & \rho & & & & & \\ & & & & & & 0 & & & & \\ & & & & & & & \ddots & & & \\ & & & & & & & & 0 & & \\ & & & & & & & & & \rho & \end{bmatrix}, \quad \boldsymbol{z} = \begin{bmatrix} x_1 \\ \vdots \\ x_5 \\ \sigma_1 \\ \vdots \\ b_5 \\ c_1 \\ \vdots \\ c_5 \\ d_1 \end{bmatrix} . \tag{3.12}$$

Der einzige Einstellungsparameter ρ beschreibt den erwarteten mittleren quadratischen Fehler der zu schätzenden Parameter. Ein höherer Wert für ρ führt zu einer schnelleren Konvergenz, da das IEKF stärker auf die Messungen reagiert. Allerdings können verrauschte Messungen zu der Annahme eines instabilen Modells führen. Bei einem instabilen Modell wachsen die Zustandswerte im Vorhersageschritt exponentiell an. Steigt weiter der Abstand zwischen Messung und Vorhersage exponentiell, da die Messwerte sich nicht entsprechend der Vorhersage verhalten, so wird die Modellannahme so stark korrigiert, dass sie praktisch verworfen wird und das Filter nicht konvergiert.

Das IEKF berechnet zu jedem Zeitschritt k den erwarteten Modellausgang

$$y_k = \boldsymbol{C}_{k-1}\boldsymbol{x}_k + \boldsymbol{D}_{k-1}\boldsymbol{u}_k \ , \tag{3.13}$$

den Anpassungsschritt der Kalman Verstärkung

$$\boldsymbol{K}_k = \boldsymbol{P}_k\boldsymbol{H}_k^\intercal[\boldsymbol{H}_k\boldsymbol{P}_k\boldsymbol{H}_k^\intercal + \boldsymbol{R}]^{-1} \tag{3.14}$$

und die Anpassung der Fehlerkovarianzmatrix

$$\boldsymbol{P}_k^\star = [\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k]\boldsymbol{P}_k \ . \tag{3.15}$$

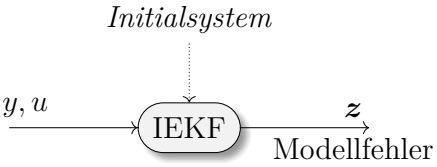


Abbildung 3.1: Darstellung eines IEKF ROS Knotens. Nachdem das IEKF mit einer Zustandsraumstruktur und Anfangswerten initialisiert wurde, werden die empfangenen Soll- und Istgrößen u, y zur Modellierung genutzt. Nach jeder Aktualisierung werden die internen Zustände $\boldsymbol{z}$ verschickt.

Die numerische Integration des Systems mit der Zeitkonstanten T, sog. Euler Integration, wird je mit dem Anpassungsschritt der Modellparameter zu

$$\hat{\boldsymbol{z}}_{k+1} = \hat{\boldsymbol{z}}_k + T \cdot \boldsymbol{f}_k + \boldsymbol{K}_k(y_k - \boldsymbol{h}_k) \ , \tag{3.16}$$

$$\boldsymbol{P}_{k+1} = \boldsymbol{P}_k^\star + T \cdot [\boldsymbol{F}_k \boldsymbol{P}_k^\star + \boldsymbol{P}_k^\star \boldsymbol{F}_k^\intercal + \boldsymbol{Q}] \tag{3.17}$$

zusammengefasst und aus implementationstechnischen Gründen gegenüber dem in Tabelle 2.5 dargestellten Ablauf in der Reihenfolge verschoben.

Der Anpassungsschritt wird durch den Fehler der Ausgangsschätzung getrieben. Aufgrund der Euler Integration ist es notwendig, dass die Zeitkonstante T so klein gewählt wird, dass das dynamische Verhalten des zu schätzenden Systems erfassen werden kann. Die Zeitkonstante ist praktisch durch die Abtastzeit der Robotersteuerung (typischerweise zwischen 2 ms und 8 ms) nach unten begrenzt.

3.1.1 Implementierung

Die Implementierung des zuvor beschriebenen IEKF wurde vollständig in der Programmiersprache Python mit Anbindung an ROS vorgenommen. Der IEKF Knoten (im Folgenden IEKF genannt) existiert eigenständig und kann unbegrenzt oft parallel betrieben werden. Jedes IEKF kann individuell mit Anfangswerten der Systemeigenwerte und der damit verbundenen Zustandsraumdimension konfiguriert werden. Ebenso empfängt jeder IEKF die entsprechend zugeordnete Soll- und Istgröße u, y und verschickt zu jeden Zeitschritt den aktuellen internen Zustand $\boldsymbol{z}$ und den Modellfehler. Abbildung 3.1 zeigt eine schematische Abbildung eines IEKF Knotens.

Die symbolische Konfiguration der Systemmatrizen $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ und $\boldsymbol{D}$ sowie die Zusammensetzung des Zustandsvektors $\boldsymbol{z}$ liegen in der Implementierung parallel zu den numerischen Werten

vor. Hierfür wird eine Kombination der Python Bibliotheken `sympy` (Meurer et al. 2017) und `numpy` (Oliphant 2006) genutzt. `Sympy` ermöglicht eine vollständig symbolische Repräsentation der Struktur der Matrizen und Vektoren mit Computeralgebra, sowie die symbolische Bildung der Jacobi-Matrizen.

Da nach einer initialen Konfiguration die symbolische Repräsentation nur noch informellen Charakter einnimmt, wird während der Laufzeit ausschließlich mit numerischen Matrizen und Vektoren gearbeitet. Der komplette algorithmische Ablauf des IEKF ist im Anhang A.3 in Algorithm 2 aufgeführt.

Da in der folgenden Reglersynthese von einer Durchgangssmatrix $\boldsymbol{D} = \mathbf{0}$ ausgegangen wird, wurden die entsprechenden Parameter in $\boldsymbol{z}$ und $\boldsymbol{Q}$ zu Null festgesetzt. Damit entfällt zwar die Möglichkeit die Konvergenz des IEKF an dem Betrag von $\boldsymbol{D}$ abzuschätzen, dafür kann das Modell direkt zur Reglersynthese herangezogen werden.

Um das IEKF mit einem zuvor angenäherten Systemmodell zu initialisieren muss sichergestellt werden, dass die jeweiligen symbolischen Repräsentationen übereinstimmen. Probleme können entstehen, wenn von unterschiedlicher Sortierung der Eigenwerte ausgegangen wird, insbesondere bei der Reihenfolge von komplexen und reellwertigen Eigenwerten. In dieser Arbeit wird daher zunächst die Eigenwerte des Initialsystems berechnet und nach aufsteigenden Realteil und ggf. zusätzlich nach aufsteigenden Betrag des Imaginärteils sortiert. Mit der sortierten Liste der Eigenwerte wird eine eindeutige kanonische Darstellung errechnet die zur symbolischen und entsprechend numerischen Repräsentation dient.

Da die ROS-Kommunikation mit unvorhersehbarer Verzögerungszeit behaftet ist, werden die jeweiligen Zeitschritte T (Vgl. (3.17) und (3.16)) über Zeitstempel bei der Datenversendung ermittelt. Ergeben sich zu große Zeitschritte (bspw. mehr als 10 Takte), werden die internen Zustände $x = \mathbf{0}$ zurückgesetzt und die Übertragung der Modellanpassung wird durch Setzen von $T = 0$ ausgesetzt. Dieser Rücksetzvorgang wird systematisch jeweils zum Eintritt in den beobachteten Ablauf ausgelöst.

Es ist möglich, dass kurzzeitig ein instabiler Zustand identifiziert wird. Die Wahrscheinlichkeit wird durch die Wahl eines großen Parameters ρ und durch große Zeitschritte T erhöht. In einem instabilen Zustand streben die Werte des Zustandsvektors $\boldsymbol{z} \to \pm\infty$, der Matrix $\boldsymbol{P}_k \to \pm\infty$ und $\boldsymbol{H} \to 1/\infty \mathrel{\widehat{=}} \texttt{NaN}$. Mit einer numerischen Begrenzung ist das IEKF in der Lage, den instabilen Zustand zu überwinden und zu einem stabilen Modell zurückzukehren. Allerdings bewirkt die heftige Rückkehrreaktion, dass das geschätzte Modell faktisch zurückgesetzt wird und im weiteren Verlauf erneut angenähert werden muss.

3.1.2 Initialisierung

In dem hier beschriebenen Ansatz muss nicht von gänzlich unbekannten Systemen ausgegangen werden, die identifiziert werden sollen. Vielmehr soll das IEKF für spezifische Situationen während eines Roboterprogrammablaufs ein grundlegendes dynamisches Modell anpassen, um so auf den Übertrag der Aufgabe in ein neues Umfeld reagieren zu können. Durch Einschätzung, ob sich ein zugrundeliegendes Systemmodell anders als angenommen verhält, können auch geänderte äußere Umstände einer kontinuierlich betriebenen Anlage erkannt werden.

Durch eine geeignete Wahl der initialen Annahmen kann schneller Konvergenz erreicht werden. Im Folgenden wird ein Ansatz beschrieben, um ein initiales Systemmodell mit festgelegter Struktur auf Basis einer einzelnen Sprungantwort anzunähern.

Das dynamische Modell wird als Verzögerungsglied zweiter Ordnung (PT2) mit signifikanter Verzögerung t_d angenommen. Für eine nieder dimensionale Annäherung wird ein PT2-Glied mit einer Padé-Approximation (Vajta 2000) erster Ordnung

$$G(s) = \frac{K}{T^2 s^2 + 2dTs + 1} \cdot \frac{\frac{t_d}{2}s + 1}{-\frac{t_d}{2}s + 1} \tag{3.18}$$

angesetzt, wobei K den Verstärkungsfaktor, d die Dämpfung und T die Zeitkonstante bezeichnet. Als Richtwert der Verzögerung wird von 3-4 Taktzyklen ausgegangen, was für die Kommunikationsrate von 125 Hz einer ungefähren Verzögerung von $t_d = 0.3\,\mathrm{s}$ entspricht.

Basierend auf einer repräsentativen Sprungantwort können die Parameter dieser Modellannahme bestimmt werden. Hierfür muss ein entsprechendes Optimierungsproblem gelöst werden. Da das IEKF intern eine ideale modale Zustandsraumrepräsentation nutzt, wird das Optimierungsproblem um die Bedingung erweitert, sodass eine numerisch stabile modale Transformation gefunden werden kann.

Genauer gesagt wird zunächst das Modell aus (3.18) numerisch in eine modale Form transformiert (Vgl. A.1). Die sich ergebene Matrix $\boldsymbol{A}$ wird von den numerisch ungenauen Werten außerhalb der Hauptdiagonalen bereinigt. Eine Sprungantwort des modalen Systems wird berechnet, die Zeitachsen der Messung und der Sprungantwort synchronisiert und die (zeit-)punktweise Differenz gebildet.

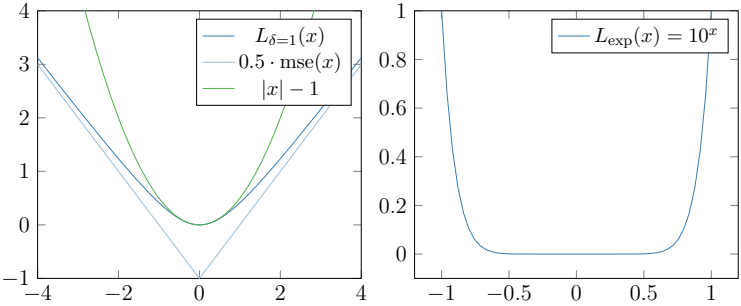


Abbildung 3.2: Gewichtungsfunktionen

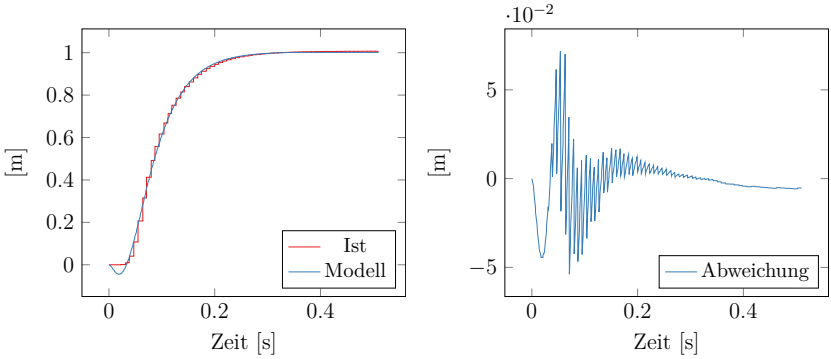


Abbildung 3.3: Angenäherte Funktion als Ergebnis der Modellierung aus einer einzelnen Sprungantwort

Um das typische Unterschwingen der Padé-Approximation erster Ordnung gegenüber den Differenzen während des weiteren Verlaufs nicht übermäßig zu priorisieren, wird die Differenz mit der Pseudo-Huber Funktion,

$$L_\delta(a) = \delta^2 \left(\sqrt{1 + (a/\delta)^2} + 1 \right) \quad , \tag{3.19}$$

gewichtet.

Die Pseudo-Huber Funktion ist eine glatte Annäherung an die stückweise definierte Huber Funktion. Diese vereinigt die moderate Gewichtung von kleinen Abweichungen einer quadratischen Wichtung mit der starken Wichtung großer Abweichungen eines linearen Betrages. Durch den Parameter δ kann das Übergangsverhalten angepasst werden. Abbildung 3.2 illustriert die Pseudo-Huber Funktion mit dem Parameter $\delta = 1$.

Des Weiteren soll der (negative) Betrag der Pole des Systems eingeschränkt werden, um weiteren numerischen Problemen vorzubeugen. Hierfür werden die Pole des Systems bestimmt. Sind diese betragsmäßig größer als ein anzugebendes Limit, so wird der Wert des Poles ebenfalls gewichtet auf die Gütefunktion addiert. Die hierfür gewählte Gewichtungsfunktion (Abbildung 3.2) ist eine einfach exponentielle Funktion hoher Ordnung

$$L_{\exp}(a) = 10^a \ . \tag{3.20}$$

Die Gütefunktion ergibt sich aus der kumulativen Summe der durch die Pseudo-Huber Funktion gewichteten Abstände und den gewichteten Betrag der Systempole, welche das Limit überschreiten.

Die so formulierte Gütefunktion kann durch einen Standard Solver wie bspw. *Powell* oder *Nelder-Mead* minimiert werden. Powell zeigt bei den untersuchten Sprungantworten etwas höhere Robustheit. Eine so angenäherte Funktion ist in Abbildung 3.3 dargestellt.

3.1.3 Simulationsergebnisse

Das vorgeschlagene Kalman Filter wird zunächst in einer Simulation überprüft. In der Simulation ist es möglich, Systemparameter so zu manipulieren, dass das Verhalten des Filters in relevanten Situationen eingeschätzt werden kann. Die untersuchten Situationen umfassen, (i) verschiedene Abtastzeiten zwischen 8 ms und 2 ms, (ii) der Umgang mit Verzugszeiten, (iii) dem Einfluss von Messrauschen.

Das in der Simulation genutzte System entspricht einem Verzögerungsglied zweiter Ordnung (PT2),

$$G_0(s) = \frac{K}{T^2 s^2 + 2dTs + 1} \tag{3.21}$$

mit

$$K = 1.0041837\,,\ T = 1.1233985\,,\ d = 0.0300641 \tag{3.22}$$

und einem Zeitverzug $t_d = 0.033$. Die Parameter wurden zuvor mittels Annäherung einer Spungantwort gefunden (Vgl. Kapitel 3.1.2). Der Zeitverzug wird mit einer Padé-Approximation dritter Ordnung angenähert.

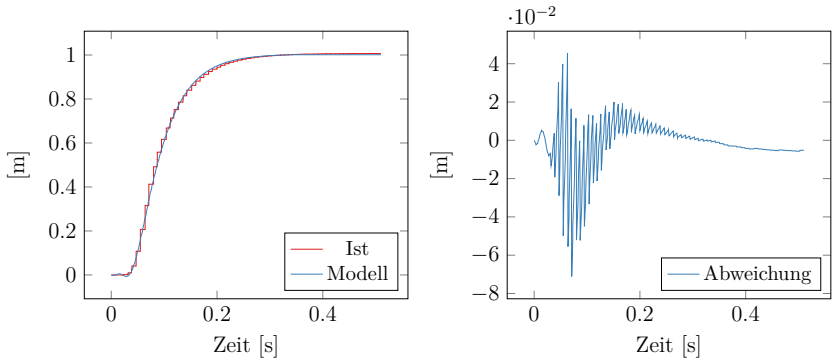


Abbildung 3.4: Sprungantwort des simulierten Modells und der Abstand zu dem gemessenen Roboterverhalten

Damit ergibt sich ein zu identifizierendes Zustandsraumsystem nach Gleichung (3.1) gerundet zu,

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & 3.67393746 \\ 100 & 0 & 0 & 0 & 30.8899128 \\ 0 & -100 & 0 & 0 & -78.2460267 \\ 0 & 0 & -100 & 0 & 83.1248282 \\ 0 & 0 & 0 & -1000 & -437.697553 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 36.89308114 \\ -60.98634759 \\ -40.32555132 \\ -11.11006912 \\ 0 \end{bmatrix} \tag{3.23}$$

$$\boldsymbol{C} = \begin{bmatrix} 0 & 0 & 0 & 0 & -0.1 \end{bmatrix}, \quad \boldsymbol{D} = \begin{bmatrix} 0 \end{bmatrix} . \tag{3.24}$$

In Abbildung 3.4 ist das zugrundeliegende Modell gegen die gemessene Sprungantwort aufgetragen. Zur Verdeutlichung der Güte ist auch der Abstand zwischen der Messung und dem Modell gezeigt.

Als Anfangswerte der Systempole des IEKF wurden die Pole,

$$\begin{bmatrix} \sigma_1 & \omega_1 & \sigma_2 \end{bmatrix} = \begin{bmatrix} -2 & 3 & -2 \end{bmatrix} \tag{3.25}$$

festgelegt. Dabei ist lediglich die Form der Pole gewählt, d. h. ein konjugiert-komplexes Polpaar und ein reelwertiger Pol. Die numerischen Werte wurden willkürlich in Hinblick auf eine gute Illustration gewählt.

Alle weiteren Modellparameter, $\boldsymbol{B}, \boldsymbol{C}$ und $\boldsymbol{D}$ werden mit zufälligen Werten initialisiert. Die initialen Zustandsvariablen $\boldsymbol{x}$ werden mit $\mathbf{0}$ angenommen. Der Einstellungsparameter wird konstant zu $\rho = 0.5$ gewählt.

Die Aufgabe des IEKF in dieser Simulation ist somit das Finden eines Modells dritter Ordnung, welches das System fünfter Ordnung annähert. Eine perfekte Annäherung ist wegen der kleineren Dimension des Modells nicht zu erwarten.

Die Simulation wird nach der Initialisierung über einen Zeitraum von 10 min durchgeführt. Mit einer Frequenz von 1000 Hz wird der nächste Zustand des zu identifizierenden Systems berechnet. Die internen Zustandsvariablen sind dem IEKF unbekannt. Zu jedem Takt des IEKF wird die Ausgangsgröße weitergereicht und Vorhersage- sowie Anpassungsschritt des IEKF durchgeführt.

In jeweils zufälligen Zeitabständen werden außerdem zufällige Führungsgrößensprünge auf das System gegeben. Sowohl die zufällige Länge der Sprungantworten als auch die zufällige Sprunghöhe sollen das zugrundeliegende System in möglichst allen relevanten Modi anregen.

Die Berechnungszeit eines Simulationsdurchlaufes (simulierte 10 min) dauert ca. 2 min Rechenzeit. Für die Anwendung bedeutet das in Anbetracht der nötigen Rechenleistung, dass mehrere IEKF Instanzen gleichzeitig betrieben werden können, um verschiedene Modellannahmen anzunähern und gegeneinander zu vergleichen.

In Abbildung 3.5 sind die Verläufe der Pole der Modellschätzung für verschiedene Laufzeit der simulierten Zeit bei einer Abtastrate von 125 Hz dargestellt. In der Simulation werden ca. 40 Sprünge pro Minute durchgeführt. Jedes Sprungexperiment enthält neue Informationen, die durch das IEKF verwendet wird. Die Modellschätzung wird besser, je mehr Informationen verwendet werden können. Nach ca. 10 min (495 Sprungexperimenten) zeichnet sich eine gewisse Konvergenz ab.

In Tabelle 3.1 sind die Verläufe der Pole der Modellschätzung und die Sprungantworten der Ergebnisse nach Ablauf einer Simulation von 10 min für verschiedene Abtastraten dargestellt.

Es ist zu erkennen, dass der Verlauf der Sprungantworten gut dem Verlauf des zugrundeliegenden Systems ähnelt. Weiter ergibt sich ein Unterschwingen vor dem eigentlichen Anstieg bei ca. $t < 0.3\,\mathrm{s}$. Dieser Verlauf ist typisch für eine Padé-Annäherung erster Ordnung. Es sei an dieser Stelle nochmals betont, dass dem IEKF lediglich die Dimension vorgegeben wurde und keine Struktur. Somit ist auch die Herausbildung einer “Padé-Struktur” ein Ergebnis des IEKF selbst.

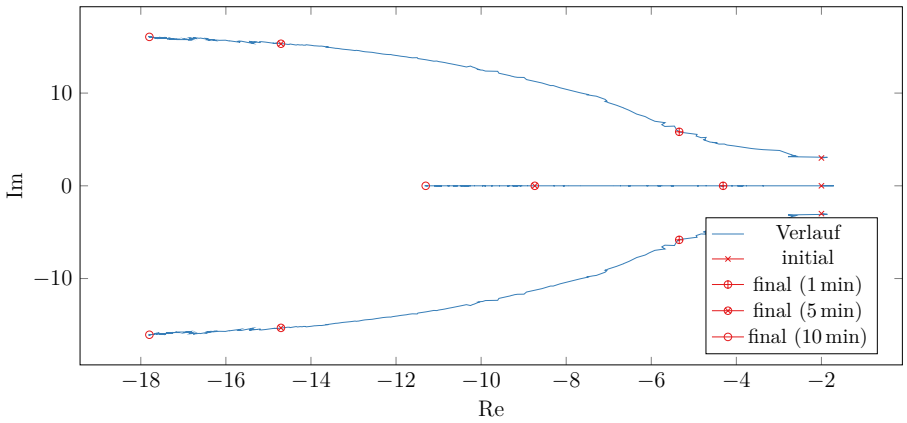


Abbildung 3.5: Polverläufe der Modellschätzung für verschiedene Laufzeiten

Des Weiteren ist zu erkennen, dass die Qualität der Annäherung mit geringerer Abtastrate abnimmt. Das ist nicht weiter überraschend, da das IEKF weniger Iterationen zur Adaptation nutzen kann. Außerdem können mit niedrigerer Frequenz, hochfrequente Modi des zu identifizierenden Systems nicht erfasst werden. Trotzdem kann auch mit der vergleichsweise niedrigen Abtastrate von 125 Hz eine gute Annäherung erreicht werden.

In Tabelle 3.2 sind die Verläufe der Pole der Modellschätzung und die jeweilige Ausgangsschätzung des IEKF aufgetragen. Auf den Ausgang des zuvor beschriebenen simulierten Modells wurde ein additives Gaußsches Rauschen $\mathcal{N}(\mu = 0, \sigma^2)$ mit verschiedenen Standardabweichungen σ aufgebracht. Die Grafiken zeigen den Verlauf der ersten 30 s mit einer Abtastrate des IEKF von 125 Hz. Das Filter zeigt sich tolerant gegenüber dem Rauschen. Selbst dem Verlauf des nahezu unkenntlichen Ausgangssignals bei $\sigma = 0.5$ wird gut gefolgt. Weiter ist anhand dieses Extremfalls zu beobachteten, dass die Pole des identifizieren Systems mit zunehmender Konvergenz in einer lokalen Umgebung bleiben und dort einer gewissen Schwingung unterliegen. Diese Ergebnisse waren zu erwarten, da das Kalman Filter auf der Annahme einer Normalverteilung der Messwerte beruht.

Es bleibt allerdings die Frage offen, wie die Konvergenz des Filters gemessen und bewertet werden kann. Best et al. schlagen die Spur der $n \times n$ Matrix $\boldsymbol{P}_k$

$$\text{spur}(\boldsymbol{P}_k) = \sum_{i=1}^{n} p_{ii}$$

Tabelle 3.1: Polverläufe und Resultat der Modellschätzung für verschiedene Abtastraten

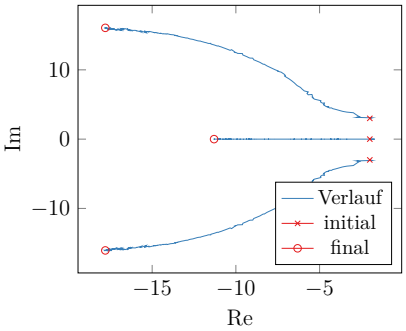
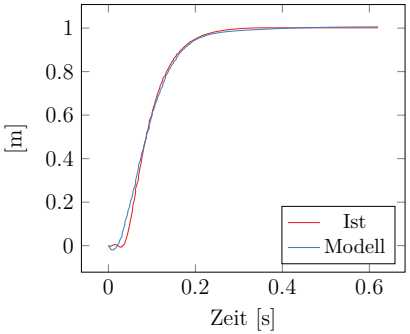
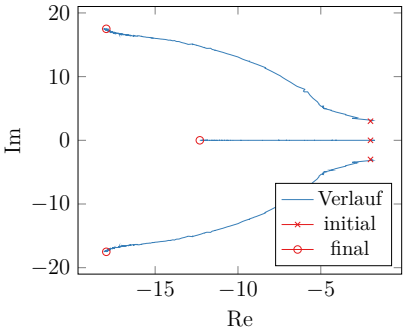
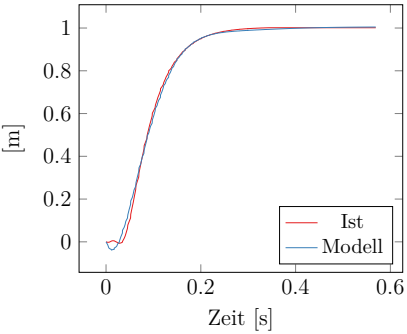
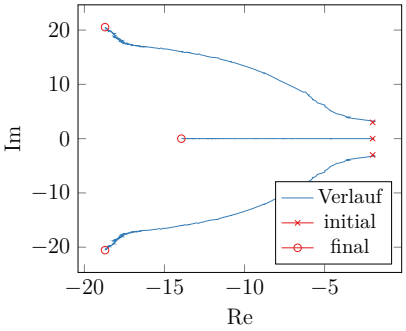
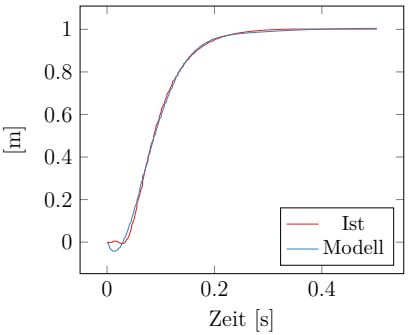
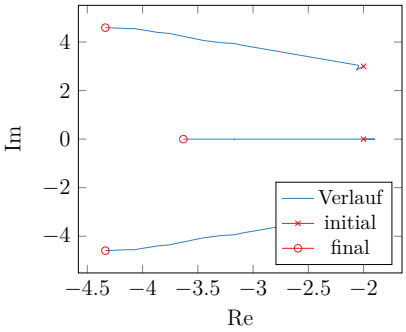
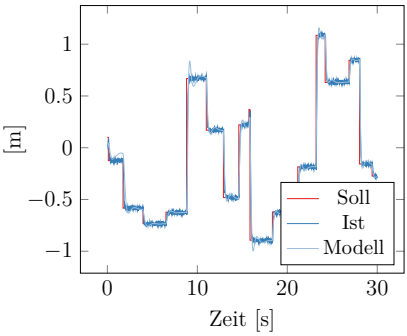
125 Hz	
	
250 Hz	
	
500 Hz	
	

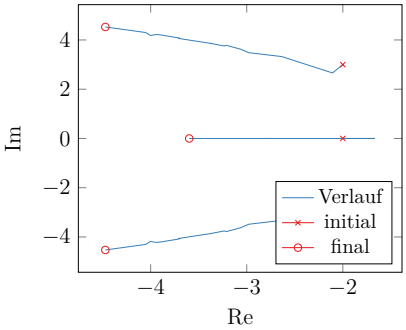
Tabelle 3.2: Polverläufe und Resultat der Modellschätzung für verschiedenes Messrauschen

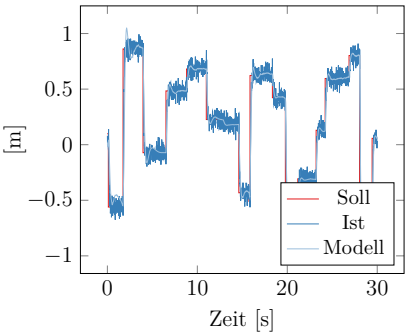
$\sigma = 0.01$



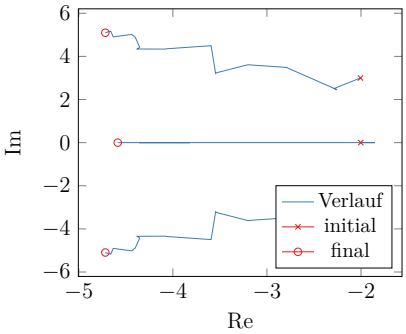


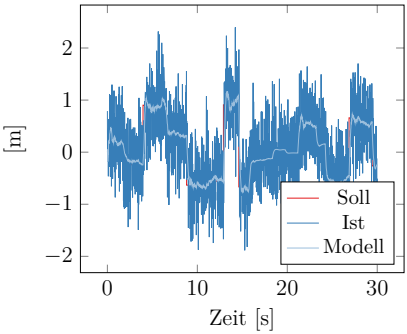
$\sigma = 0.05$





$\sigma = 0.5$





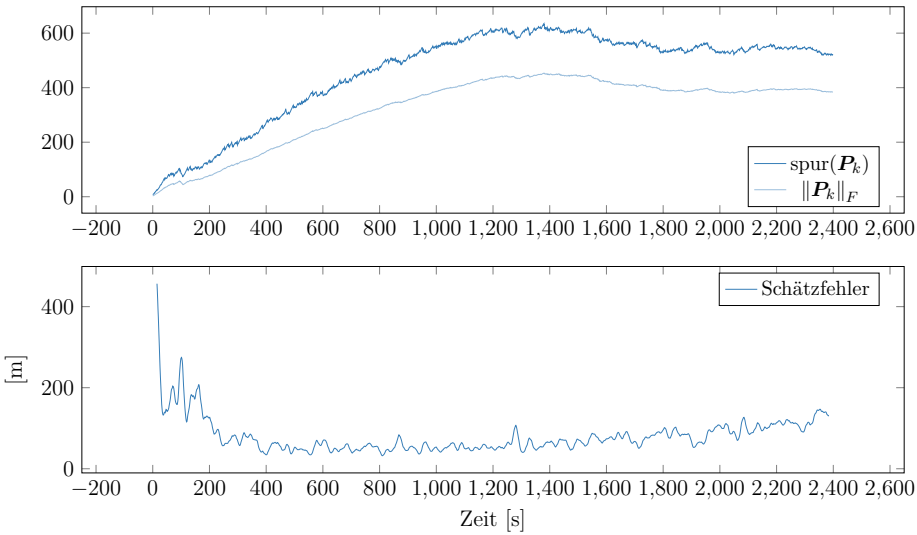


Abbildung 3.6: Verläufe von Konvergenzmaßen

vor. Einen ähnlichen Verlauf, allerdings reduzierten Einfluss des aktuellen Experimentes zeigt die Frobeniusnorm

$$\|\boldsymbol{P}_k\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} |p_{ij}|^2} \ ,$$

welche als die Wurzel aus der Summe der Betragsquadrate aller Matrixelemente p_{ij} definiert ist. Die Frobeniusnorm definiert somit eine Euklidische Länge der Matrix $\boldsymbol{P}_k$.

Die Fehlerkovarianzmatrix $\boldsymbol{P}_k$ und somit deren Normen drücken qualitativ den Informationsgehalt der aktuellen Daten und damit deren qualitativen "Nutzen" zur Schätzung aus.

Ein weiteres Maß der Konvergenz kann über den Schätzfehler, d. h. den Abstand des gemessenen Ausgangs und der entsprechenden Schätzung, angegeben werden. Eine gewichtete Summe über die zuletzt vergangenen Iterationen wird mit einem gleitenden Fenster (engl. Finite Impulse Response: FIR) Filter realisiert.

In Abbildung 3.6 sind die Verläufe der Spur und der Frobeniusnorm neben dem gefilterten Schätzungsfehler dargestellt. Als gleitendes Filter wurde ein Bartlett Fenster mit der Länge von 30000 Werten (entspricht 30 s) gewählt. Die gesamte Simulation betrachtete 40 min Simulationszeit mit der Abtastrate des IEKF von 125 Hz. Der Schätzfehler kann nur herangezogen werden, wenn die internen Zustandsvariablen $\boldsymbol{x}$ mitgeschätzt werden.

Es ist zu erkennen, dass sowohl die Spur als auch die Frobeniusnorm sich einem Grenzwert annähern (für $t > 1500\,\mathrm{s}$). Allerdings kann über dem Verlauf des Schätzungsfehlers die Konvergenz bedeutend früher ($t > 500\,\mathrm{s}$) angenommen werden. Außerdem kann ungefähr ab dem Zeitpunkt $t = 2300\,\mathrm{s}$ eine fallende Tendenz in der Spur und der Frobeniusnorm erkannt werden. Der Schätzfehler tendiert allerdings zu einem höheren Wert. Das IEKF bestätigt somit ein höheres Vertrauen in die geschätzten Werte obwohl der Schätzfehler sich erhöht hat.

Über einen längeren zeitlichen Verlauf können entsprechende Schwingungen sich wiederholen. Daher werden an dieser Stelle beide Konvergenzmaße herangezogen. Wird das Modell mit dem global über dem Verlauf betrachteten geringsten Schätzfehler bewahrt, so kann das Risiko eines ungünstigen Zeitpunktes zu Auswahl des geschätzten Modells entgegengewirkt werden. Durch die Betrachtung der Frobeniusnorm kann zusätzlich sichergestellt werden, dass Schätzungsfehler während Bewegungen gewichtet werden und nicht das Fenster nur den gleichbleibenden stationären Fehler annähert.

Zur Illustration des Verhaltens wurde in Abbildung 3.7 die Schätzung des IEKF fortgeführt, obwohl keine Sprungexperimente bei $t > 12\,\mathrm{s}$ mehr ausgeführt werden. Es ist zu erkennen, dass der gleitende Schätzfehler gegen Null strebt, wobei die Spur und die Frobeniusnorm kontinuierlich ansteigen. Bei dieser Simulation wurde zur besseren Illustration das Fenster des Schätzfehlers auf 5000 Werte vermindert (entspricht 5 s), was ein schnelleres Abfallen des geglätteten Schätzfehlers zur Folge hat. Bei ausbleibender Information, bspw. durch Stillstand, steigen die Normen stärker als der gewählte Parameter ρ. Daher wird der gleitende Schätzfehler als Konvergenzmaß unter der Voraussetzung herangezogen, dass die Steigung der Normen unter dem Parameter ρ liegt. Für eine numerische Ableitung eignet sich die Frobeniusnorm eher als die Spur, da diese weniger Rauschen unterliegt.

Das IEKF hat in den vorherigen Simulationen sehr gute Ergebnisse gezeigt. Es wurde eine geeignetes reduziertes Modell angenähert, wobei Konvergenz nach 5 min-10 min Simulationszeit erreicht wurde. Das IEKF hat erfolgreich das anzunähernde System fünfter Ordnung, mit der Annäherung des Zeitverzugs durch eine Padé-Approximation dritter Ordnung, auf ein Modell dritter Ordnung reduziert. Ein so angenähertes reduziertes lineares Modell des Roboterverhaltens soll im Folgenden als Modell zur Synthese einer ARF eingesetzt werden.

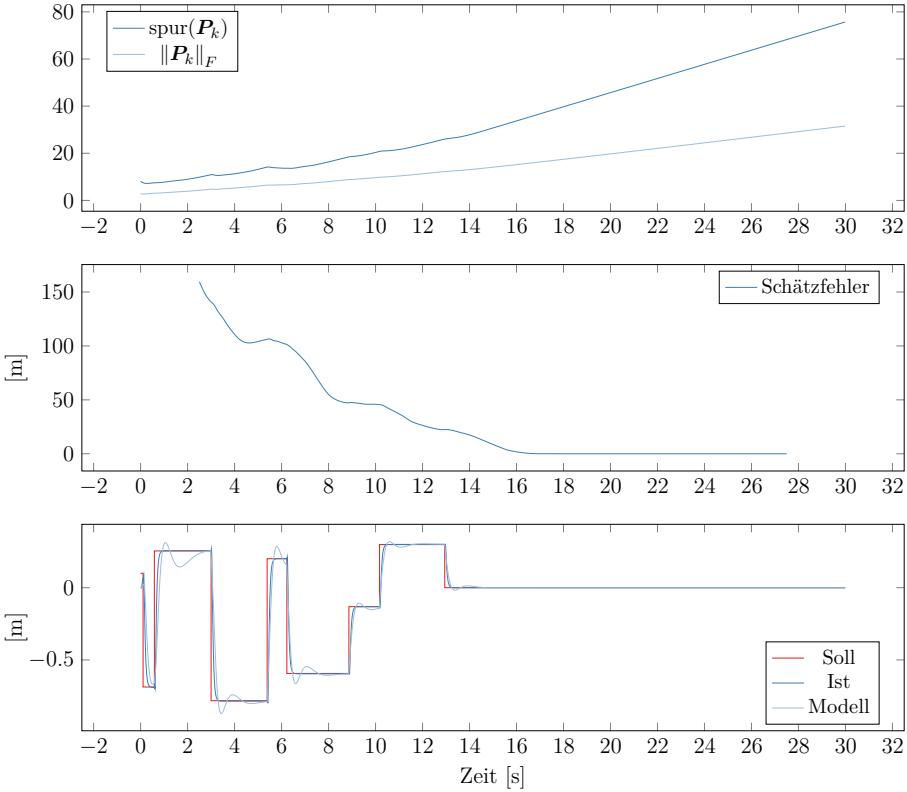


Abbildung 3.7: Verläufe von Konvergenzmaßen bei Roboterstillstand

3.2 Modellbasierter Entwurf einer sättigenden dynamische Ausgangsrückführung

In diesem Kapitel wird auf den Entwurf einer dynamischen Ausgangsrückführung für die Positionsregelung des Roboters eingegangen. Eine Modellannäherung anhand von Geschwindigkeitssprungexperimenten ist bei einer Fahrt im freien Raum unproblematisch, solange keine physikalischen Hindernisse im Verfahrweg des Roboters vermieden werden müssen.

Der modellbasierte Regleransatz der sättigenden dynamischen Ausgangsrückführung (ARF) wurde aus zwei Gründen ausgewählt. Der Entwurf nutzt die Stellgrößenbeschränkung des Systems aktiv aus, um nahezu zeitoptimales Verhalten zu erreichen, was bedeutet, dass der Roboter eine möglichst lange Zeit mit der maximal zugelassenen Geschwindigkeit verfährt, und die Parameter ergeben sich aus einer konvexen Optimierung ohne notwendiges Einwirken des Bedieners.

Das hier betrachtete Regelungssystem besteht prinzipiell aus zwei Komponenten. Zum Einem aus einem Zustandsraumregler, der durch Auswertung des aktuellen Zustandsvektors, beispielsweise zusammengesetzt aus Position, Geschwindigkeit und Beschleunigung, und einem entsprechenden Systemmodell, eine kontinuierliche Systemanregung (Stellgröße $\boldsymbol{u}$) berechnet um das zu System möglichst schnell in einen gewünschten Zustand zu überführen. Zum anderen aus einem Zustandsschätzer, welcher die internen Zustände des Systems schätzt, da dieser typischerweise nicht (vollständig) gemessen werden können.

Für beide Komponenten werden spezifische Formen angenommen, die jeweils durch den Entwurf parametrisiert werden sollen. Es wird von Grenzen der internen Zustände X_0 und Grenzen der für den Ausgang zulässigen Systemanregung $\boldsymbol{u}_{\max}$ ausgegangen, die nicht überschritten werden können oder dürfen. In der linearen Regelungstechnik werden solche Sättigungsphänomene vermieden und somit meist in dem Entwurf vernachlässigt.

Der zulässige Wertebereich innerhalb $\pm\boldsymbol{u}_{\max}$ soll voll ausgenutzt werden um höhere Regelungsgüte, bzw. schnellere Ausregelung zu erreichen. Dadurch kann das Seperationstheorem der linearen Regelungstechnik nicht mehr angewendet werden, welches den getrennten Entwurf von Reglern und Beobachtern ermöglicht. Vielmehr muss auf die Wechselwirkung von Regler und Beobachter eingegangen werden und der Entwurf gleichzeitig oder iterativ abgestimmt erfolgen.

Lens beschreibt in (Lens 2010) einen Ansatz zum gleichzeitigen Entwurf eines Regler-Beobachter-Systems, indem die strukturellen Grenzen zwischen beiden Komponenten aufgelöst werden. Es

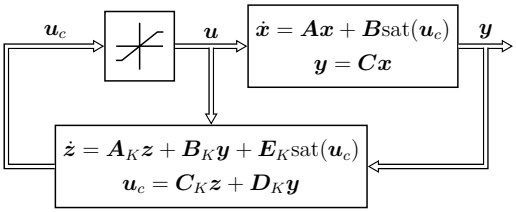


Abbildung 3.8: Blockdiagramm der vorgeschlagenen dynamischen Ausgangsrückführung (Lens 2010)

.

wird eine allgemeine dynamische Ausgangsrückführung mit der gleichen Ordnung wie die Regelstrecke angesetzt. Damit ergibt sich ein quasi-konvexes LMI Optimierungsproblem, welches effektiv global gelöst werden kann. Der dynamische Regler mittels Ausgangsrückführung, kurz auch die (dynamische) Ausgangsrückführung (ARF), besteht aus einem dynamischen System, zu dem die gemessenen Ausgänge der Regelstrecke als Eingang dienen. Die Ausgänge des dynamischen Reglers sind die Stellgrößen für die Regelstrecke.

Das folgende Regelungssystem

$$K : \begin{cases} \dot{\boldsymbol{z}} & = \boldsymbol{A}_K\boldsymbol{z} + \boldsymbol{B}_K\boldsymbol{y} + \boldsymbol{E}_K\mathrm{sat}(\boldsymbol{u}_c) \\ \boldsymbol{u}_c & = \boldsymbol{C}_K\boldsymbol{z} + \boldsymbol{D}_K\boldsymbol{y} \end{cases} \tag{3.26}$$

wird für das System

$$P : \begin{cases} \dot{\boldsymbol{x}} & = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\mathrm{sat}(\boldsymbol{u}_c)\,,\ \boldsymbol{x}_0 \in X_0 \\ \boldsymbol{y} & = \boldsymbol{C}\boldsymbol{x} \end{cases} \tag{3.27}$$

vorgeschlagen.

Das zu regelnde Zustandsraumsystem P (3.27) enthält das nichtlineare Sättigungsglied $\mathrm{sat}(\boldsymbol{u}_c)$, welches die k Eingangsgrößen in $\boldsymbol{u}_c$ mit jeweils $\pm u_{\max,k}$ begrenzt. Die initialen Zustände $\boldsymbol{x}_0$ befinden sich innerhalb des zulässigen Gebiets X_0.

Die Ausgangsrückführung K (ARF) besteht aus der Zustandsübergangsmatrix $\boldsymbol{A}_K$, der Systemausgangsmatrix $\boldsymbol{B}_K$, der Ausgangsmatrix $\boldsymbol{C}_K$ und der Durchgangssmatrix $\boldsymbol{D}_K$. Zusätzlich ist die Zustandsübergangsgleichung mit der Matrix $\boldsymbol{E}_K$ erweitert, welche die gesättigte Stellgröße zur internen Zustandsschätzung einbezieht. Das System K (3.26) ist wegen des Sättigungsglieds $\mathrm{sat}(\cdot)$ nichtlinear. Das Blockdiagramm in Abbildung 3.8 zeigt die Kopplungen der Systeme.

Um die Stabilität des entworfenen sättigenden Regelungssystems nachweisen zu können, ist die Existenz eines analogen Regelungssystems

$$H : \begin{cases} \dot{\boldsymbol{z}} & = \boldsymbol{A}_K \boldsymbol{z} + \boldsymbol{B}_K \boldsymbol{y} + \boldsymbol{E}_K \boldsymbol{u}_H \\ \boldsymbol{u}_H & = \boldsymbol{C}_H \boldsymbol{z} + \boldsymbol{D}_H \boldsymbol{y} \end{cases} \tag{3.28}$$

notwendig, welches die gleiche Dynamik besitzt, jedoch die Stellgröße so bestimmt, dass keine Sättigung auftritt.

Hiermit entfällt das Sättigungsglied bei dem System H gegenüber K und Ljapunov Stabilität kann daher zunächst für K und anschließend auch für H nachgewiesen werden. Für die Herleitung und den formalen Nachweis sei an dieser Stelle auf (Lens 2010) verwiesen.

Die gesuchten Größen des zu spezifizierenden Optimierungsproblems umfassen die Parametermatrizen der ARF und des analogen Systems H.

Mit der Forderung $n_z = n_x = n$, die Dimension n_x des zu regelnden Systems P gleicht der Dimension n_z der ARF, kann eine kongruente Variablentransformation (Chilali et al. 1996) angewandt werden. Die Transformation ermöglicht eine Variablensubstitution mit $\boldsymbol{X}$ und $\boldsymbol{Y}$, die zu einer konvexen LMI Formulierung führt.

Das Optimierungsproblem (3.29) sucht nach gültigen semidefiniten Matrizenvariablen $\boldsymbol{X}$, $\boldsymbol{Y}$, $\hat{\boldsymbol{A}}_K$, $\hat{\boldsymbol{B}}_K$, $\hat{\boldsymbol{C}}_K$, $\hat{\boldsymbol{D}}_K$, $\boldsymbol{W}$, $\hat{\boldsymbol{C}}_H$, $\hat{\boldsymbol{D}}_H$ mit einen maximalen Wert für die skalare Variablen δ.

$$\max \ \delta \text{, sodass}$$

$$\begin{bmatrix} \boldsymbol{X} & \boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{Y} \end{bmatrix} \succ 0 \ , \tag{3.29a}$$

$$\begin{bmatrix} \boldsymbol{\Gamma}_{11} & \boldsymbol{\Gamma}_{12} \\ \boldsymbol{\Gamma}_{12}^{\mathsf{T}} & \boldsymbol{\Gamma}_{22} \end{bmatrix} \prec -2\delta \begin{bmatrix} \boldsymbol{X} & \boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{Y} \end{bmatrix} \ , \tag{3.29b}$$

$$\begin{bmatrix} \boldsymbol{\Lambda}_{11,i} & \boldsymbol{\Lambda}_{12,i} \\ \boldsymbol{\Lambda}_{12,i}^{\mathsf{T}} & \boldsymbol{\Lambda}_{22,i} \end{bmatrix} \prec 0 \, , \ \forall i = 2, \ldots, 2^{n_u} \ , \tag{3.29c}$$

$$\begin{bmatrix} \boldsymbol{X} & \boldsymbol{I} & \hat{\boldsymbol{C}}_H^{\mathsf{T}} \\ \boldsymbol{I} & \boldsymbol{Y} & (\hat{\boldsymbol{D}}_H \boldsymbol{C})^{\mathsf{T}} \\ \hat{\boldsymbol{C}}_H & \hat{\boldsymbol{D}}_H \boldsymbol{C} & \boldsymbol{W} \end{bmatrix} \succeq 0 \ , \tag{3.29d}$$

$$w_{jj} \leq u_{\max,j}^2 \ , \tag{3.29e}$$

$$\boldsymbol{x}_{0,k}^{\mathsf{T}} \boldsymbol{Y} \boldsymbol{x}_{0,k} \leq 1 \ , \tag{3.29f}$$

mit den Substitutionen

$$
\begin{aligned}
\boldsymbol{\Gamma}_{11} &= \boldsymbol{X}\boldsymbol{A}^\intercal + \boldsymbol{A}\boldsymbol{X} + \boldsymbol{B}\hat{\boldsymbol{C}}_K + (\boldsymbol{B}\hat{\boldsymbol{C}}_K)^\intercal \ , \\
\boldsymbol{\Gamma}_{12} &= \hat{\boldsymbol{A}}_K^\intercal + \boldsymbol{A} + \boldsymbol{B}\hat{\boldsymbol{D}}_K\boldsymbol{C} \ , \\
\boldsymbol{\Gamma}_{22} &= \boldsymbol{A}^\intercal\boldsymbol{Y} + \boldsymbol{Y}\boldsymbol{A} + \hat{\boldsymbol{B}}_K\boldsymbol{C} + (\hat{\boldsymbol{B}}_K\boldsymbol{C})^\intercal \ , \\
\boldsymbol{\Lambda}_{11,i} &= \boldsymbol{X}\boldsymbol{A}^\intercal + \boldsymbol{A}\boldsymbol{X} + \boldsymbol{B}\hat{\boldsymbol{C}}_i + (\boldsymbol{B}\hat{\boldsymbol{C}}_i)^\intercal \ , \\
\boldsymbol{\Lambda}_{12,i} &= \hat{\boldsymbol{A}}_K^\intercal + \boldsymbol{A} + \boldsymbol{B}\hat{\boldsymbol{D}}_i\boldsymbol{C} \ , \\
\boldsymbol{\Lambda}_{22,i} &= \boldsymbol{A}^\intercal\boldsymbol{Y} + \boldsymbol{Y}\boldsymbol{A} + \hat{\boldsymbol{B}}_K\boldsymbol{C} + (\hat{\boldsymbol{B}}_K\boldsymbol{C})^\intercal \ , \\
\hat{\boldsymbol{C}}_i &= \boldsymbol{\Delta}_i\hat{\boldsymbol{C}}_K + \boldsymbol{\Delta}_i^-\hat{\boldsymbol{C}}_H \ , \\
\hat{\boldsymbol{D}}_i &= \boldsymbol{\Delta}_i\hat{\boldsymbol{D}}_K + \boldsymbol{\Delta}_i^-\hat{\boldsymbol{D}}_H \ , \\
\boldsymbol{\Delta}_1 &= \boldsymbol{I}_{n_u} \ , \\
\boldsymbol{\Delta}_i &= \operatorname{diag}(\alpha_1, \ldots, \alpha_{n_u}) \ , \\
&\qquad \alpha_j \in \{0;1\}, \quad \forall j = 1, \ldots, n_u \ , \\
\boldsymbol{\Delta}_i^- &= \boldsymbol{I}_{n_u} - \boldsymbol{\Delta}_i \ .
\end{aligned}
$$

In den Gleichungen (3.29), repräsentiert $\boldsymbol{I}$ eine $n \times n$ Einheitsmatrix, $\boldsymbol{I}_{n_u}$ repräsentiert eine $n_u \times n_u$ Einheitsmatrix, w_{jj} steht für den $j$ten Eintrag auf der Hauptdiagonalen der Matrix $\boldsymbol{W}$, $\boldsymbol{x}_{0,k}$ repräsentiert die $k$te Kante des zulässigen Gebiets X_0.

Die Bedingungen (3.29a), (3.29b) und (3.29c) fordern die Existenz einer Ljapunov Funktion für das sättigende und das nicht sättigende System mit jeweils K und H. Bedingung (3.29f) fordert den Einschluss des Anfangsgebietes X_0 in die Ljapunov Funktionen. Die Bedingungen (3.29d) und (3.29e) fordern die Einhaltung der Stellgrößen- und Zustandsbegrenzungen.

Weitere Nebenbedingungen können eingeführt werden, um die Dynamik des geregelten Systems zu beschränken. Hiermit können numerische Probleme verringert werden. Hierfür werden die resultierenden Eigenwerte λ_i des geregelten Systems in einen zulässigen Bereich beschränkt. Um die Real- und Imaginärteile der Eigenwerte λ_i nach

$$
\begin{aligned}
\operatorname{Re}(\lambda_i) &\leq -\rho \ \text{ und} \\
|\operatorname{Im}(\lambda_i)| &\leq \vartheta |\operatorname{Re}(\lambda_i)|
\end{aligned}
$$

mit den Parameter ρ und ϑ einzuschränken, unterliegt das Optimierungsproblem (3.29) den zusätzlichen Bedingungen

$$
\boldsymbol{\Xi}^\intercal + \boldsymbol{\Xi} \succeq -\rho \begin{bmatrix} \boldsymbol{X} & \boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{Y} \end{bmatrix} \ , \tag{3.29g}
$$

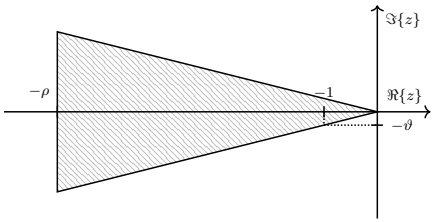


Abbildung 3.9: Grafische Darstellung der Begrenzung der platzierten Eigenwerte im Entwurfsansatz durch ρ und ϑ. Die Winkel des gleichschenklichen Dreiecks werden durch ϑ und die Höhe durch ρ definiert.

$$\begin{bmatrix} \vartheta(\boldsymbol{\Xi}^\mathsf{T}+\boldsymbol{\Xi}) & \boldsymbol{\Xi}^\mathsf{T}-\boldsymbol{\Xi} \\ -\boldsymbol{\Xi}^\mathsf{T}+\boldsymbol{\Xi} & \vartheta(\boldsymbol{\Xi}^\mathsf{T}+\boldsymbol{\Xi}) \end{bmatrix} \prec 0 \tag{3.29h}$$

mit

$$\boldsymbol{\Xi} = \begin{bmatrix} \boldsymbol{A}\boldsymbol{X}+\boldsymbol{B}\hat{\boldsymbol{C}}_K & \boldsymbol{A}+\boldsymbol{B}\hat{\boldsymbol{D}}_K\boldsymbol{C} \\ \hat{\boldsymbol{A}}_K & \boldsymbol{Y}\boldsymbol{A}+\hat{\boldsymbol{B}}_K\boldsymbol{C} \end{bmatrix} .$$

Der, durch die Parameter ρ und ϑ bestimmte Bereich der Eigenwerte ist in Abbildung 3.9 dargestellt. Die Winkel des gleichschenkligen Dreiecks werden durch ϑ und die Höhe durch ρ definiert.

Nach erfolgreicher Lösung des Problem (3.29) müssen die Parametermatrizen der ARF rekonstruiert werden. Durch die angewendete kongruente Transformation sind $\boldsymbol{X}$ und $\boldsymbol{Y}$ zerlegbar zu

$$\boldsymbol{M}\boldsymbol{N}^\mathsf{T} = \boldsymbol{I} - \boldsymbol{X}\boldsymbol{Y} \ . \tag{3.30}$$

Die quadratischen, invertierbaren Matrizen $\boldsymbol{M}$ und $\boldsymbol{N}$ können immer mit einer geeigneten Zerlegung, beispielsweise mit einer QR-Zerlegung gefunden werden.

Zur Rekonstruktion der Matrizen der ARF werden die Gleichungen

$$\boldsymbol{E}_K = \boldsymbol{N}^{-1}\boldsymbol{Y}\boldsymbol{B} \ , \tag{3.31a}$$
$$\boldsymbol{D}_K = \hat{\boldsymbol{D}}_K \ , \tag{3.31b}$$
$$\boldsymbol{C}_K = (\hat{\boldsymbol{C}}_K - \boldsymbol{D}_K\boldsymbol{C}\boldsymbol{X})(\boldsymbol{M}^{-1})^\mathsf{T} \ , \tag{3.31c}$$
$$\boldsymbol{B}_K = \boldsymbol{N}^{-1}\hat{\boldsymbol{B}}_K \text{ und} \tag{3.31d}$$
$$\boldsymbol{A}_K = \boldsymbol{N}^{-1}(\hat{\boldsymbol{A}}_K - \boldsymbol{Y}\boldsymbol{A}\boldsymbol{X})(\boldsymbol{M}^{-1})^\mathsf{T} - \boldsymbol{B}_K\boldsymbol{C}\boldsymbol{X}(\boldsymbol{M}^{-1})^\mathsf{T} \tag{3.31e}$$

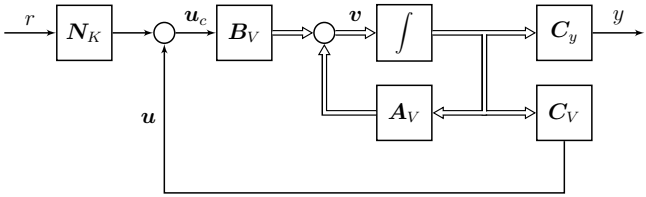


Abbildung 3.10: Blockdiagramm der erweiterten Zustandsraumdarstellung

angewendet.

Wie bei Zustandsregelungen üblich, ist auch bei der dynamischen Rückführung ein Vorfilter $\boldsymbol{N}_K$ notwendig, um stationäre Genauigkeit für eine gegebene Führungsgröße $\boldsymbol{x}_d$ sicherzustellen. Dieses Vorfilter wird im Folgenden für einen skalaren Sollwert r und einen skalaren Ausgangswert y hergeleitet, damit der praktische Einsatz der ARF möglich ist.

Hierfür wird das geregelte System nach Abbildung 3.10 mit dem erweiterten Zustandsvektor $\boldsymbol{v}$ betrachtet, welches sich aus dem Zustand $\boldsymbol{x}$ der Strecke und dem internen Zustand $\boldsymbol{z}$ der ARF zusammensetzt. Die entsprechende Zustandsraumdarstellung ergibt sich zu

$$V : \begin{cases} \dot{\boldsymbol{v}} & = \boldsymbol{A}_V \boldsymbol{v} + \boldsymbol{B}_V \boldsymbol{u}_c \\ \boldsymbol{u}_c & = \boldsymbol{C}_V \boldsymbol{v} + \boldsymbol{N}_K r \\ y & = \boldsymbol{C}_y \boldsymbol{v} \end{cases} \tag{3.32}$$

mit

$$\begin{aligned} \boldsymbol{v} &= \begin{bmatrix} \boldsymbol{x} & \boldsymbol{z} \end{bmatrix}^\mathsf{T} \ , \\ \boldsymbol{A}_V &= \begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{B}_K \boldsymbol{C} & \boldsymbol{A}_K \end{bmatrix} \ , \\ \boldsymbol{B}_V &= \begin{bmatrix} \boldsymbol{B} & \boldsymbol{0} \end{bmatrix}^\mathsf{T} \ , \\ \boldsymbol{C}_V &= \begin{bmatrix} \boldsymbol{D}_K \boldsymbol{C} & \boldsymbol{C}_K \end{bmatrix} \ , \\ \boldsymbol{C}_y &= \begin{bmatrix} \boldsymbol{C} & \boldsymbol{0} \end{bmatrix} \ . \end{aligned}$$

Im stationären Fall ist die Änderung des Zustands $\dot{\boldsymbol{v}} = \boldsymbol{0}$ und der stationäre Ausgang y soll dem Sollwert $r \stackrel{!}{=} y$ gleichen.

Damit ergibt sich durch Einsetzen der Gleichungen (3.32),

$$\mathbf{0} = \boldsymbol{A}_V \boldsymbol{v} + \boldsymbol{B}_V \boldsymbol{C}_V \boldsymbol{v} + \boldsymbol{B}_V \boldsymbol{N}_K r \ , \tag{3.33}$$

$$y = r = \boldsymbol{C}_y \boldsymbol{v} \ , \tag{3.34}$$

und nach Umformung das Vorfilter $\boldsymbol{N}_K$

$$\begin{cases} \boldsymbol{N}_K = \left[-\boldsymbol{C}_y [\boldsymbol{A}_V + \boldsymbol{B}_V \boldsymbol{C}_V]^{-1} \boldsymbol{B}_V \right]^{-1} , & \forall r \neq 0 \ , \\ y = r = 0 \, , & \text{sonst} \ . \end{cases} \tag{3.35}$$

Lens arbeitet in seinen Untersuchungen bislang unter der Annahme von zeitkontinuierlichen Systemen. Allerdings kann die Annahme im Zuge dieser Arbeit nicht gehalten werden. Bei direkter Anwendung der synthetisierten ARF zeigen sich bereits bei einer Taktung der Roboterkommunikation von 250 Hz erste unerwünschte Schwingungseffekte. Daher wird in dieser Arbeit das nach (Lens 2010) synthetisierte System um einen Diskretisierungsschritt erweitert.

Die hierzu nötigen Herleitungen sind analog zu der Diskretisierung eines Zustandsraumsystems im Bereich der *Digitalen Regelung*. Da die Form der ARF allerdings von der klassischen Zustandsraumdarstellung abweicht, können die vorhandenen Softwarewerkzeuge nicht direkt eingesetzt werden. Eine Herleitung der entsprechenden Diskretisierung der Zustandsraumstruktur der ARF ist in Anhang A.2 durchgeführt.

3.2.1 Implementierung

Das zuvor in Kapitel 3.2 eingeführte Optimierungsproblem (3.29) soll implementiert werden. Die zugrundeliegenden Vorarbeiten von Lens nutzen MATLAB mit der Erweiterung *Robust Control Toolbox* für die LMI Optimierung.

In dieser Arbeit wird auf den Einsatz von MATLAB verzichtet. Als Umgebung für die Programmierung wird weiterhin die Sprache Python mit dem Paket *cvxpy* für die Formulierung von konvexen Optimierungsproblemen eingesetzt.

Ein zuvor identifiziertem Modell des Robotersystems entsprechend

$$\begin{cases} \dot{\tilde{\boldsymbol{x}}} = \tilde{\boldsymbol{A}} \tilde{\boldsymbol{x}} + \tilde{\boldsymbol{b}} \boldsymbol{u} \\ \tilde{\boldsymbol{y}} = \tilde{\boldsymbol{C}} \tilde{\boldsymbol{x}} \end{cases} \tag{3.36}$$

liegt vor. Das Modell beschreibt das dynamische Geschwindigkeitsverhalten $\tilde{\boldsymbol{y}}$ bei einer Geschwindigkeits-Sollvorgabe. Dieses Modell kann direkt experimentell ermittelt werden. Die Zustandsraummatrizen sind mit einer Tilde markiert um diese im Folgenden von positionsbasierten Modell (3.37) zu unterscheiden.

Das in Kapitel 3.1 identifizierte Modell ist eine Annäherung des Verhaltens des offenen Regelkreises auf Geschwindigkeits-Sollvorgaben. Dem System werden Zielgeschwindigkeiten vorgegeben und die Systemgeschwindigkeit wird beobachtet.

Um eine Positionsregelung zu entwerfen, muss das Modell um den Zustand der Position erweitert werden. Das ist einfach unter der Annahme möglich, dass die Position sich aus der Integration der Geschwindigkeit ergibt. Das resultierende System (mit Positionsausgang) ist ohne Regelung instabil, da der zugefügte (ideale) Integrator einen Systempol bei Null hinzufügt.

Der formale Vorgang zum Anfügen eines Ausgangsintegrators lautet

$$\boldsymbol{A} = \begin{bmatrix} 0 & \tilde{\boldsymbol{C}} \\ \boldsymbol{0} & \tilde{\boldsymbol{A}} \end{bmatrix}, \; \boldsymbol{b} = \begin{bmatrix} 0 \\ \tilde{\boldsymbol{b}} \end{bmatrix} \text{ und } \boldsymbol{C} = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} . \tag{3.37}$$

Die Maximalwerte der Ausgangssättigung $\pm u_{\max,k}$ und die zulässigen Grenzen $\boldsymbol{x}_0$ des Anfangszustandsgebiets X_0 sind noch zu wählen. Diese richten sich nach maschinenspezifischen Eigenschaften, die ggf. dem Roboterdatenblatt entnommen werden können. Die Werte ergeben sich aus dem Aufgabenraum, des der Regelung unterliegenden Modells.

Da die internen Zustände $\boldsymbol{x}$ der Modellierung unterliegen und somit deren Begrenzungen physikalisch nicht unmittelbar deutbar sind, ist es sinnvoll, das System (3.37) in die modale Steuerungsnormalform zu transformieren. In dieser Form bildet der Zustandsvektor die zugrundeliegende Differenzialgleichung in genau einem Zustand ab. Die anderen Zustände bilden sich aus den Integrationen der Differenzialgleichung.

Aus numerischen Gründen kann es vorteilhaft sein, nicht das System selbst mit der Transformationsmatrix $\boldsymbol{T}$ in Steuerungsnormalform zu transformieren, sondern die in Steuerungsnormalform angebenden Grenzen als beispielsweise

$$\bar{\boldsymbol{x}}_0 = \begin{bmatrix} \text{Position} & \text{Geschwindigkeit} & \text{Beschleunigung} & \text{Ruck} \end{bmatrix}^\intercal$$

anzugeben und in die nicht in Steuerungsnormalform vorliegende Repräsentation $\boldsymbol{x}_0 = \boldsymbol{T}\bar{\boldsymbol{x}}_0$ zu transformieren.

Zur Erstellung des Optimierungsproblems werden alle Nebenbedingungen spezifiziert und letztendlich an das `Problem` Objekt angehängt. Die Listen an Randbedingungen aus (3.29c) und (3.29e) können analog zu (3.29f) folgendermaßen

```
constraints_3.30f = [ x0_k.T * Y * x0_k <= 1 for x0_k in X0 ]
```

formuliert werden.

Um das Optimierungsproblem lösen zu können, muss die Optimierungsvariable δ als `Parameter` deklariert werden. Streng genommen ist die Nebenbedingung (3.29b) nicht konvex, da sich die Variablen $\boldsymbol{X}$ und $\boldsymbol{Y}$ mit δ multiplizieren. Wird δ auf einen Wert festgelegt und nicht weiter als Optimierungsvariable betrachtet, kann das resultierende konvexe Validierungsproblem gelöst werden. Ist das Problem lösbar, wird δ sukzessive erhöht, bis keine Lösung mehr gefunden werden kann. Der Optimalwert der Variable δ ergibt sich zu dem größten gefundenen Wert $\delta_{\max}$, für den das Problem (3.29) lösbar ist. Diese Klasse an Optimierungsproblemen wird auch als quasikonvex bezeichnet.

Um den Wert $\delta_{\max}$ effektiv zu finden wird ein Bisektionsalgorithmus eingesetzt. Ein Bisektionsalgorithmus unterteilt den Suchraum in jedem Schritt mittig und untersucht jeweils die Zentren der beiden Seiten auf Lösbarkeit. Die Unterteilung und Untersuchung wird sukzessiv auf der lösbaren Seite mit größten δ wiederholt, bis der Suchraum ausreichend klein geworden ist.

Eine solche Bisektion erreicht deutlich schnellere Konvergenz als eine lineare Suche. Ein solcher Algorithmus ist bereits in *YALMIP* integriert. Da dies allerdings in der aktuellen Version von *cvxpy* nicht der Fall ist, wurde ein Bisektionsalgorithmus nachträglich implementiert (Vgl. Anhang A.3, Algorithm 1).

Zur Lösung des Problems wird es zunächst als Validierungsproblem deklariert und mit dem Bezeichner des Parameters δ an den Bisektionsalgorithmus übergeben.

Da die gesuchte Variable $\delta \geq 0$ ist, kann als untere Schranke stets der Wert 0 angesetzt werden. Bei fehlender Angabe einer oberen Schranke wird diese vor der eigentlichen Bisektion selbstständig gefunden. Hierfür wird der Wert für $\delta \geq 1$ solange verdoppelt, bis das Validierungsproblem als unlösbar gilt. Der größte als lösbar gefundene Wert für δ wird als untere Schranke genutzt.

Obwohl konvexe Optimierungsprobleme ein eindeutiges globales Optimum besitzen, können verschiedene Solver zu unterschiedlichen Ergebnissen führen. So können bspw. die numerische Robustheit, die interne Realisierung von konvexen Repräsentationen (sog. Atomen), die Konditionierung der Matrizen oder die solver-spezifischen Annäherungen der Gradienten Einfluss nehmen. Weiter wird ein Solver nach endlich vielen Iterationsschritten abgebrochen, selbst wenn das Problem noch nicht mit ausreichender Genauigkeit gelöst wurde.

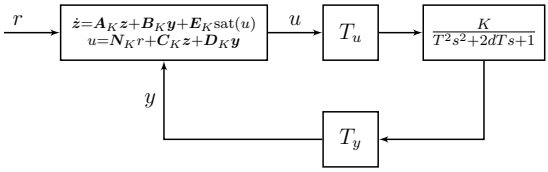


Abbildung 3.11: Blockdiagram der Simulation der Ausgangsrückführung

Aus diesen Gründen wurde der Bisektionsalgorithmus so erweitert, dass bei einem Fehler oder Abbruch des Solvers weitere alternative Solver angegeben werden können und ggf. automatisch eingesetzt werden.

3.2.2 Simulationsergebnisse

In dem folgenden Kapitel soll, ähnlich wie bei dem vorangehenden Kapitel 3.1.3, eine entworfenen Ausgangsrückführung (ARF) auf ein simuliertes System angewendet werden um die Robustheit gegenüber Verzögerungszeiten und Einstellregeln für die Parameter der Polbegrenzung ρ und ϑ abzuleiten.

Die an der Simulation untersuchten Situationen umfassen, i) den Einfluss von systemeingangs- und sensorseitigen Verzugszeiten, ii) den Einfluss der Reglertaktung und iii) den Einfluss der Polbegrenzung durch die Parameter ρ und ϑ.

Die ARF wird auf eine Simulation des zu regelnden Systems angewendet, welches es erlaubt, die zu untersuchenden Parameter einzustellen. Dafür werden die eingangs- und rückführungsseitigen Verzugszeiten T_u und T_y jeweils durch nichtlineare Verzugsglieder repräsentiert. Zur Illustration wird hier auf ein vereinfachtes Blockdiagram in Abbildung 3.11 verwiesen, welches sich auf das Blockdiagram 2.5 aus Kapitel 2.3.3 bezieht.

Die Simulation des geregelten Systems wird mit Zeitabständen von 1 ms durchgeführt. Je nach angegebener Reglertaktrate wird die ARF nur zu den entsprechenden Taktungen ausgeführt und aktualisiert, während das System mit voller Taktrate von 1000 Hz berechnet wird.

Der Entwurf basiert auf dem initial angenäherten PT2 Verzögerungssystem mit einer Padé-Approximation erster Ordnung der Verzugszeit aus Gleichung (3.18) und den bekannten Parametern

$$K = 1.0042\,,\ T = 1.1234\,,\ d = 0.03006\,,\ t_d = 0.0331\ ,$$

welches mittels (3.37) um einen Integrator erweitert wurde.

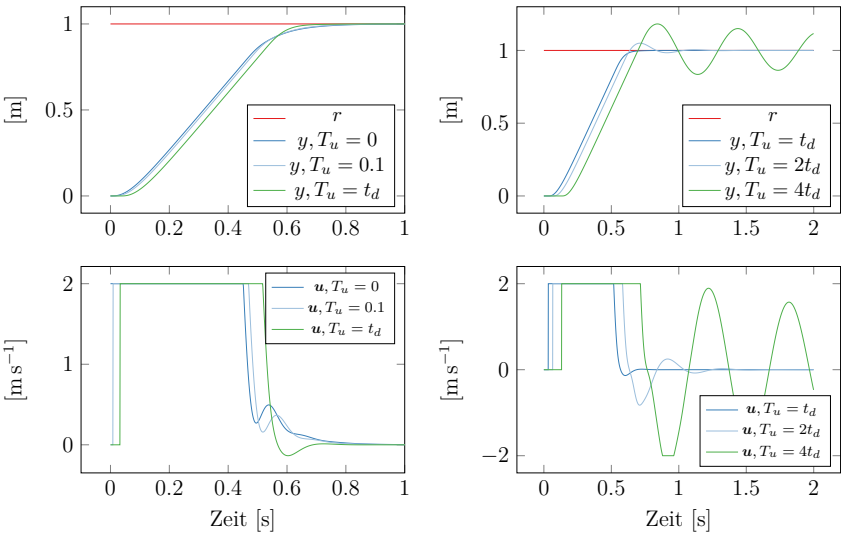


Abbildung 3.12: Variation der eingangsseitigen Verzugszeiten T_u bei vernachlässigten Verzug $T_y = 0$. Die Regelungstaktung entspricht 1000 Hz.

Als Ausgangsbegrenzung wurde $u_{\max} = 2\,\mathrm{m\,s^{-1}}$ festgelegt. Die Eckpunkte des zulässigen Einzugsgebietes X_0 wurden zu

$$\bar{\boldsymbol{x}}_0 = \begin{bmatrix} \pm 1 & \pm 2 & \pm 10 & \pm 100 \end{bmatrix}^\intercal$$

gewählt.

Zunächst werden die eingangsseitigen Verzugszeiten T_u variiert. Abbildung 3.12 zeigt die Regel- und Stellgrößenverläufe für einen Entwurf mit den Parametern $\rho = 100$ und $\vartheta = 1$. Für Verzugszeiten $T_u \leq t_d$ zeigt sich ein sehr schnelles Einregelverhalten. Der Sprung ist bei ca. $t \geq 0.8\,\mathrm{s}$ abgeschlossen. Für deutlich größere Verzugszeiten ergibt sich eine schwach gedämpfte Schwingung um die Führungsgröße.

Etwas sensibler reagiert das geregelte System auf zusätzliche rückführungsseitigen Verzugszeiten T_y. Für die in Abbildung 3.13 aufgezeigten Sprungantworten wurde die rückführungsseitigen Verzugszeiten T_y variiert, wobei die zuvor diskutierten eingangsseitigen Verzugszeiten auf $T_u = t_d$ festgelegt wurden.

Da sich die Verzugszeiten des Gesamtsystems in dieser Konfiguration addieren, muss die ARF mit insgesamt längeren Verzugszeiten umgehen. Systematische rückführungsseitigen Verzugszeiten von $T_y \leq 0.01\,\mathrm{s}$ sind für eine Kommunikationsraten mit der Robotersteuerung von mehr

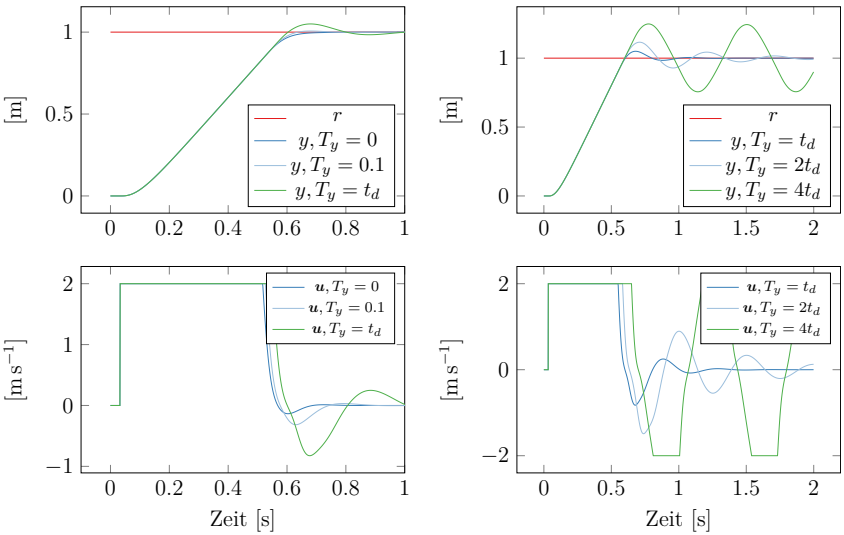


Abbildung 3.13: Variation der rückführungsseitigen Verzugszeiten T_y bei konstanten Verzug $T_u = t_d$. Die Regelungstaktung entspricht 1000 Hz.

als 100 Hz realistisch. Für diese Verzugszeiten wird immer noch ein schnelles Einregelverhalten erreicht. Der Sprung ist bei ca. $t \geq 0.9\,\mathrm{s}$ abgeschlossen.

Für die Untersuchung des Einfluss der Reglertaktung wird im Folgenden der eingangsseitige Verzug $T_u = t_d$ und $T_y = 0$ festgelegt. Die Regelung wird, entsprechend der eingestellten Taktung, nur zu diskreten Zeitpunkten ausgeführt. In den dazwischen liegenden Zeitspannen wird der letzte Wert des Reglerausgangs gehalten. Die Taktung Δt ist dem zeit-diskreten Regler bekannt und wird entsprechend

$$\boldsymbol{z}(k) = \dot{\boldsymbol{z}}(k-1) \cdot \Delta t$$

für den internen Zustandsübergang der ARF vom Zeitpunkt $k-1$ zum Zeitpunkt k eingesetzt.

Die Sprungantworten in Abbildung 3.14 illustrieren die Regelungs- und Stellgrößenverläufe für verschiedene Reglertaktungen. Es sind keine markanten Unterschiede in dem Verlauf der Regelgröße zu erkennen. Für Regelfrequenzen unter 125 Hz werden Veränderungen in den Stellgrößenverläufen sichtbar. Für die Taktung von 25 Hz ist ein verlängerter Einschwingvorgang erkennbar. Somit kann für die Anwendung an Robotersystemen mit typischerweise über 100 Hz die diskretisierte ARF eingesetzt werden.

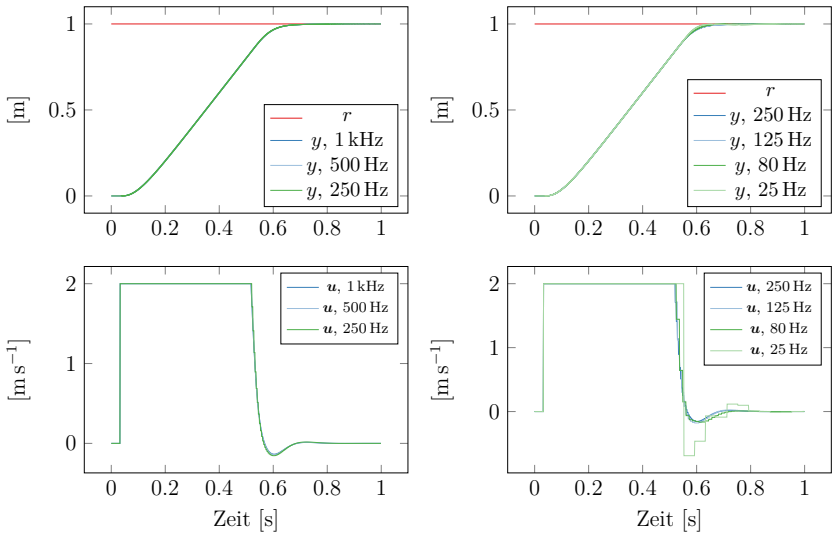


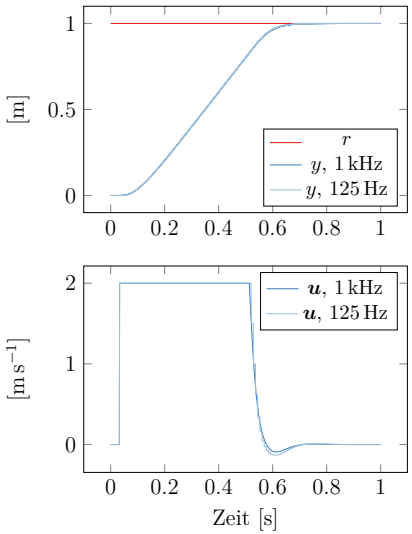
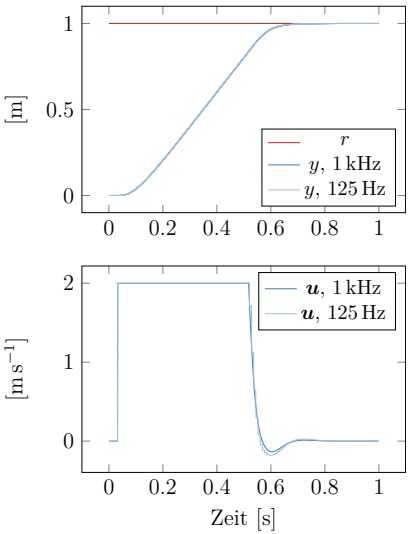
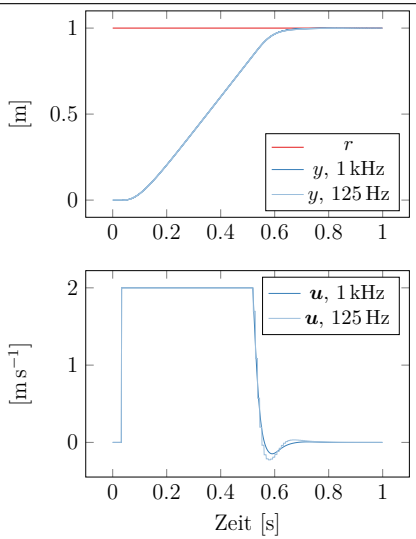
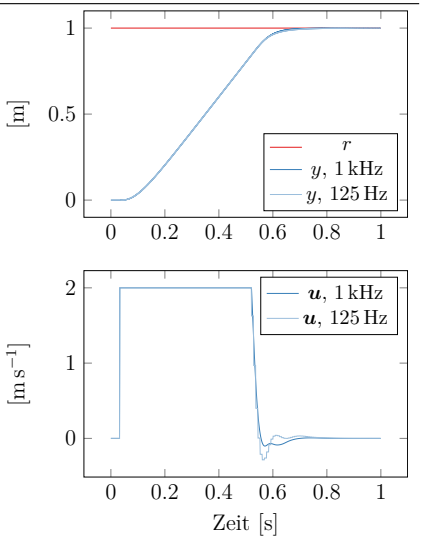
Abbildung 3.14: Variation der Regelungstaktung bei konstanten Verzug $T_u = t_d$ bei $T_y = 0$

Um numerische Probleme zu vermeiden können die Parameter ρ und ϑ eingesetzt werden um die durch den Entwurf platzierten Eigenwerte einzugrenzen. Betragsmäßig große Eigenwerte fordern durch die stärkere Dynamik eine höhere numerische Genauigkeit. Allerdings kann das dynamische Verhalten durch eine zu restriktive Eingrenzung negativ beeinflusst oder das Optimierungsproblem unlösbar werden. Im Folgenden wird der Einfluss der Begrenzung anhand der Simulation untersucht.

Die Parameter ρ und ϑ spannen einen zulässiger Bereich für die Eigenwerte des geregelten Systems auf. Der Parameter ρ begrenzt den maximalen Betrag des Realteils während der Parameter ϑ als Begrenzung des Imaginärteils relativ zum Realteil wirkt. Durch die Maximierung der Variable δ des Problems (3.29) werden die platzierten Eigenwerte immer weiter in die negative Halbebene getrieben. Abbildung 3.15 zeigt davon eine prinzipielle Darstellung.

Für die Positionsregelung von Robotern ist ein Überschwingen zu vermeiden, selbst wenn dadurch nicht die zeitoptimale Systemantwort erreicht werden kann. Für den Entwurf bedeutet das, dass der Parameter ϑ möglichst klein gewählt werden sollte. Setzen von $\vartheta = 0$ ist allerdings zu restriktiv für den Entwurf, sodass in den meisten Fällen keine Lösung gefunden werden kann.

Tabelle 3.3: Sprungantworten für verschiedene $\vartheta_{\max}$ bei gleichbleibenden $\rho = 100$

$\rho = 100.0$ $\vartheta = 0.75\,,\ \vartheta_{\max} = 75$	$\rho = 100.0$ $\vartheta = 1.0\,,\ \vartheta_{\max} = 100$
	
$\rho = 100.0$ $\vartheta = 2.0\,,\ \vartheta_{\max} = 200$	$\rho = 100.0$ $\vartheta = 5.0\,,\ \vartheta_{\max} = 500$
	

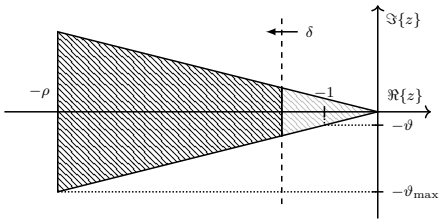


Abbildung 3.15: Grafische Darstellung des Suchraums der zu platzierten Eigenwerte des geregelten System bei Maximierung der Bisektionsvariable δ. Durch die Parameter ρ und ϑ wird ein zulässiger Bereich aufgespannt (grau schraffiert), welcher bei Vergrößerung Variable δ in negativer Richtung verkleinert wird (schwarz schraffiert).

Zur besseren Handhabe des Parameters ϑ wird eine Hilfsvariable

$$\vartheta_{\max} = \rho \cdot \vartheta$$

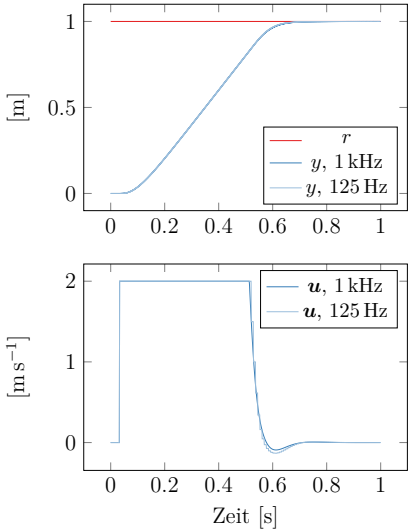
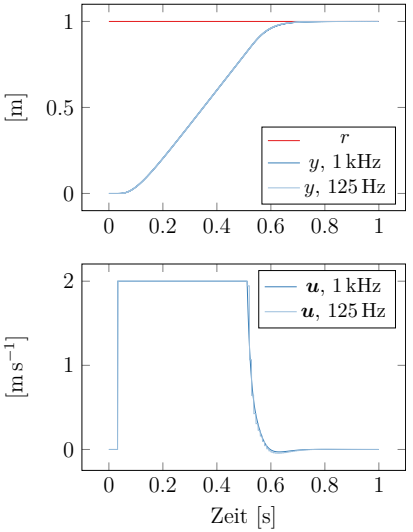
eingeführt. $\vartheta_{\max}$ beschreibt dabei, analog zu dem Parameter ρ, den betragsmäßig maximal erlaubten Imaginärteil der platzierten Eigenwerte.

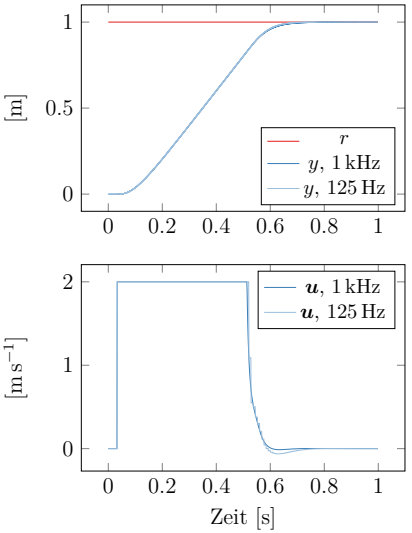
Tabelle 3.3 zeigt Sprungantworten für verschiedene Werte von $\vartheta_{\max}$ bei gleichbleibenden $\rho = 100$. Mit steigenden $\vartheta_{\max}$ nimmt das Überschwingen, bzw. das Schwingen der Stellgröße zu. Da die Auswirkungen auf die Istgröße nur schwach wahrnehmbar sind, wird ein akzeptabler Wert mit $\vartheta_{\max} = 75$ gewählt. Des Weiteren wird nun in Tabelle 3.4 $\vartheta_{\max} = 75$ festgelegt und der Wert für ρ erhöht. Es zeigt sich eine minimale Verkürzung der Einregelzeit mit höheren Werten für ρ.

Eines der Ziele der Untersuchung der ARF mit einem simulierten System war es sicherzustellen, dass der Regelungsansatz sich robust gegenüber systemeingangs- und sensorseitigen Verzugszeiten verhält. Anhand der Simulation kann angegeben werden, dass die ARF empfindlich auf eine fehlerhafte Modellierung der Verzugszeiten reagiert. In den Experimenten konnten nur Verzugszeiten von $2t_d$ ausgeglichen werden, wobei sich bereits hierbei deutliche Einschwingvorgänge zeigten. Eine korrekte Modellierung der Verzugszeiten und harte Echtzeitbedingungen sind somit für den korrekten Betrieb der ARF grundlegend. Weitestgehend robust verhält sich die ARF gegenüber der Reglertaktung. Mit typischen Regler- bzw. Roboterkommunikationsfrequenzen von 125 Hz sind gute Ergebnisse zu erzielen.

Ein weiteres Ziel der Untersuchung war es festzustellen ob und wie die freien Entwurfsparameter ρ und ϑ durch einen Bediener eingestellt werden sollten. Wie sich an den Simulationen verschiedener Einstellungen zeigt, bieten die freien Parameter kaum Optimierungspotential in

Tabelle 3.4: Sprungantworten für verschiedene ρ bei gleichbleibenden $\vartheta_{\max} = 75$

$\rho = 100.0$ $\vartheta = 0.75\,,\ \vartheta_{\max} = 75$	$\rho = 200.0$ $\vartheta = 0.375\,,\ \vartheta_{\max} = 75$
	

$\rho = 500.0$ $\vartheta = 0.15\,,\ \vartheta_{\max} = 75$


Bezug auf bessere Reglerperformanz. Die Werte müssen eine Lösung des Optimierungsproblems zulassen, allerdings praktisch nicht weiter feinjustiert werden. Die Begrenzung ρ des Realwertes der platzierten Pole ist hierbei maßgeblich um numerische Probleme vorzubeugen, wogegen eine zu kleine Wahl der Begrenzung ϑ der Imaginärteile der platzierten Pole keine Lösung des Optimierungsproblems zulässt. Daher wurde die Hilfsvariable $\vartheta_{\max}$ eingeführt, welche erlaubt die Begrenzungen voneinander unabhängig einzustellen. Bei numerischen Problemen kann der Wert von ρ, bei gleichbleibenden $\vartheta_{\max}$ verringert werden. Auf diese Weise kann eine Lösung bei kleinerem ρ gefunden werden. Mit Werten von $\rho < 200$ wurden in den Versuchen keine numerischen Probleme beobachtet.

3.3 Modellfreier Regler mit vorgegebenem Performanzband

In diesem Kapitel wird auf den Entwurf und die Anpassung eines modellfreien Reglerkonzepts eingegangen. Speziell für Kontaktsituationen sind die Modellbildung und die dafür erforderlichen Experimente schwierig durchzuführen. Eine reine Steuerung, wie bei den Sprungexperimenten in Kapitel 3.1, würde den Roboter zu stark belasten oder gar beschädigen.
Aus den oben genannten Gründen wird für die Skills zur Kraftregelung nicht auf ein modellbasiertes Verfahren zurückgegriffen. Das zugrunde liegende Konzept bezieht sich auf einen *Regler mit vorgegebenem Performanzband* (engl.: prescribed performance controller: PPC) (Kanakis et al. 2018).

Im Folgenden wird der generelle Ansatz des PPC dargestellt und für den Einbezug des PPC in das *pitasc* Rahmenwerk notwendigen Anpassungen und deren Veränderungen vorgestellt. Zusätzlich wird eine angepasste Struktur des PPC erarbeitet und Stabilität nachgewiesen.

Das grundlegende Ziel hinter PPC ist der Entwurf eines zeitkontinuierlichen Reglers, der die Einhaltung von gegebenen Kontaktkraftgrenzen garantiert. Ein aufgebauter Kontakt wird beibehalten und ein "Ablösen" von der Oberfläche verhindert. Dieses Verhalten ergibt sich, ohne dass zu große Kräfte aufgewendet werden. Zusätzlich werden für die Einregelphase ideale Verläufe und ein Toleranzband für die Regelabweichung vorgegeben.

Das PPC Konzept erreicht die vorher genannten Ziele dadurch, dass die Regelabweichung e durch eine Transformation $\varepsilon(e, P)$, mit Hinblick der Einhaltung der gegebenen Performanzgrenzen P moduliert wird. Dabei wird der Einschluss des Regelabweichungverlaufs $e(t)$ in das Performanzband P als *vorgegebene Performanz* (engl.: prescribed performance) bezeichnet.

Die Regelabweichung e für einen Kraftregler ist durch die Differenz aus dem gemessenen Kraftwert f und der Führungsgröße f_d als

$$e = f - f_d \tag{3.38}$$

definiert.

Um Überschwingen und somit den Verlust eines bereits aufgebauten Kontakts zu verhindern muss für die Regelabweichung $e(t) > 0\,,\ \forall e(0) \geq 0$ bzw. $e(t) < 0\,,\ \forall e(0) \leq 0$ gelten. Für die realistische Annahme von Sensorrauschen wird der stationäre Fehler sich allerdings nur in einem Toleranzband um Null einschließen lassen.

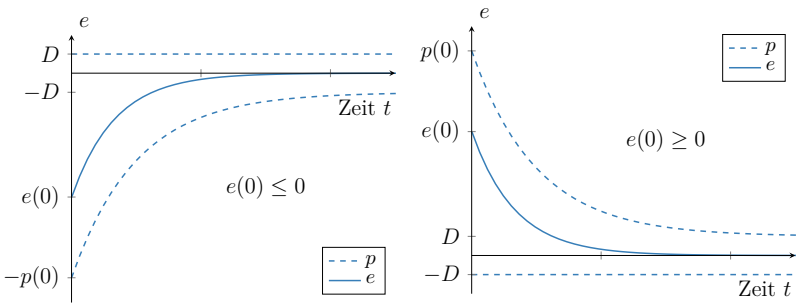


Abbildung 3.16: Darstellung der Performanzgrenzen mit idealisierter Darstellung der Regelabweichung

Hierzu wird eine positive Konstante D eingeführt, welche die Fehlertoleranz im stationären Zustand beschreibt. Der Bereich zwischen $\pm D$ sollte mindestens das stationäre Sensorrauschen einschließen. Das Performanzband P wird zu

$$P = \begin{cases} -D < e(t) < p(t)\,, & \forall e(0) \geq 0 \\ -p(t) < e(t) < D\,, & \forall e(0) \leq 0 \end{cases} \tag{3.39}$$

gewählt.

Die Performanzfunktion $p(t)$ muss als kontinuierlich differenzierbare, beschränkte, streng positive und monoton abfallende Funktion gewählt werden. Eine typische Wahl der Performanzfunktion $p(t)$ ist die Exponentialfunktion[5],

$$p(t) = (p_0 - D)\exp(-lt) + D \ , \tag{3.40}$$

mit p_0 als positive Konstante, welche die zu erwartende maximale Regelabweichung zum Beginn der Regelung $e(t = 0)$ einschließt und es gilt

$$p_0 \geq |e(0)| \ .$$

Der positive Parameter l repräsentiert die Abklingrate von $p(t)$. Für große Werte von l klingt $p(t)$ schneller ab.

Im Bezug zu dem zuvor definierten Performanzband P wird eine glatte streng monoton steigende Fehlertransformation $\varepsilon(t)$ gesucht. Zur besseren Lesbarkeit wird im Folgenden auf die

[5]Für die Exponentialfunktion $x \to e^x$ wurde die alternative Schreibweise $x \to \exp(x)$ gewählt, um Verwechslungen der Funktion mit der Regelabweichung $e(\cdot)$ zu vermeiden.

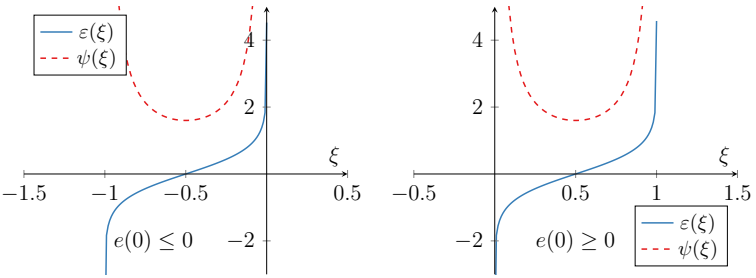


Abbildung 3.17: Darstellung der Fehlertransformation

explizite Darstellung der zeitlichen Abhängigkeit verzichtet.

Die vorgeschlagene Transformation aus (Kanakis et al. 2017) wird auf die Definition (3.39) von P angewendet und ergibt die Logarithmusfunktion

$$\varepsilon(e,p) = \begin{cases} \ln\left(\frac{D+e}{p-e}\right), & \forall e(0) \geq 0 \\ \ln\left(\frac{p+e}{D-e}\right), & \forall e(0) \leq 0 \end{cases} . \tag{3.41}$$

Um die Transformation (3.41) in die in (Kanakis et al. 2017) gegebene Form umzuwandeln und der analogen Herleitung folgen zu können, werden mit den Koordinatentransformationen

$$\begin{cases} \tilde{p} = p + D\,,\ \tilde{e} = e + D\,, & \forall e(0) \geq 0 \\ \tilde{p} = p + D\,,\ \tilde{e} = e - D\,, & \forall e(0) \leq 0 \end{cases} , \tag{3.42}$$

die Regelabweichung e und die Performanzfunktion p um den konstanten Anteil D verschoben. Durch die Verschiebung bildet sich ein Band von $\pm D$ im stationären Zustand welches Sensorrauschen toleriert. Diese Koordinatentransformation (3.42) hat keinen Einfluss auf die jeweiligen zeitlichen Ableitungen, sodass $\dot{p} = \dot{\tilde{p}}$ und $\dot{e} = \dot{\tilde{e}}$ gilt.

Weiter wird der normalisierte Fehler

$$\xi = \frac{\tilde{e}}{\tilde{p}} \tag{3.43}$$

eingeführt. Mit (3.42) und (3.43) ergibt sich die Fehlertransformation (3.41) zu

$$\varepsilon(\xi) = \begin{cases} \ln\left(\frac{\xi}{1-\xi}\right), & \forall e(0) \geq 0 \\ \ln\left(\frac{1+\xi}{-\xi}\right), & \forall e(0) \leq 0 \end{cases} . \tag{3.44}$$

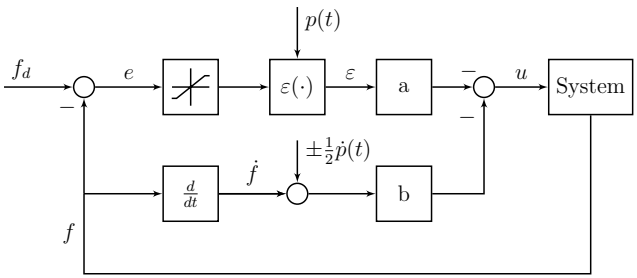


Abbildung 3.18: Blockdiagramm der PPC Reglerstruktur

Da ε für Regelabweichungen e innerhalb der Performanzgrenze p durch die Schranken $[0;1]$ für positive Regelabweichungen und $[-1;0]$ für negative Regelabweichungen beschränkt sind, bildet (3.44) eine bijektive Abbildung nach $\varepsilon : \hat{\mathbb{R}} \to \mathbb{R}$.

Für die folgenden Stabilitätsbetrachtungen sei an dieser Stelle die partielle Ableitung

$$\psi = \frac{\partial \varepsilon}{\partial \tilde{e}} = \frac{\partial \varepsilon}{\partial \xi} \frac{1}{\tilde{p}} \tag{3.45}$$

eingeführt. Der Minimalwert der Funktion ψ beträgt $\psi_{\min} = 4/\tilde{p}$. Unter der Annahme einer endlichen Leistungsfähigkeit $\tilde{p}$ des kontrollierten Roboters und der damit einhergehenden Leistungsgrenze $\tilde{p}_{\max}$, ist $\tilde{p}$ mit $\tilde{p}_{\max}$ begrenzt. In diesem Fall existiert auch eine konstante Funktion $\psi_{\min} = 4/\tilde{p}_{\max}$ als untere Schranke von ψ.

3.3.1 Reglerstruktur und Stabilität

Die Reglerstruktur wird mit der PD-Regler ähnlichen Struktur

$$u = \begin{cases} -a\varepsilon - b(\dot{f} - \frac{1}{2}\dot{p}), & \forall e(0) \geq 0 \\ -a\varepsilon - b(\dot{f} + \frac{1}{2}\dot{p}), & \forall e(0) \leq 0 \end{cases}, \tag{3.46}$$

festgelegt (Halt et al. 2019). Dabei sind a, b positive Verstärkungsfaktoren. ε und ψ entsprechen der transformierten Abweichung und der partiellen Ableitung nach Gleichungen (3.41) und (3.45). $\dot{f}$ bezeichnet die zeitliche Ableitung der gemessenen Kraftwerte und $\dot{p}$ die zeitliche Ableitung der Performanzfunktion p. Die Reglerstruktur in Abbildung 3.18 als Blockdiagramm dargestellt.

Im Folgenden wird die asymptotische Stabilität nach Ljapunov innerhalb der Performanzgrenzen nachgewiesen. Hierfür wird für die vorkommende Kontaktkräfte f eine maximal zulässige obere Grenze $f_{\max}$ angenommen. Außerdem wird eine untere Grenze $f_{\min}$ angenommen, bei der von einem Kontakt ausgegangen werden kann. Hiermit ergibt sich

$$\mathcal{F} = \{f \in \mathbb{R} \mid \ f_{\max} \geq f \geq f_{\min}\} \tag{3.47}$$

als Menge $\mathcal{F}$ der Kräfte mit den Zwangsbedingungen

$$\begin{cases} \sup(p + f_d) \leq f_{\max}\,, \ \inf(-D + f_d) \geq f_{\min}\,, & \forall e(0) \geq 0 \\ \sup(D + f_d) \leq f_{\max}\,, \ \inf(-p + f_d) \geq f_{\min}\,, & \forall e(0) \leq 0 \end{cases} \tag{3.48}$$

für die Führungsgröße f_d zur Einhaltung der Grenzen.

Die Reaktionskraft f wird als elastisch angenommen und plastische Verformungen werden vernachlässigt. Das angenommene generalisierte Kraftmodell F

$$f = F(d), \forall d \geq 0, F : [0, +\infty) \to [0, +\infty) \tag{3.49}$$

beschreibt eine positive, streng monoton steigende, kontinuierlich differenzierbare Funktion F der Verformung d. Hierin inbegriffen ist die Kontaktmechanik eines Zylinders in eine nachgiebige Oberfläche

$$F \propto d^2 \ ,$$

bei dem die Kraft F proportional zum Quadrat der Verformung d ist.

Die zeitliche Ableitung von f ergibt

$$\dot{f} = \frac{\partial F(d)}{\partial d} v(t) \tag{3.50}$$

mit der Stellgröße der kartesischen Geschwindigkeit $v(t)$, welche normal zu der berührten Oberfläche in dem durch den Roboter geführten Werkzeug steht. Für Kräfte aus der beschränkten Menge $\mathcal{F}$ ist auch die Ableitung $\frac{\partial F(d)}{\partial d}$ beschränkt und positiv.

Der Geschwindigkeitsfehler

$$e_v = v(t) - u \tag{3.51}$$

beschreibt die Abweichung der vorgegebenen Geschwindigkeit u von der wirklichen Geschwindigkeit $v(t)$. Es wird angenommen, dass der Roboter aufgrund der unterlagerten Achsregelung

Geschwindigkeitsvorgaben optimal folgt. So wird der Geschwindigkeitsfehler e_v vernachlässigbar klein und strebt gegen Null.

Nach Einsetzen und Umformen von (3.50), (3.51) und (3.46) ergibt sich

$$\dot{f} = \begin{cases} K(-a\varepsilon + \frac{b}{2}\dot{p} + e_v), & \forall e(0) \geq 0 \\ K(-a\varepsilon - \frac{b}{2}\dot{p} + e_v), & \forall e(0) \leq 0 \end{cases} \tag{3.52}$$

mit

$$K = \frac{\frac{\partial F(d)}{\partial d}}{1 + b\frac{\partial F(d)}{\partial d}} \ .$$

K ist positiv und beschränkt, da b positiv und $\frac{\partial F(d)}{\partial d}$ auf $\mathcal{F}$ positiv und beschränkt ist.

Für den Nachweis der Stabilität wird der Ljapunov Funktionskandidat

$$V = \frac{1}{2}\varepsilon^2 \tag{3.53}$$

mit der zeitlichen Ableitung

$$\dot{V} = -A\psi\varepsilon^2 + B\psi\varepsilon \tag{3.54}$$

mit $A = aK$ und $B = K(\pm\frac{b}{2}\dot{p} + e_v) - \dot{f}_d - \dot{p}\frac{\tilde{e}}{\tilde{p}}$ herangezogen.

Führen wir nun zusammen,

a) a, b sind positive frei wählbare Konstanten,

b) K ist beschränkt für $f \in \mathcal{F}$,

c) e_v strebt gegen Null,

d) $\dot{f}_d$ ist bekannt und beschränkt,

e) $\dot{\tilde{p}} = \dot{p}$ ist beschränkt für begrenzte Leistungsfähigkeit eines Roboters,

f) $\tilde{e}$ ist beschränkt.

Somit besteht A und B ausschließlich aus Termen, welche beschränkt gewählt wurden oder deren Beschränktheit nachgewiesen wurde. Daher ist es möglich, positive Konstanten λ_A, λ_B zur Abschätzung von A und B zu finden,

$$A \geq \lambda_A > 0 \ , \tag{3.55}$$

$$\lambda_B \geq |B| \ . \tag{3.56}$$

Hiermit kann Gleichung (3.54) mit

$$\dot{V} \leq -\lambda_A \psi |\varepsilon|^2 + \lambda_B \psi |\varepsilon| \tag{3.57}$$

abgeschätzt werden. Die Abschätzungen λ_A, λ_B unterstützen bei der folgenden Untersuchung der Stabilitätsgrenzen.

Mit $\psi > \psi_{\min} > 0$ ist aus Gleichung (3.57) zu beobachteten,

$$\begin{cases} \dot{V} \leq 0\,, \ \forall\, |\varepsilon| \in \left\{ \frac{\lambda_B}{\lambda_A}, 0 \right\} \ , \\ \dot{V} < 0\,, \ \forall\, |\varepsilon| > \frac{\lambda_B}{\lambda_A} \ , \end{cases} \tag{3.58}$$

dass für $\varepsilon > \frac{\lambda_B}{\lambda_A}$ *asymptotische Stabilität* und für $0 \leq \varepsilon \leq \frac{\lambda_B}{\lambda_A}$ *Ljapunov Stabilität* nachgewiesen ist.

Damit ist das geregelte System mit dem Regler aus (3.46) für alle f innerhalb der Menge $\mathcal{F}$ mindestens stabil im Sinne von Ljapunov. □

Für den Beweis, dass f mit einem Startwert innerhalb der Menge $\mathcal{F}$ diese Menge niemals verlassen wird, wendet (Bechlioulis et al. 2010) einen Widerspruchsbeweis an.

Zunächst wird aus (3.58) festgehalten, dass ε für Werte außerhalb der Grenze $\frac{\lambda_B}{\lambda_A}$ streng monoton fällt. Für Werte zwischen $\frac{\lambda_B}{\lambda_A}$ und 0 können die Werte fallen oder gleich bleiben.

Wird ein initialer Wert $\varepsilon(0)$ größer als die Grenze $\frac{\lambda_B}{\lambda_A}$ angenommen, so ist dieser Wert zwingend der Maximalwert, da dieser im weiteren zeitlichen Verlauf fallen wird. Liegt der initiale Wert auf oder innerhalb der Grenze $\frac{\lambda_B}{\lambda_A}$, so kann ausgeschlossen werden, dass diese Schranke überschritten wird. Es gilt also,

$$|\varepsilon| \leq \max \left\{ \varepsilon(0), \frac{\lambda_B}{\lambda_A} \right\}, \forall t > 0\,, \ \forall f \in \mathcal{F} \ . \tag{3.59}$$

Der maximale Wert für $|\varepsilon|$ ist also beschränkt.

Für den Fall $e(0) \geq 0$ wird nun angenommen, dass zu einem Zeitpunkt t^* die Regelabweichung $e(t^*)$ exakt auf einer Performanzgrenze liegt.

Für die obere Grenze $e(t^*) = p(t^*)$ ergibt sich die Kraft mit (3.38) zu

$$f(t^*) = p(t^*) + f_d \ . \tag{3.60}$$

Wenn die Kraft zu einem vorherigen Zeitpunkt $f(t^{*-})$ innerhalb der Performanzgrenzen lag und sich in Richtung der Grenze p entwickelt hat, so war die Kraft zu diesem Zeitpunkt

$$f(t^{*-}) \leq p(t^{*-}) + f_d \ . \tag{3.61}$$

Für die kontinuierliche Funktion (3.61) gilt der Grenzübergang

$$\lim_{t \to t^{*-}} f(t) = p(t^{*-}) + f_d \ . \tag{3.62}$$

Mit (3.41) ergibt sich allerdings

$$\lim_{t \to t^{*-}} \epsilon(t) = +\infty \ , \tag{3.63}$$

was den Feststellungen aus (3.59) widerspricht, dass $|\varepsilon|$ begrenzt ist.

Für die analoge Betrachtung der unteren Grenze $e(t^*) = -D$, ergibt sich ebenfalls der zu Abschätzung (3.59) widersprüchliche Grenzwert von

$$\lim_{t \to t^{*-}} \epsilon(t) = -\infty \ .$$

Für den Fall $e(0) \leq (0)$ ergeben sich entsprechende analoge Widersprüche.

Damit ist der Nachweis durch Widerspruch erbracht und es kann festgehalten werden, dass für ein kontinuierliches System die Regelabweichung $e(t)$ die Menge $\mathcal{F}$ nicht verlassen wird. □

3.3.2 Implementierung

Nachdem im vorherigen Kapitel die Reglerstruktur (3.46) vorgestellt und die Stabilität nachgewiesen wurde, wird im Folgenden weiter auf die Umsetzung eingegangen.

Im Gegensatz zu (Kanakis et al. 2018) werden Transformationen in den zu regelnden Kraftraum durch die iTaSC Implementierung übernommen. Dadurch vereinfacht sich die zu implementierende Reglerstruktur, wobei die funktionalen Komponenten des PPC weitestgehend übernommen werden können.

Für den realistischen Fall müssen Vorkehrungen und Anpassungen vorgenommen werden, damit keine numerischen Probleme auftreten und das vorgegebene Performanzband stets den Regelfehler e einschließt. Die Fehlertransformation ε nimmt nahe den Definitionsgrenzen betragsmäßige sehr große Werte an, welche zu numerischen Problemen führen können. Ebenso

ist die Fehlertransformation nicht mehr definiert, sollte sich der anfängliche Fehler $e(0)$ nicht innerhalb der Performanzgrenzen befinden. Zum anderen kann nicht garantiert werden, dass der Roboter überhaupt in der Lage ist die Performanzgrenzen einzuhalten. Für den Stabilitätsnachweis wird davon ausgegangen, dass dem Roboter genug Leistung zur Verfügung steht, um dem vorgegebenem Performanzbands zu folgen. Fällt jedoch die Performanzgrenze schneller als der Roboter durch seine Leistungsbegrenzung folgt, kann der Verlauf des Fehlers das Performanzband nicht einhalten. Der Regelfehler ist weiterhin abklingend, jedoch ist die Fehlertransformation in diesem Fall nicht mehr definiert.

Um dieser Fehlerquelle vorzubeugen, wurden die folgenden Maßnahmen zugefügt: (i) eine Sättigung des Eingangs der Fehlertransformation, (ii) eine zeitliche Verschiebung τ der Performanzfunktion $p(t-\tau)$, sollte die unverschobene Performanzgrenze nicht eingehalten werden können und (iii) eine Erhöhung des Verstärkungsfaktors a, wenn p während des Einschwingvorgangs verschoben und (iv) eine Erniedrigung des Verstärkungsfaktors a, wenn p innerhalb des stationären Bandes $\pm D$ verletzt wurde.

Auf die Maßnahmen wird im Folgenden einzeln eingegangen. Die Maßnahmen ii) und iii) haben keinen Einfluss auf den Beweis aus Kapitel 3.3.1 und somit auf die Stabilität der Regelung.

Um den Einfluss der Sättigung aus i) auf die Stabilität zu analysieren wird, die zeitliche Ableitung der Regelabweichung e betrachtet. Für die konstante Führungsgröße f_d ergibt sich aus Gleichung (3.38) die zu Gleichung (3.50) identische Ableitung

$$\dot{e} = \begin{cases} K(-a\varepsilon + \frac{b}{2}\dot{p} + e_v)\,, & \forall e(0) \geq 0 \\ K(-a\varepsilon - \frac{b}{2}\dot{p} + e_v)\,, & \forall e(0) \leq 0 \end{cases} . \tag{3.64}$$

In Sättigung kann angenommen werden, dass der Wert der Fehlertransformation ε groß gegenüber den Werten der abgeleiteten Performanzfunktion $\dot{p}$ und des Geschwindigkeitsfehlers e_v ist. Es gilt die Abschätzung

$$|a\varepsilon| \gg |\tfrac{b}{2}\dot{p}| + |e_v| \ . \tag{3.65}$$

In Gleichung (3.64) sind alle Werte außer ε bekannt oder beschränkt und K ist positiv. Daher kann festgestellt werden, dass

1. $\dot{e} < 0$ für Fehler e, welche gegen die obere Performanzgrenze streben und dass
2. $\dot{e} > 0$ für Fehler e, welche gegen die untere Performanzgrenze streben.

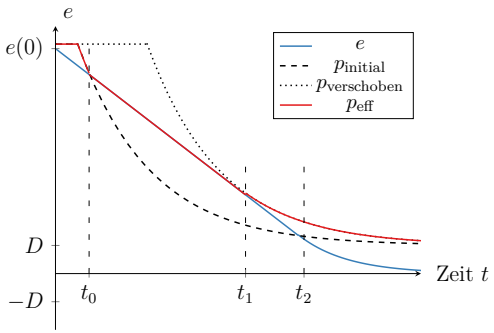


Abbildung 3.19: Darstellung der zeitlichen Verschiebung der Performanzfunktion zur Einhaltung der Performanzgrenzen. Zur besseren Lesbarkeit wurde die untere Grenze nicht eingezeichnet.

Durch die Wahl der Performanzgrenzen P in (3.39) ist sichergestellt, dass Änderung der Grenzen gegen Null streben, da

$$\lim_{t\to\infty} \dot{p} \to 0 \,\text{und}\; \dot{D} = 0$$

gilt.

Dadurch ist garantiert, dass der Fehler des geregelten Systems mit Fehlersättigung in die Performanzgrenzen getrieben wird. □

Wird die Performanzgrenze p überschritten, kann der PPC in der zuvor hergeleiteten Form nicht eingesetzt werden. Die Einhaltung des Performanzbands P ist eine fundamentale Voraussetzung. In einem kontinuierlichen System ist nachgewiesen, dass die Regelung jede verfügbare Leistung einsetzt, um das Performanzband einzuhalten. In einem zeitdiskreten System ist es allerdings möglich, dass der Regelfehler zum nächsten Abtastzeitpunkt das Performanzband P aufgrund von zwischenzeitlichen Störungen verlassen hat.

Um hiermit umgehen zu können, wird die Performanzfunktion p bei Verletzung der Grenzen online angepasst (Maßnahme ii)). Mit dem aktuellen (gesättigten) Fehler $e(\tau)$ wird eine neue Performanzfunktion $\tilde{p}$ so festgelegt, dass der aktuelle Wert $e(\tau)$ auf der Performanzgrenze $\tilde{p}(\tau) = e(\tau)$ liegt. Dies kommt einer zeitlichen Verschiebung der Performanzfunktion gleich.

Die Abbildung 3.19 illustriert einen beispielhaften Verlauf eines Regelfehlers e und Performanzgrenzen p. In diesem konstruierten Fall wurde die Abklingrate des Fehlers stark begrenzt, weshalb die Performanzfunktion p zu einem Zeitpunkt t_0 verletzt wird. Die Performanzfunktion wird in jedem Zeitschritt verschoben, bis zum Zeitpunkt t_1 an dem die Grenzen wieder eingehalten werden können. Ab dem Zeitpunkt t_2 wird die ursprüngliche Performanzfunktion

wieder eingehalten. Es ergibt sich eine stetige, stückweise differenzierbare Performanzfunktion p_{eff}.

Die Performanzgrenzen werden durch eine zu langsame Dynamik des geregelten Systems verletzt. Um die Regelung mit PPC trotzdem zu ermöglichen, muss ein neues Performanzband angenommen werden, welches den aktuellen Fehler einschließt.

Es ist allerdings möglich, dass durch zu konservative Einstellung der Regelungsparameter das erforderliche Verhalten nicht eingehalten wird. Dieser Effekt äußert sich auch, wenn durch eine geringere Regelungstaktrate die Stellgröße nicht ausreichend oft aktualisiert wird. In diesem Fällen kann auf die eingeregelte Dynamik mit den Parametern a und b Einfluss genommen werden.

Da Parameter b durch die Ableitung $\dot{f}$ sensibel auf Sensorrauschen reagiert, sollte dieser Parameter im Hinblick auf die Qualität der Messung gewählt werden.

Mit Maßnahme iii) wird bei einer Verletzung der Performanzgrenzen zusätzlich zu der Verschiebung der Performanzfunktion ebenfalls der Regelungsparameter a erhöht. Klingt die Regelabweichung durch den erhöhten Wert a schneller ab, so können die ursprünglichen Performanzgrenzen enger eingehalten werden. Befindet sich der Fehler e allerdings im Bereich des stationären Bands $\pm D$, so wurde die Verletzung der Performanzgrenzen möglicherweise durch eine zu starke Reaktion auf stationäres Rauschen verursacht. Um dies zu vermeiden, ergibt sich durch Maßnahme iv) eine Verringerung des Verstärkungsfaktors a. Durch die stetige Wiederholung der Anpassung iii) und iv) kann die Änderungsrate von a konservativ gewählt werden.

Mit den Maßnahmen ist es möglich die PPC Regelung auch unter fehlerhafter Einstellung des Performanzbands oder den Parametern a und b zu betreiben. Bei Auslösen der Maßnahmen wird der Bediener informiert und kann entsprechende Anpassungen an den Einstellungen vornehmen, welche es erlauben den PPC Regler in dem vorgesehenen Performanzband zu betreiben. In den weiteren Untersuchungen sollen nun Erkenntnisse gesammelt werden, wie sich der hergeleitete zeitkontinuierliche PPC Regler auf einem zeitdiskreten und gestörten System verhält.

3.3.3 Simulationsergebnisse

In dem folgenden Kapitel wird, ähnlich wie bei dem vorangehenden Kapitel 3.2.2, die beschriebene PPC Struktur auf ein simuliertes System angewendet.

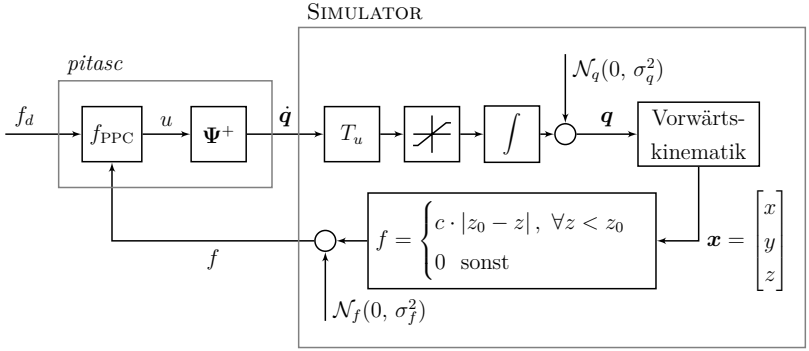


Abbildung 3.20: Blockdiagramm der Simulation

Die an der Simulation untersuchten Situationen umfassen i) den Einfluss von Verzugszeiten, ii) der Einfluss von Messrauschen, iii) das Verhalten bei verschiedenen Umgebungsnachgiebigkeiten c, iv) den Einfluss der Reglertaktung und v) der Einfluss bei zu konservativ eingestellten a bzw. zu langsamer Dynamik.

Im Gegensatz zu den bisherigen Simulationen anhand von Differenzialgleichungen wird der PPC in das gesamte *pitasc* Rahmenwerk eingesetzt und lediglich der geregelte Roboter simuliert.

Die Robotersimulation empfängt Achswinkelgeschwindigkeiten als Stellgröße, integriert und ggf. verzögert diese bevor die resultierenden Achswinkelstellungen an eine Vorwärtskinematik übergeben werden. Die mit der Vorwärtskinematik ermittelten Posen des Roboters dienen als Messungen im kartesischen Aufgabenraum. Als kinematisches Modell des simulierten Roboters wird ein UR5 der Firma Universal Robots angenommen.

Zusätzlich wird die z Koordinate der Endeffektor Position überwacht. Unterschreitet diese eine virtuelle Oberfläche z_0, so werden mit einer definierten Nachgiebigkeit c virtuelle Kontaktkräfte errechnet. Verformungen des Roboters oder Rückwirkung der virtuellen Kraft auf die kartesische Position des Roboterwerkzeugs werden nicht simuliert. Das Blockdiagramm der für die Simulation relevanten Teile ist in Abbildung 3.20 dargestellt.

Jede Simulation initialisiert den Roboter in einer Ausgangsstellung der Achsen. Der Roboter wird danach mit einer konstanten kartesischen Geschwindigkeit in Richtung der virtuellen Oberfläche z_0 bewegt, bis eine Kraft größer als 2 N detektiert wird. Für die folgende Ausregelung der Kontaktkraft auf 45 N wird der zu untersuchende PPC Regler mit einem entsprechenden Skill aktiviert. Die dargestellten Verläufe beginnen mit Aktivierung des Skills zur Ausregelung der Kontaktkraft. Wenn nicht anders erwähnt, wird die Simulation mit einer Taktung von

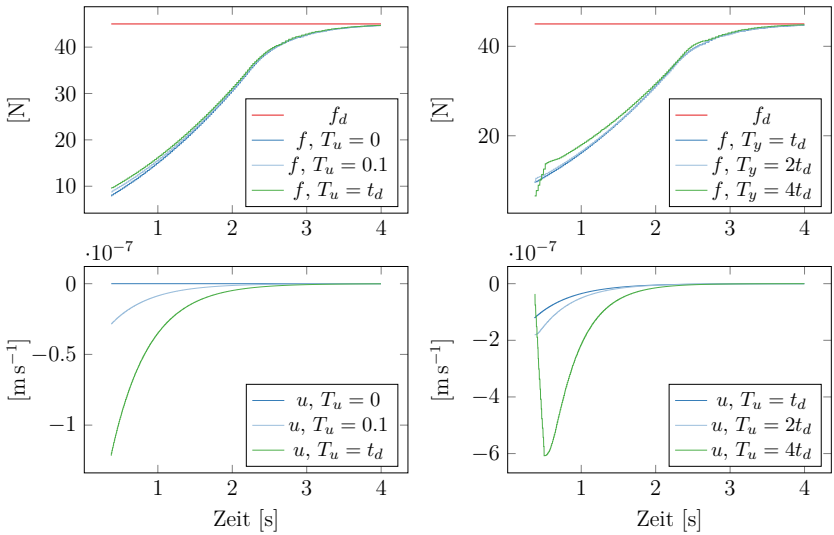


Abbildung 3.21: Variation des Verzug T_u

125 Hz durchgeführt. Für die Parameter des PPC werden $a = 0.0002$, $b = 0.00002$ und $D = 3$ gewählt.

In Abbildung 3.21 sind die Soll- und Stellgrößenverläufe für verschiedene Verzugszeiten T_u dargestellt. Es wird kein Rauschen beaufschlagt, d. h. $\sigma_q^2 = 0$ und $\sigma_f^2 = 0$. Die Steifigkeit ist gering mit $c = 5000\,\mathrm{N\,m^{-1}}$. Es ist zu erkennen, dass das geregelte System sich sehr robust im Umgang mit Verzugszeiten verhält. Die verzögerten Systemantworten werden durch den PPC in das Zentrum des Performanzbands getrieben.

Für die weiteren Experimente wird die Verzugszeit zu festgelegt $T_u = t_d = 0.033\,\mathrm{s}$. Die Steifigkeit bleibt konstant auf $c = 5000\,\mathrm{N\,m^{-1}}$. Das roboterseitige Achsrauschen wird weiterhin vernachlässigt, $\sigma_q^2 = 0$. In Abbildung 3.22 ist zu erkennen, dass sich mit stärkeren Messrauschen die Einregelzeit vergrößert. Für ein Rauschen mit $\sigma_f^2 < 5\,\mathrm{N}^2$ ergeben sich ständige Verletzung der Performanzgrenzen um den stationären Wert. Da das Rauschen nicht mehr in das lineare Performanzband $\pm D$ eingeschlossen ist, sind streng genommen die Voraussetzungen für einen stabilen PPC nicht gegeben. Durch die Verschiebung der Performanzfunktion kann trotzdem stabiles Verhalten erreicht werden.

Weitere Unsicherheit wird in Abbildung 3.23 eingeführt. Zusätzlich zu der Verzugszeit $T_u = t_d$ und dem Sensorrauschen $\mathcal{N}_f(0, 0.5)$ werden die achsseitigen Stellgrößen mit weißen Rauschen beaufschlagt. Die Auswirkung von Rauschen der Achspositionen auf die kartesische Pose des

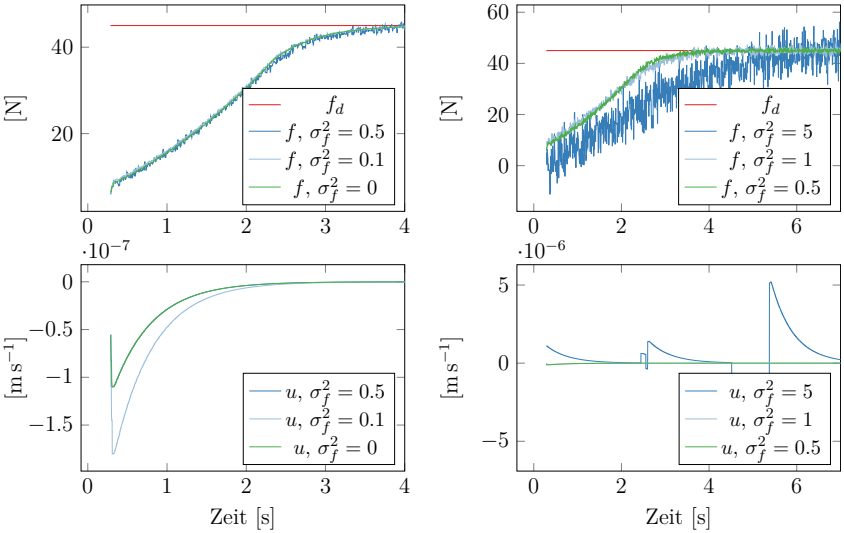


Abbildung 3.22: Variation des Sensorrauschens $\mathcal{N}_f(0, \sigma_f^2)$

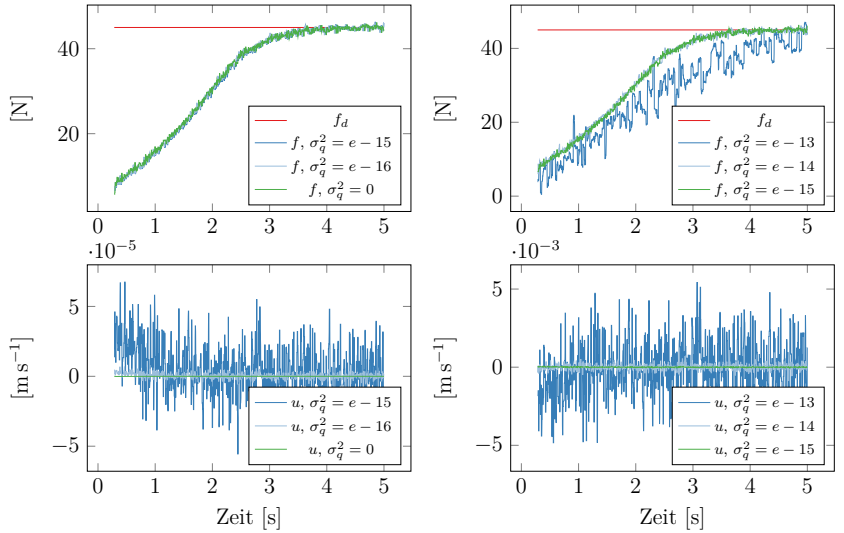


Abbildung 3.23: Variation des roboterseitige Achsrauschens $\mathcal{N}_q(0, \sigma_q^2)$

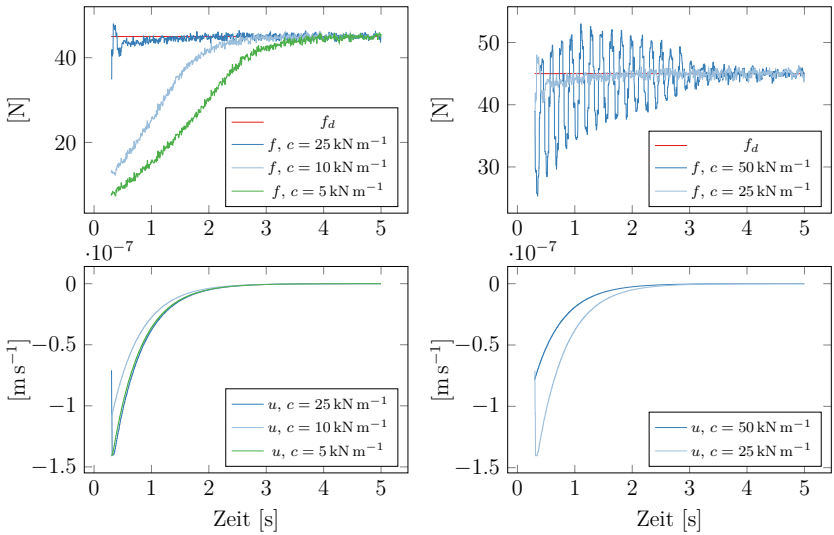


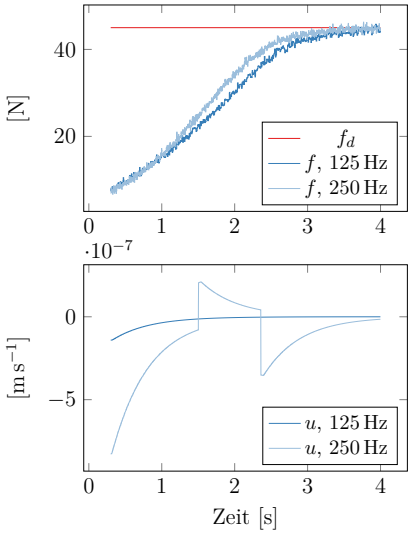
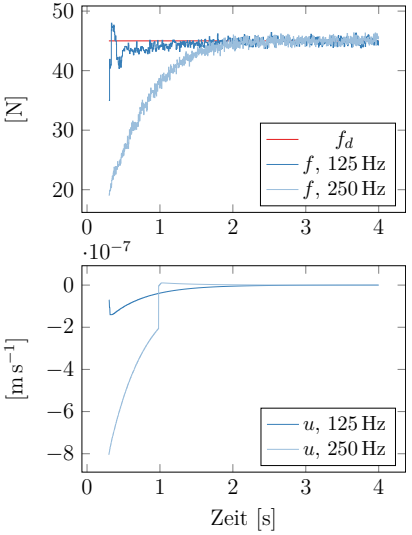
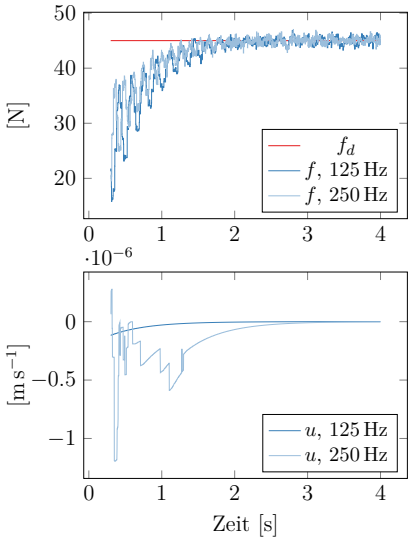
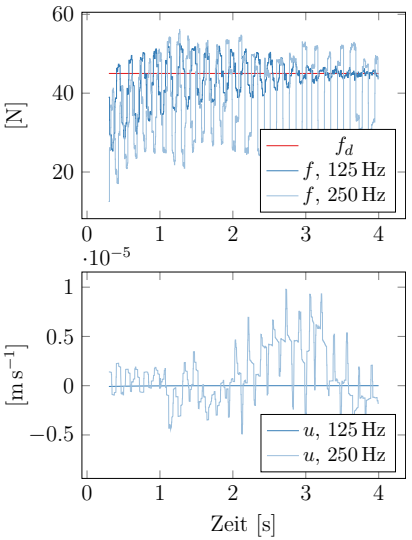
Abbildung 3.24: Variation der Umgebungssteifigkeit c

Roboters ergibt sich im allgemeinen Fall durch Überlagerung. Ein Rauschen von $\mathcal{N}_q(0, 10^{-3})$ ist deutlich im Stillstand des simulierten Roboters als Vibration visuell wahrnehmbar. In Abbildung 3.23 kann allerdings festgestellt werden, dass der PPC sich ebenfalls robust gegenüber diesem Rauschen verhält.

In Abbildung 3.24 werden die Simulationsparameter $T_u = t_d$, $\mathcal{N}_f(0, 0.5)$, $\mathcal{N}_q(0, 10^{-6})$ konstant gehalten. Lediglich die Steifigkeit c wird variiert. Mit einer höheren Steifigkeit ergeben sich früher höhere Kraftmesswerte, da weiterhin die Reglertaktrate auf 125 Hz (8 ms) festgelegt ist. Durch die Regelverstärkung kann so eine Schwingung oder ein Verlust des Kontakts auftreten. Es ist zu beobachteten, dass bei höherer Steifigkeit c die Führungsgröße schneller erreicht wird. Ab einer kritischen Steifigkeit hier ca. bei $c > 45\,000\,\mathrm{N\,m^{-1}}$ kann das Performanzband um den stationären Zustand von $\pm D$ nicht mehr eingehalten werden. Es ergibt sich eine Dauerschwingung um den stationären Wert. Ein Steifigkeitswert von $c > 45\,000\,\mathrm{N\,m^{-1}}$ entspricht einem harten Kontakt, bei dem optisch bei manueller Montage keine Verformung sichtbar wäre. In diesem Steifigkeitsbereich nimmt die Nachgiebigkeit der Roboterstruktur einen zunehmenden Einfluss und eine elastische Verformung des Roboterarms kann nicht mehr vernachlässigt werden.

In Tabelle 3.5 werden Simulationen mit den Reglertaktungen 125 Hz und 250 Hz sowie verschiedenen Steifigkeiten c aufgezeigt. Interessanterweise ist zu erkennen, dass durch eine niedrigere

Tabelle 3.5: Variation der Reglertaktung bei verschiedenen Umgebungssteifigkeiten c

$c = 5000\,\mathrm{N\,m^{-1}}$	$c = 25\,000\,\mathrm{N\,m^{-1}}$
	
$c = 45\,000\,\mathrm{N\,m^{-1}}$	$c = 50\,000\,\mathrm{N\,m^{-1}}$
	

Taktung die Führungsgröße schneller erreicht werden kann. Dieser Effekt entsteht, da durch die niedrigere Taktung die Performanzgrenzen früher verletzt werden und der Faktors a erhöht wird. Der erhöhte Faktor bedingt ein insgesamt schnelleres Einregelverhalten.

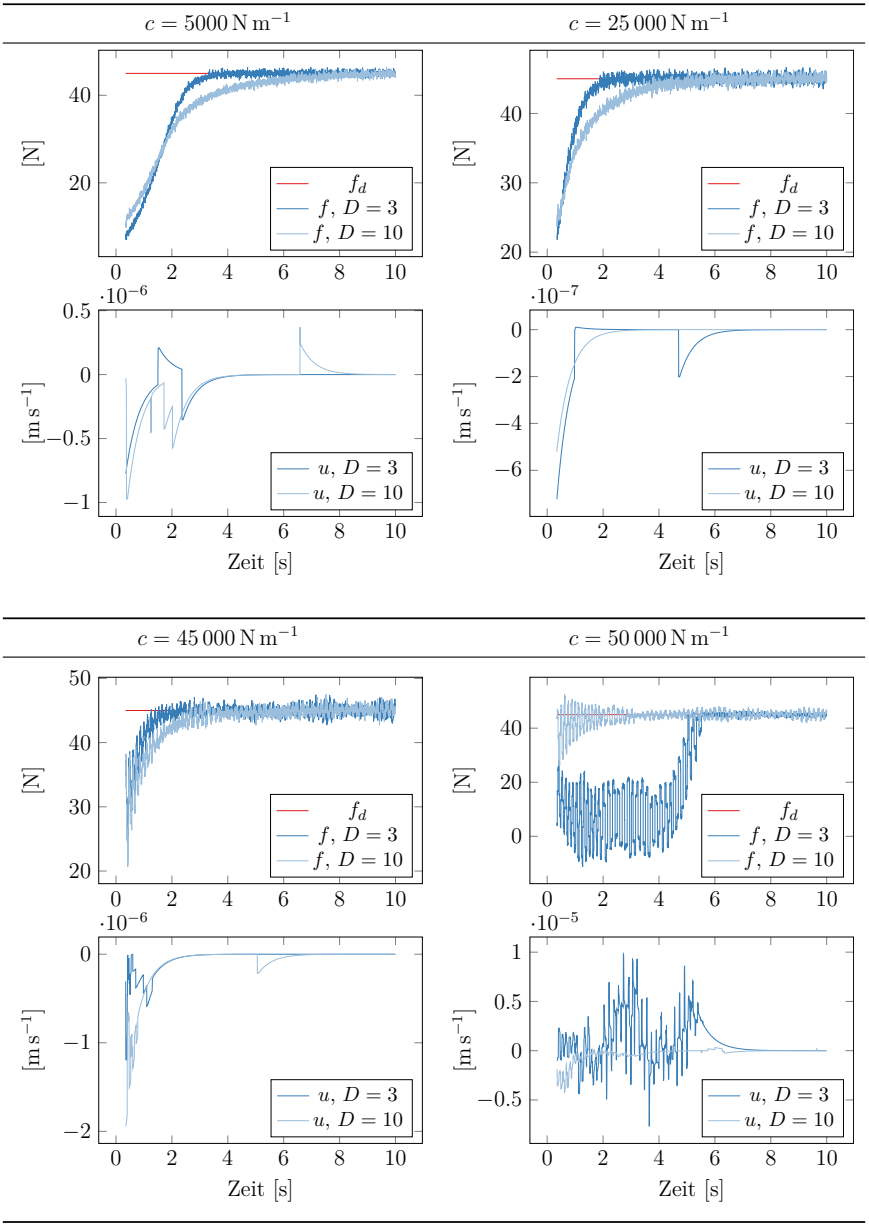
Anhand von Tabelle 3.6 zeigt sich, dass eine zu enge Wahl des stationären Bandes D sich negativ für sehr steife Umgebungen auswirken kann. Mit der Wahl von $D = 3$ kann kein stabiles Verhalten für $c > 45\,000\,\mathrm{N\,m^{-1}}$ garantiert werden. Hingegen zeigen sich keine Schwierigkeiten bei der deutlich größeren Wahl von $D = 10$. Trotzdem beeinflusst eine größere Wahl von D den zeitlichen Verlauf der Regelantwort und bedingt ein langsameres Einregelverhalten.

Ziel der Untersuchung der PPC Regelung anhand der Simulationen war es das zeitkontinuierliche Regelungskonzept auf die Robustheit gegenüber praktisch auftretenden Störungen durch Verzugszeiten und zeitdiskreter Taktung zu prüfen. Es zeigt sich, dass der PPC Regler mit den absichernden Maßnahmen i) bis iv) robust gegenüber Verzugszeiten $T_u \leq 4t_d = 132\,\mathrm{ms}$ ist und somit das Vierfache der typischen Verzögerungszeiten t_d von 3-4 Taktzyklen abdeckt. Bessere Ergebnissen bei höhere Taktung waren zu erwarten, da das PPC Konzept auf einem zeitkontinuierlichen Ansatz beruht. In den Versuchen zeigt sich allerdings, dass der Betrieb auch mit 125 Hz möglich ist. Bei großen Steifigkeiten für $c > 45\,000\,\mathrm{N\,m^{-1}}$ sind Störungseffekte durch die Taktung zu erkennen. Daher sollte für steife Roboter und harte Kontakte mit niedrigeren Kontaktgeschwindigkeiten oder hohen Taktungen verfahren werden.

Wird das stationäre Performanzband D durch Sensorrauschen oder Quantisierungseffekte gestört, ergeben sich ungewollte Nachregeleffekte. Die Wahl von D sollte somit das zu erwartende Rauschen einschließen.

Da sich die zuvor untersuchten Regelungskonzepte unter den anzunehmenden Herausforderungen und Störungen qualifiziert haben, muss im Folgenden experimentell belegt werden, dass Regelungen auch unter realen Bedingungen einsetzbar sind. Störungen, wie elastische Verformung des Roboterarms, Nichtlinearitäten und Sättigungen der realen Maschinen sowie nichtdeterministische Kommunikationszeiten müssen im praktischen Betrieb bewältigt werden und machen eine experimentelle Erprobung und die Bewertung der Verfahren gegenüber dem Stand der Technik notwendig.

Tabelle 3.6: Variation des stationären Bandes D bei verschiedenen Umgebungssteifigkeiten c und einer Reglertaktrate von 250 Hz



4 Experimentelle Untersuchung

In diesem Kapitel werden die zuvor anhand von Simulationen erprobten Ansätze an realen Robotern untersucht. Hierfür wurden repräsentative, auf die grundlegende Funktionalität reduzierte Aufgaben ausgewählt, welche sich in typischen Produktionsaufgaben finden. Zunächst werden die einzelnen Ansätze möglichst isoliert betrachtet und darauf folgend sukzessiv kombiniert und integriert.

In diesen Untersuchungen wird gezeigt, wie die zuvor ausgearbeiteten Regelungsansätze im Kontext eines Robotersystems eingesetzt werden können und inwieweit die Ansätze zu einer Leistungssteigerung und Verbesserung der Übertragbarkeit der Programme zwischen verschiedenen Robotern beitragen.

Konkret wird experimentell gezeigt, dass eine synthetisierte Ausgangsrückführung (ARF) auch unter realen Bedingungen das gewünschte Verhalten zeigt, das mithilfe eines identifizierenden erweiterten Kalman Filters ein geeignetes Modell angenähert werden kann und dass die Variation des Reglers mit vorgegebener Performanz selbst mit einem Performanzband betrieben werden kann, welches nicht für die jeweiligen Robotersysteme angepasst wurde.

4.1 Aufbau der Experimente

Zunächst wird auf die eingesetzten Geräte, deren Kommunikation und den mechanischen Aufbau eingegangen, um die Rahmenbedingungen der Experimente darzustellen.

4.1.1 Eingesetzte Roboter und Sensoren

Als ausführende Roboter wurden drei verschiedene Manipulatoren mit jeweils verschiedenen Eigenschaften ausgewählt. Alle Roboter fallen in eine niedrige Traglastklasse und sind typisch für das Anwendungsgebiet der automatisierten Montage. Für Kraftregelungsaufgaben wurden die Roboterarme mit verschiedenen 6D-Kraft-Momenten-Sensoren[6] (KMS) ausgestattet.

[6] 6D Kraft-Momenten-Sensoren messen die generalisierten Kräfte, d. h. die Kräfte und die Momente in, bzw. um alle drei Raumrichtungen.

Abbildung 4.1: Bilder der Roboter: Panda, Denso VS087, UR5

- **Panda** der Firma Franka Emika GmbH, München ist ein 7-achsiger Roboterarm (Abbildung 4.1 links). Der Roboter ist mit Momentensensoren in allen Achsen ausgestattet. Mit diesen Messdaten berechnet die Robotersteuerung eine resultierende kartesische Kraft am Roboterwerkzeug. Ein zusätzliches Softwarespaket erlaubt die Programmierung von kraftgeregelten Aufgaben. Die Traglast beträgt 3 kg bei einer maximalen Reichweite von 855 mm. Als maximal zulässige kartesische Geschwindigkeit werden $2\,\mathrm{m\,s^{-1}}$ angegeben (Franka Emika 2019b). Eine für Forschung vorgesehene Kommunikationsschnittstelle erlaubt einen Zugriff auf die internen Messungen und Stellgrößen mit bis zu 1 kHz.

- **Denso** VS087 der Firma Denso Wave, Japan ist ein 6-achsiger Roboterarm (Abbildung 4.1 mittig). Der Roboter erlaubt eine Traglast von 7 kg mit einer maximalen Reichweite von 905 mm. Dieser Roboter ist mit einer angegebenen maximalen kartesischen Geschwindigkeit von ca. $11\,\mathrm{m\,s^{-1}}$ sehr schnell und bietet eine hohe effektive Steifigkeit, da er, im Gegensatz zu den anderen aufgeführten Robotermodellen, nicht für den kollaborativen Einsatz vorgesehen. Der Roboter erlaubt einen Zugriff auf die internen Messungen und Stellgrößen mit einer Frequenz von 125 Hz. An den Roboterflansch wurde ein Kraft-Momenten-Sensor *Dyn pick WEF-6A200* der Firma Wacoh-Tech Inc., Japan angebracht. Der Sensor deckt einen Messbereich von ±200 N und ±4 N m mit einer maximalen Messunsicherheit von 6 N bzw. 0.12 N m ab (Wacoh Tech 2019). Der Kraftsensor liefert Messdaten mit einer Frequenz von bis zu 1 kHz.

- **UR5** der Firma Universal Robots A/S, Dänemark ist ein 6-achsiger Roboterarm (Abbildung 4.1 rechts). Der Roboter gilt als Pionier der *kollaborativen Roboter*. Der Leichtbauarm trägt eine Last von 5 kg mit einer maximalen Reichweite von 850 mm. Als typische kartesische Geschwindigkeit werden $1\,\mathrm{m\,s^{-1}}$ angegeben (Universal-Robots 2019). Der Roboter erlaubt mit dem Kommunikationsprotokoll *RTDE* Zugriff auf die internen Messungen und Stellgrößen mit 125 Hz. An dem Roboterflansch wurde ein Kraft-Momenten-Sensor *KMS40* der Firma Weiss Robotics angebracht. Der Sensor bedient

einen Messbereich von ±120 N und ±3 N m mit einer effektiven Auflösung von 0.005 N, bzw. 0.001 N m (Weiss Robotics 2019). Der Kraftsensor liefert Messdaten mit einer Frequenz von 500 Hz.

4.1.2 Software Umgebung

Wie bereits zuvor erwähnt werden alle Software Komponenten durch das Roboterbetriebssystem ROS verbunden. Jedes Programm, welche auf ROS Kommunikationsschnittstellen zurückgreift, wird als ROS-Knoten bezeichnet.

Die Messdaten der Roboterachsen werden durch die jeweilige Steuerung an kommunikationsspezifische ROS-Knoten (Treiber) gesendet. Ebenso werden die Verbindungen zu den Sensoren hergestellt. Die Treiber öffnen die Kommunikation mit den Geräten und stellen die empfangenen Daten als ROS Nachrichten (*rostopics*) bereit. Für die Ansteuerung der Roboterachsen werden ebenfalls rostopics durch Treiber angeboten. Die hier anliegenden Daten werden durch den Treiber aufgegriffen und an die Robotersteuerung weitergeleitet. In den meisten Fällen sind sowohl Messdatenausgang als auch Steuerungseingang in demselben Treiber realisiert.

Die interne Realisierung des Treibers und der Kommunikation zwischen Treiber und Roboter sind je Robotertyp und Roboterhersteller unterschiedlich. Die für *pitasc* nötige Stellgröße von Achsgeschwindigkeiten ist zwar untypisch, allerdings in den meisten Fällen durch die Robotersteuerung verfügbar. Häufig finden sich auch Schnittstellen zur taktgebundenen relativen Achspositionskorrektur. In diesem Fall ergibt sich die Achsgeschwindigkeitsvorgabe aus dem Quotient der gegebenen Achspositionskorrektur und der eingestellten Taktrate, solange die neue Achsposition innerhalb eines Taktes erreicht werden kann. Werden die Sollachswinkel nicht erreicht, wird die Roboteransteuerung mit einer Fehlermeldung beendet. Bei taktbasierter absoluter Sollvorgabe kann die der Sollgeschwindigkeit entsprechende Sollentfernung (Winkelabstand) für den nächsten Takt errechnet werden. Durch Addition der Sollentfernung auf die aktuelle Achsstellung des Roboters gelingt eine Geschwindigkeitsvorgabe. Bei absoluten Achswinkelvorgaben an den Roboter ist immer eine Integration der Geschwindigkeitsvorgaben zur direkten Anwendung des iTaSC Formalismus notwendig, welche wiederum in der roboterinternen Regelung aufgelöst werden. Dadurch entstehen i.A. ungünstige dynamische Effekte.

Das ausführende Gesamtsystem *pitasc* bildet einen einzelnen ROS-Knoten mit diversen ein- und ausgehenden rostopics. Innerhalb von *pitasc* werden zu diskreten Zeitpunkten, die Signale von anliegenden rostopics einsynchronisiert. Die übergeordnete Zustandsmaschine wird aktualisiert und bei einem Zustandsübergang die Konfiguration des iTaSC Algorithmus geändert. Die Konfiguration des iTaSC Algorithmus bestimmt den aktuellen Aufgabenraum, in dem die

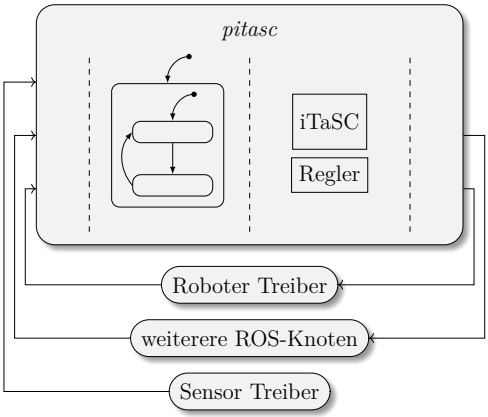


Abbildung 4.2: Darstellung der internen Struktur von *pitasc* und die ROS Kommunikation zu Roboter, Sensor und weiteren Services

relevanten Messdaten transformiert werden. Die Regelungsschleife wird innerhalb des aktiven Aufgabenraums ausgeführt und der Regelungsausgang auf rostopics der Treibereingänge gelegt.

Die ausgeführte Roboteraufgabe ergibt sich aus einer Beschreibung des Zustandsdiagramms mit spezifischen Einstellungen der iTaSC Schleife in jedem Zustand. Die Zuordnung der Eingangs- und Ausgangssignale wird weitestgehend automatisch aus der Aufgabenbeschreibung gefolgert. Weitere anlagen- oder roboterspezifische Einstellungen werden neben der Beschreibung der Roboteraufgabe durch den Programmierer angegeben.

Die jeweiligen Roboteraufgaben werden in der *pitasc* DSL beschrieben. Die Beschreibungssprache enthält alle Felder, welche für die Ausführung von *pitasc* notwendig sind. Die DSL erlaubt an Skills beliebige weitere Funktionen anzuhängen oder vordefinierte Spezifikationen zu ersetzen. So können die vorausgewählten Regelungsalgorithmen mit den zu erprobenden Reglern ersetzt werden. Die Regelungsalgorithmen sind in entsprechenden Python Klassen implementiert und in der *pitasc* DSL modelliert.

Die *pitasc* Ausführung wird mit einer eigenen Frequenz durchgeführt. Jeden Zyklus werden die aktuell anliegenden Daten aufgenommen, die jeweilige Aufgabenraumtransformation durchgeführt und die zugehörigen Komponenten, insbesondere die Regler ausgeführt. Die Stellgröße wird an den Roboter-Treiber weitergegeben.

Zu startende ROS-Knoten werden in einer `roslaunch`-Datei (XML Syntax) aufgeführt. Jede Ausführung eines Experimentes wird durch die entsprechende roslaunch Datei in Kombination mit einer *pitasc* Datei vollständig beschrieben.

Eine grafische Darstellung der ROS-Knoten mit denen sich ergebenden rostopic Verbindungen ist in Abbildung 4.2 gegeben. Für die Verdeutlichung der Experimente wird die Darstellung insofern vereinfacht, als dass das entsprechende Zustandsdiagramm mit jeweils angedeutetem Datenaustausch mit relevanten ROS-Knoten dargestellt wird. Auf die Darstellung der Roboter- und Sensorenknoten und deren Datenströme wird verzichtet, da diese Verbindungen für alle Bewegungen vorausgesetzt werden. Die Zustandsmaschine ist in dieser Darstellung symbolisch dargestellt. In den folgenden Experimenten wird die Zustandsmaschine konkret dargestellt, da diese sowohl die einzelnen Prozessschritte als auch die jeweilige Konfiguration des iTaSC Algorithmus und des Reglers beschreiben. Die eingesetzten Skills richten sich nach Tabelle A.1 und wurden ggf. um einen zu erprobenden Regler (ARF oder PPC) erweitert. Die Auswahl der in den Abläufen eingesetzten Regler ist im *pitasc*-Knoten unterhalb des iTaSC Blocks vermerkt. Skills welche während der Ausführung den aktiven Regler ändern tragen den Bezeichner des Reglers im Namen.

4.1.3 Vergleichsregler

Zum Vergleich der in dieser Arbeit dargestellten Regelungsansätze mit dem Stand der Technik wird je eine Positionsregelungs- und eine Kraftregelungsmethodik herangezogen.

Die Positionsregelung basiert auf einem trapezförmigen Geschwindigkeitsprofil. In Abbildung 4.3 die Geschwindigkeitsverläufe der internen Regelung eines UR5 Roboters mit verschiedenen maximalen Geschwindigkeiten aufgetragen. Mit konstanter Beschleunigung wird die Geschwindigkeit erhöht, bis entweder die maximale Geschwindigkeit erreicht wurde oder der Entschleunigungsvorgang zum Erreichen der Zielposition eingeleitet wird. Die hier aufgezeigten Verläufe resultieren direkt aus einem kartesischen Bewegungsbefehl an den internen Robotercontroller. Daher wird dieser Regler als *nativer Regler*, d. h. als robotereigener, interner Regler bezeichnet.

Für Kraftregelung kommt ein Impedanz-Regler erster Ordnung (im Folgenden nur *Impedanzregler*) zum Einsatz. Impedanzregelung ist eine in der Literatur übliche Form der Kraftregelung. Die Struktur entspricht einer PI-Regelung. In dieser Arbeit wurde zur Rauschunterdrückung die Frequenzantwort der Regelung mittels eines Filters mit unendlicher Impulsantwort (IIR Filter)

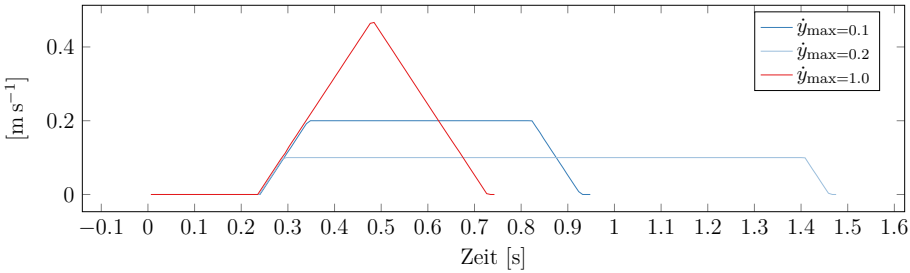


Abbildung 4.3: Trapezprofil der Geschwindigkeitsvorgaben des internen UR Positionsreglers

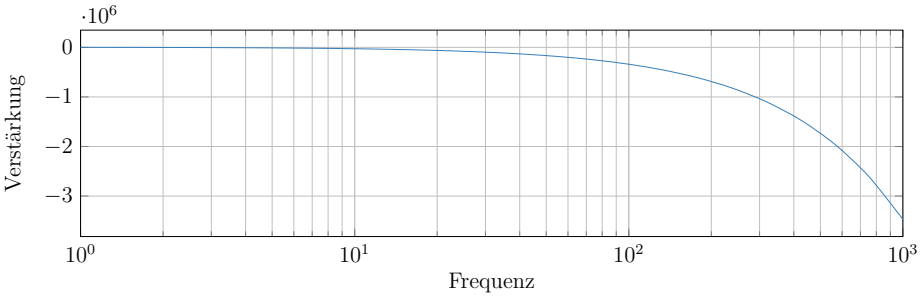


Abbildung 4.4: Verstärkungskennlinie für eines zeitdiskreten IIR Filters zur Realisierung eines Impedanz-Reglers erster Ordnung mit der Eckfrequenz von 5 Hz, einer Nachgiebigkeit von 0.0005 N m^{-1} und der Abtastfrequenz von 125 Hz

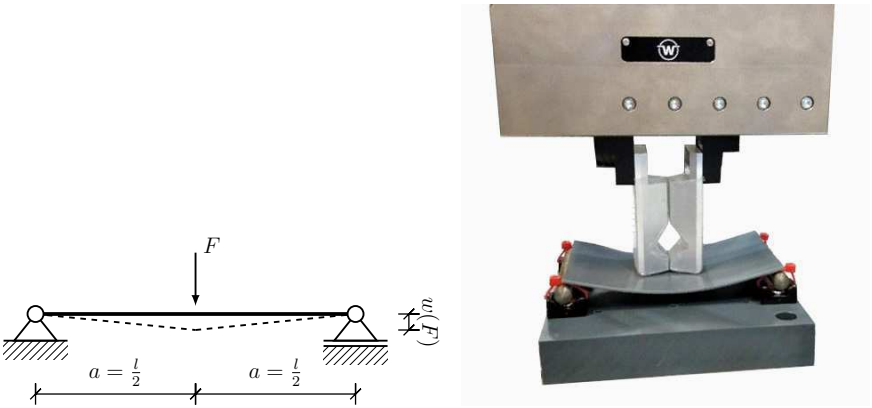


Abbildung 4.5: Mechanischer Aufbau zur Untersuchung verschiedener Umgebungssteifigkeiten

konstruiert. Der Verstärkungsfaktor bestimmt die Nachgiebigkeit des geregelten Systems. Zusätzlich wird eine Eckfrequenz zur Unterdrückung von hochfrequentem Sensorrauschen angenommen. Zur Diskretisierung des IIR Filters werden die Filterkoeffizienten durch eine bilineare Transformation bestimmt. Mit diesen Anpassungen bestimmt sich die Struktur des Impedanzreglers zu einem PPT1-Glied. In Abbildung 4.4 ist ein Frequenzgang des Filters, bzw. des Verstärkungsfaktors mit der Eckfrequenz von 5 Hz, einer Nachgiebigkeit von $0.0005\,\mathrm{N\,m^{-1}}$ und der Abtastfrequenz von 125 Hz dargestellt.

4.1.4 Umgebungssteifigkeit

Um die Kraftregelung im Umgang mit verschiedenen Umgebungssteifigkeiten zu erproben, wurde ein mechanischer Aufbau eingesetzt. Der mechanische Aufbau ist in Abbildung 4.5 dargestellt. Eine Kontaktkraft wird mit einer frei gelagerten Platte aufgebracht. Abhängig vom eingesetzten Material und Abmaß der Platte kann die Umgebungssteifigkeit für die Experimente variiert werden.

Die kraftabhängige Durchbiegung $w(F)$ und somit die effektive Nachgiebigkeit $c = F/w$ kann mittels linearer Balkentheorie abgeschätzt werden. Vereinfacht wird dazu angenommen, dass eine Einzelkraft F in der Mitte des Balkens $a = l/2$ wirkt. In diesem Spezialfall ergibt sich die Durchbiegung w zu

$$w = \frac{IE \cdot l^3}{48} F \tag{4.1}$$

mit dem materialabhängigen Elastizitätsmodul E und dem Flächenträgheitsmoment I. Das Flächenträgheitsmoment I bestimmt sich für rechteckige Vollplatten zu

$$I = \frac{h^3 b}{12} \tag{4.2}$$

mit den Dicke der Platte h und der Breite b. Hiermit ergibt sich bspw. für eine PVC-Platte mit dem Elastizitätsmodul $E = 2800\,\mathrm{MN\,m^{-2}}$, der Länge $l = 150\,\mathrm{mm}$, der Breite $b = 100\,\mathrm{mm}$ und der Stärke von $h = 2\,\mathrm{mm}$ eine Nachgiebigkeit von $c \approx 2655\,\mathrm{N\,m^{-1}}$.

4.2 Durchführung

Im Folgenden werden die einzelnen Experimente beschrieben und dargestellt. Zunächst werden die Teilkomponenten einzeln erprobt und im weiteren Verlauf zunehmend zu einem Gesamtsystem integriert.

4.2.1 Kartesische Bewegung mit sättigender dynamischer Ausgangsrückführung

Als erster Regelungsansatz wird die Ausgangsrückführung (ARF) aus Kapitel 3.2 erprobt. Die ARF wird auf Basis eines bekannten Modells synthetisiert und mit einem Positionssprung erprobt.

Hierfür wurde im Vorfeld eine einzelne Sprungantwort aufgenommen. Für eine konstante Zeit wird eine lineare kartesische Geschwindigkeitsvorgabe in den Achswinkelraum transformiert und hiermit der Roboter angesteuert. Zur Modellbildung wird ausschließlich die Beschleunigungsfahrt betrachtet. Das Abbremsverhalten wird nicht in die Modellbildung einbezogen. Mit der Aufzeichnung der Geschwindigkeitsführungsgröße und der entsprechenden Systemantwort wird mit dem in Kapitel 3.1.2 beschriebenen Vorgehen ein lineares Modell gebildet. Die entsprechenden gefundenen Modelle und Sprungantworten mit den jeweils synthetisierten ARF sind in Tabelle 4.1 und Tabelle 4.2 dargestellt.

Das Modell bildet die Grundlage der Reglersynthese nach Kapitel 3.2.1. Wie auch dort wird das geschwindigkeitsbasierte Modell mit Gleichung (3.37) um einen Integrator erweitert, um einen Positionsregler auszulegen. Die im Vorfeld synthetisierten Reglerkoeffizienten werden bereitgestellt und von der *pitasc* Implementierung der ARF bei Eintritt in den entsprechenden Skill eingelesen.

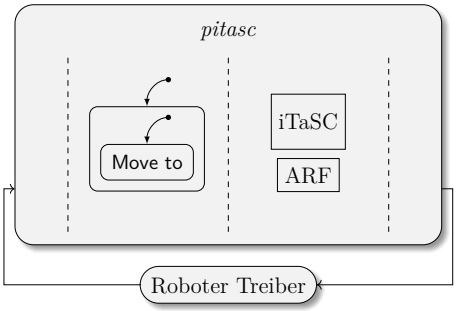


Abbildung 4.6: Darstellung des Programmablaufs eines Positionssprungs mit einer Ausgangsrückführung (ARF). Der Skill Move to bestimmt den Regler ARF zur Positionsregelung.

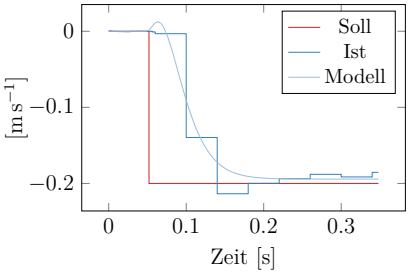
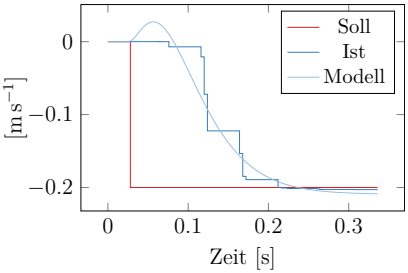
Das Zustandsdiagramm dieses Experiments in Abbildung 4.6 besteht aus zwei Zuständen. Der erste Zustand realisiert eine lineare Bewegung des Roboters zu der Ausgangsposition. Ist diese Position erreicht, wird die nächste kartesische Position angefahren. Für die zweite Bewegung wird die Positionsreglung durch die ARF übernommen. Die Abfolge der beiden Skills kann beliebig oft wiederholt werden. Der exemplarische Ablauf steht stellvertretend für eine beliebige Roboteraufgabe, welche eine durch die ARF geregelte lineare Bewegung enthält.

In den ersten Darstellungen in Tabelle 4.1 ist zunächst zu erkennen, dass sich die charakteristische Antwort der Roboter auf die Geschwindigkeitssprünge unterscheidet. Während der Denso Roboter ein überdämpftes Verhalten zeigt, ist bei UR5 und Panda ein leichtes Überschwingen zu erkennen. Zur Modellannäherung mit dem Verfahren aus Kapitel 3.1.2 müssen die aufgezeichneten Daten äquidistant verteilt sein. Als Basis für alle Modellannäherungen wurde auf eine Abtastzeit von 4 ms interpoliert. Der Betrag der Pole des Modells wird auf 100 beschränkt und der Parameter der Pseudo-Huber Funktion zu $\delta = 0.01$ gewählt. Als Solver wird *Powell* eingesetzt. Die Modellapproximation eines PT2-Glieds mit Totzeit nach Kapitel 3.1.2 neigt zur Modellbildung rein reeller Systeme. Ein Überschwingen, durch die Approximation mit konjugiert-komplexen Polpaaren, wird somit praktisch nicht modelliert.

Tabelle 4.2 zeigt die jeweiligen Soll- und Istgrößenverläufe eines Positionssprungs mit der entsprechenden auf Basis des angenäherten Modells in Tabelle 4.1 synthetisierten ARF. Die Ausgangsbegrenzung wurde zu $u_{\max} = 0.2\,\mathrm{m\,s^{-1}}$, und die Kanten des zulässigen Einzugsgebietes X_0 zu

$$\bar{\boldsymbol{x}}_0 = \begin{bmatrix} \pm 1 & \pm 2 & \pm 10 & \pm 100 \end{bmatrix}^\intercal$$

Tabelle 4.1: Initiale Annäherung der Robotermodelle mit einzelner Sprungantwort

Panda	**Denso**
	

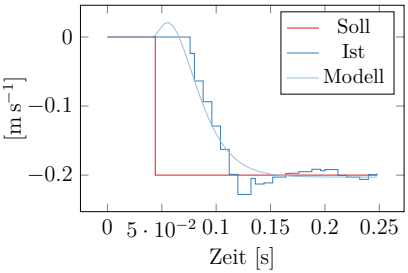
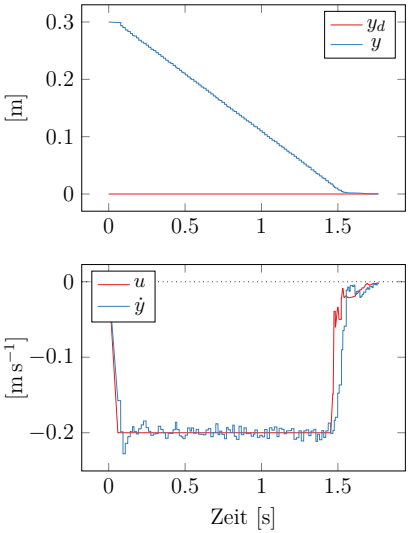
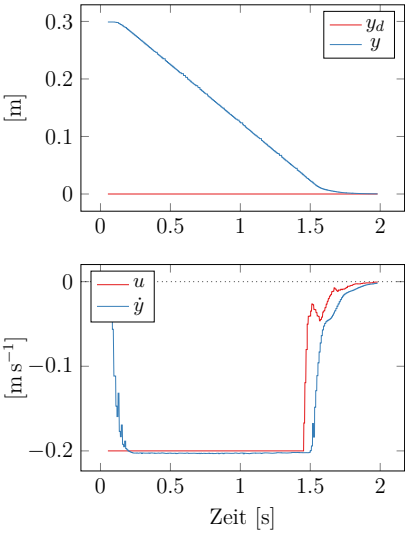
UR5


Tabelle 4.2: Auswertung der Ausgangsrückführung mit verschiedenen Robotern

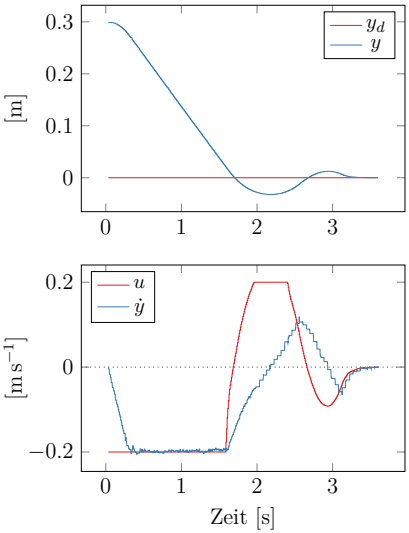
Panda



Denso



UR5



festgelegt. Die Begrenzungsparameter des Realteils der Pole des geregelten Systems wurden mit $\rho = 150$ größer als die betragsmäßig größtmöglichen Pole des angenäherten Modells gewählt. Die Wahl der Begrenzung des Imaginärteils der Pole richtet sich nach den Erfahrungen aus den Simulationen aus Kapitel 3.2.2 und wurde mit $\varphi_{\max} = 50$ festgelegt. Als Sollvorgabe der Sprungantworten wurde jeweils eine lineare Aufwärtsbewegung des Endeffektors mit einer Distanz von 30 cm gegeben.

Sowohl der Panda als auch der Denso Roboter zeigen einen erwartungsgemäßen Verlauf. Die Regelgröße sättigt sehr schnell auf die zulässige Maximalgeschwindigkeit $u_{\max}$. Betritt die Sollvorgabe den ungesättigten Bereich, zeigt sich ein typischer exponentiell abklingender Stellgrößenverlauf u. Die jeweilige Istgröße $\dot{y}$ folgt prinzipiell u unter dem Einfluss von Rauschen und Zeitverzug.

Im Falle des Denso Roboters reagiert die Istgröße $\dot{y}$ träge auf die Sollvorgabe während des Einschwingvorgangs. Der Grund für dieses Verhalten liegt in der internen Implementierung des Robotertreibers. Über die verfügbare Softwareschnittstelle ist eine direkte geschwindigkeitsbasierte Ansteuerung der Achsen nicht vorgesehen. Es muss ein Umweg über die Vorgabe einer taktbasierten Positionsvorgabe der Achswinkel gegangen werden. Werden die jeweiligen Sollpositionen aus der aktuellen Position und eine Extrapolation mit der gewünschten Sollgeschwindigkeit über den nächsten Takt berechnet, so befindet sich der interne Regler zu jedem Zeitpunkt auf der Bremsrampe der roboterinternen Regelung. Dadurch wird die vorgegebene Geschwindigkeit nie vollständig erreicht und effektiv ca. um einen Faktor von 1/8 verringert. Um die angeforderte Geschwindigkeit erreichen zu können, wird die Extrapolation der Sollposition nicht mit der jeweils aktuellen Position durchgeführt, sondern auf Basis der im letzten Takt theoretisch erreichten Position, sprich der letzten Sollpositionsvorgabe. Dadurch wird eine im vorherigen Takt ggf. nicht erreichte Position weiter verschoben, als es die aktuelle Geschwindigkeitsvorgabe fordern würde. Die Bremsrampe wird verlassen und die gewünschte Geschwindigkeit kann erreicht werden. Der entstehende Nachlauf schlägt sich allerdings negativ auf die Sprungantwort nieder.

Durch die Modellbildung anhand eines einzelnen Geschwindigkeitssprungexperiments wird der Nachlauf nicht modelliert und somit auch nicht in die Synthese der ARF einbezogen. Die entsprechenden negativen Auswirkungen verstärken sich proportional mit der Robotergeschwindigkeit und können mitunter zu einem unerwünschten Einschwingverhalten führen.

Ein weiterer Einschwingvorgang ist in dem Verlauf des UR5 Roboters zu erkennen. Wie auch bei dem Denso Roboter konnte das Modell auf Basis einer einzelnen Sprungantwort nicht ausreichend genau gebildet werden.

In den folgenden Experimenten in Kapitel 4.2.3 wird die Modellannäherung mittels eines IEKF wiederholt und ein geeigneteres Modell angenähert, welches für die Synthese einer schnelleren ARF eingesetzt wird.

4.2.2 Kontinuierliche Modellüberwachung

Im Folgenden wird isoliert der Modellidentifikationsansatz aus Kapitel 3.1 erprobt. Mit einem zufällig gewählten Anfangszustand nähert sich das Modell des IEKF dem dynamischen Verhalten des untersuchten Roboters an. Ein so angenähertes Modell bildet im weiteren Verlauf die Grundlage der modellbasierten ARF Synthese.

Ein zusätzlicher ROS-Knoten zur kontinuierlichen Modellüberwachung mittels IEKF wird in diesem Experiment an den vorherigen Ablauf angefügt. Während der Ausführung werden für das IEKF relevante Daten als Stell- und Istgröße im Aufgabenraum bereitgestellt. Das IEKF greift diese Daten auf und nähert die Modellparameter zusammengefasst im Zustandsvektor $\boldsymbol{z}$ nach (3.7) an. Die Zustandsraumstruktur des Modells in Form einer entsprechenden initialen Blockdiagonalmatrix $\boldsymbol{A}$ sowie initiale Werte der Matrizen $\boldsymbol{B}$ und $\boldsymbol{C}$ werden vor Beginn der Roboterbewegung an das IEKF übergeben. Als initiale Annahme kann das der ARF zugrundeliegende Modell oder zufällige Werte genutzt werden.

Es wird nicht der gesamte Ablauf des Roboterprogramms durch das IEKF überwacht, sondern nur ein Teilablauf. Dieser Teilablauf ist eine der beiden zyklisch wiederholten linearen Bewegungen. Liegt bereits eine synthetisierte ARF vor, so kann auch diese direkt überwacht werden.

Das entsprechende Zustandsdiagramm in Abbildung 4.7 basiert auf der vorherigen Abfolge aus 4.2.1, wobei die Wahl des Reglers prinzipiell nicht weiter relevant ist. Lediglich der Datenaustausch zu dem IEKF-Knoten wurde eingerichtet.

Das IEKF erhält während der Ausführung des ersten Skills die erforderlichen Daten. Somit wird ein spezielles Modell gebildet, welches genau für diesen Teilablauf gültig ist. Zusätzlich zu den Erkenntnissen aus den Simulationsvorgängen soll in diesem Experiment beobachtet werden, wie sich das IEKF verhält, wenn die Informationen aus voneinander getrennten Zeitserien bezogen werden. Hierbei ist zu erwähnen, dass beim Eintritt in den beobachteten Skill die internen Zustände des IEKF zurückgesetzt werden, wenn mithilfe von Zeitstempeln festgestellt wird, dass diese veraltet sind.

Zur exemplarischen Darstellung der Modellüberwachung wird die Annäherung eines UR5 Roboters Modells betrachtet. Das IEKF wurde mit zwei konjugiert komplexen Polpaaren und einem

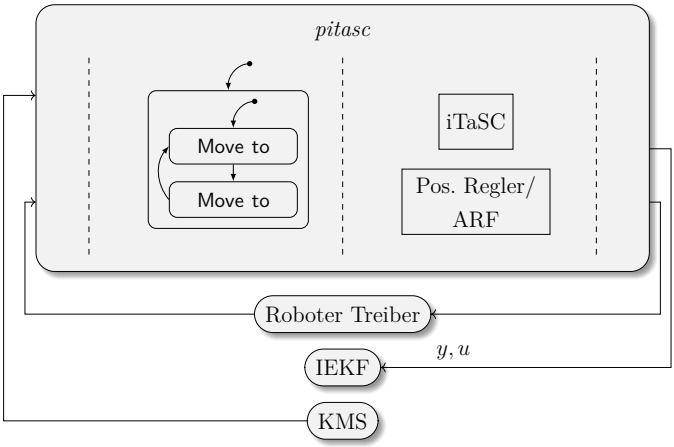


Abbildung 4.7: Darstellung des Programmaufbaus zur Identifikation eines Streckenmodells. Der Roboter wird zwischen zwei Zielpositionen verfahren. Die Realisierung der Regelung spielt zur Identifikation keine Rolle.

weiteren reellwertigen Pol mit zufälligen Werten initialisiert. Die anfängliche Annäherung des Modells ist nicht weiter dargestellt. Bis ca. 5 min ist eine akzeptable Annäherung erreicht, welche im Verlauf der ersten Sprungantworten in Abbildung 4.8 dargestellt ist. Das Überschwingen aufgrund der komplexen Polpaare ist noch auffällig und wurde bislang noch nicht durch das IEKF kompensiert.

Der UR5 erlaubt eine manuelle Anpassung der Einstellung der Geschwindigkeitsskalierung während der aktiven Ausführung eines Roboterprogramms. Dieses wird im Folgenden genutzt, um eine Veränderung der Roboterdynamik nachzustellen. Nach ca. 3 min wird die Verfahrgeschwindigkeit des Roboters auf ca. 50 % reduziert.

Trotz gleichbleibender Sollvorgaben verringert sich die Systemantwort des Roboters aus Sicht des IEKFs. Diese Veränderung findet außerhalb des Einflusses der *pitasc* Robotersteuerung oder des IEKFs statt. Die Komponenten haben keine weitere Kenntnis über die manuelle Anpassung.

Ein größerer Modellfehler wird bei der zweiten dargestellten Sprungantwort sichtbar. Das Filter beginnt mit der Anpassung der Modellparameter. Bereits nach zwei weiteren Sprüngen wurde die Höhe der Sprungantwort des Modells also der Verstärkungsfaktor des Modells deutlich reduziert.

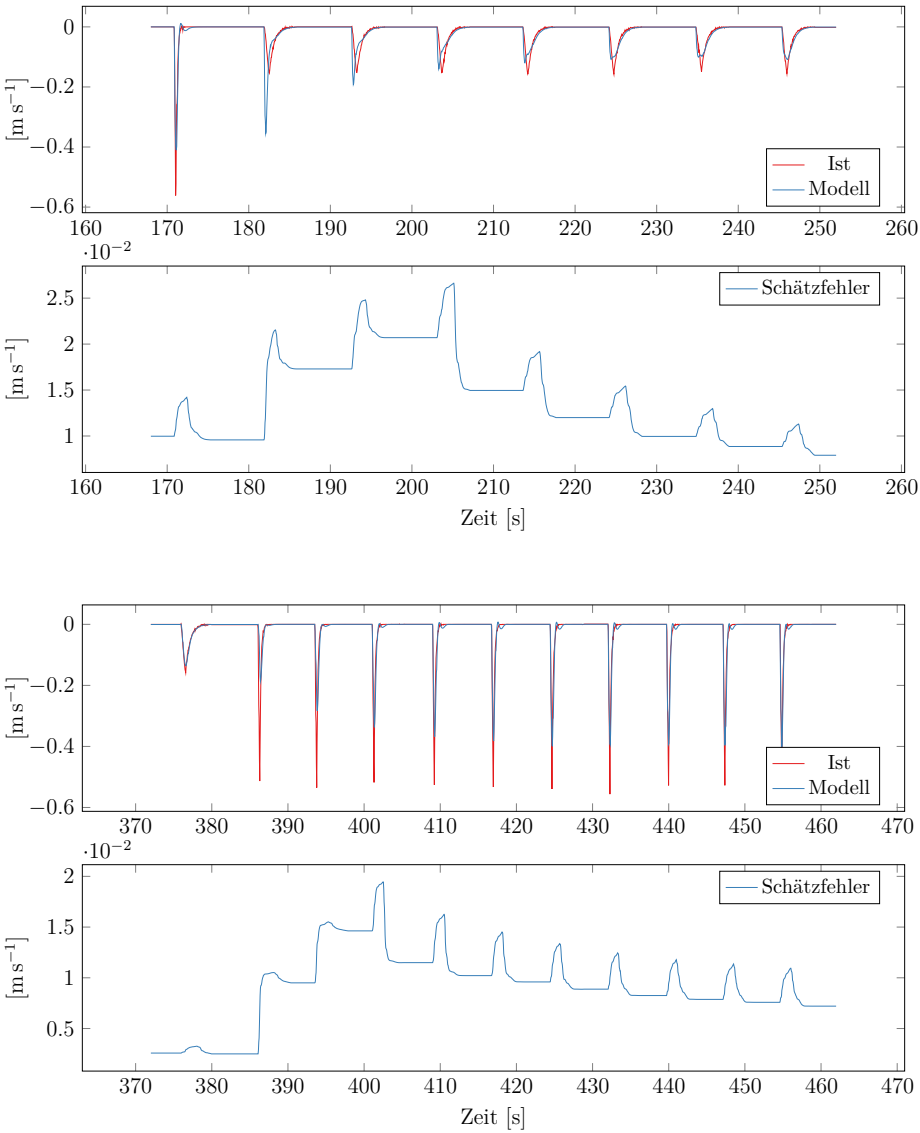


Abbildung 4.8: Modellanpassung einen schwingungsfähigen PT2 Modells bei externer Manipulation der Robotergeschwindigkeit

Eine umgekehrte Manipulation wird zu dem weiteren Zeitpunkt von ca. 6.5 min wiederholt. Auch hierbei reagiert das IEKF binnen weniger Sprüngen. Der Modellfehler pendelt sich auf den vorherigen Wert von ca. 0.01 ein.

Im Vergleich der Zeitserien in Abbildung 4.8 fällt weiter auf, dass der absolute Wert des Modellfehlers bei reduzierter Geschwindigkeit einen kleineren Wert von ca. 0.005 erreichen konnte. Somit konnte das IEKF in den Experimenten das langsamere System besser annähern.

Das IEKF war in den Experimenten in der Lage über einen Verlauf von ca. 5 min ein lineares Modell anzunähern, welches zur Synthese einer ARF eingesetzt werden soll. Auf eine Systemänderung während der Identifikation wurde geeignet reagiert und das unterliegende Modell angepasst.

4.2.3 Synthese einer Ausgangsrückführung mit identifizierten Modell

Aufbauend auf den vorherigen Experimenten wird im Folgenden erprobt, wie ein durch einen IEKF angepasstes Modell zur Synthese von verbesserten Reglerparameter einer Ausgangsrückführung (ARF) eingesetzt werden kann. Ziel der Experimente ist es sicherzustellen, dass das durch den IEKF angenähertes lineares Modell zur Synthese einer ARF geeignet ist.

Neben dem IEKF-Knoten wird ein weiterer ROS-Knoten notwendig, welcher die Parametersynthese implementiert (genannt: *Synth Server*). Das Ergebnis der Synthese, also die Parameter der ARF auf Basis des durch den IEKF identifizierten Zustandsraummodells, wird in dem folgenden Bewegungsskill eingesetzt. Durch die Trennung in einen separaten ROS-Knoten wird die Synthese der Regelungsparameter parallel zum Bewegungsablauf durchgeführt und beeinflusst nicht durch zusätzlich Rechenleistungsbedarf den Roboterablauf.

Das entsprechende Zustandsdiagramm ist in Abbildung 4.9 dargestellt. Nachdem der überwachte Teilablauf verlassen wurde, kann der aktuelle Fehler des angepassten Modells mit dem realen Ist-Verlauf verglichen und mit einem Grenzwert überprüft werden. Die Überprüfung wird in dem Skill Check Error realisiert. Liegt der Modellfehler unter einer akzeptablen Grenze, so wird die Synthese von Reglerparametern durch den Skill Synth angefordert. Mit dem Ergebnis der Synthese werden die Parameter der ARF aktualisiert. Unterschreitet der Fehler nicht die Akzeptanzgrenze, so wird die Anforderung einer Synthese übersprungen. Nach einer abgeschlossenen Synthese und somit einer erfolgreichen Reparametrisierung des Reglers wird die Fehlerakzeptanzgrenze prozentual um 25 % verschoben, um eine erneute Reglersynthese nur auszulösen, wenn der Modellfehler weiter verringert wurde.

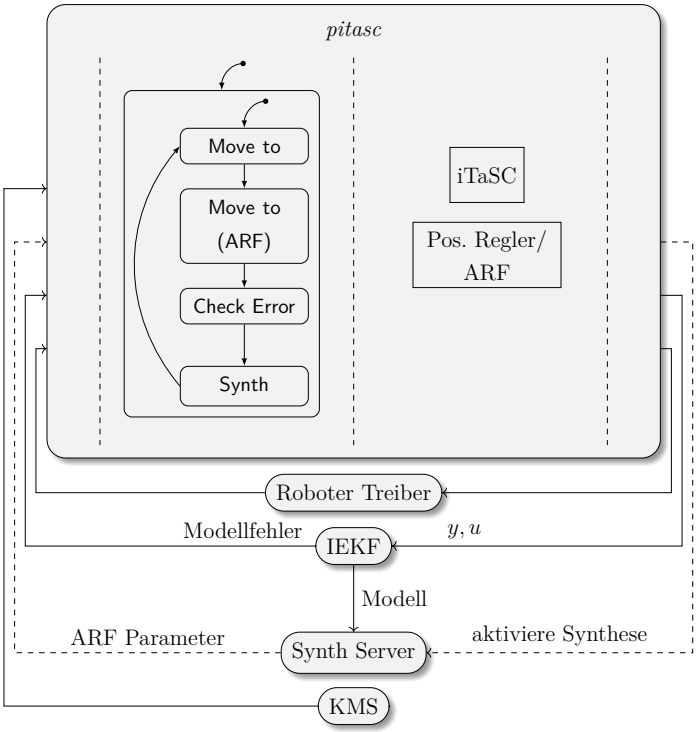


Abbildung 4.9: Darstellung des Programmablaufs einer ARF Synthese mit paralleler IEKF Überwachung. Der Skill Move to (ARF) bestimmt den Regler ARF zur Positionsregelung. Datenströme via *rostopics* sind mit durchgezogenen Pfeilen dargestellt. Bei dem gestrichelt dargestellten *rosservice* wurde zum besseren Verständnis eine Unterteilung der Anfrage und der Antwort beschrieben.

In Abbildung 4.10 ist eine exemplarische Zeitserie einer initialen Modellannäherung des Denso VS087 Roboters dargestellt. Für die Berechnung des Modellfehlers wird ein endliches Fenster von Datenpunkten eingesetzt. Daher kann eine verlässliche Aussage erst nach dem Sammeln von entsprechend vielen Datenpunkten gegeben werden. Ein negativer Fehlerwert zeigt fehlende Datenpunkte an. In der dargestellten Zeitserie wird nach ca. 16 s ein Modellfehlerwert abgegeben.

Wie bei dem vorherigen Experiment wurde die Modellform mit einem reellen Pol und einem konjugiert komplexen Polpaar vorgegeben und die anfänglichen numerischen Parameter der Systemmatrizen wurden zufällig gewählt.

Zunächst werden die stationäre Abweichung und der Verstärkungsfaktor des Modells angepasst. Nach ca. 1 min zeigt das Modell ein starkes Überschwingverhalten. Hingegen wurden die stationäre Abweichung und der Verstärkungsfaktor weitestgehend angenähert. Im weiteren Verlauf nähert das IEKF das überdämpfte Verhalten des Robotersystems an.

Der Modellfehler verläuft näherungsweise exponentiell abklingend. Ein Grenzwert des Modellfehlers von $3 \cdot 10^{-3}$ wird nach ca. 330 s unterschritten. Zu diesem Zeitpunkt wird die Reglersynthese ausgelöst. Der Grenzwert und der Zeitpunkt der Auslösung wurde im Verlauf des Modellfehlers in Abbildung 4.10 durch ein rote Linie, sowie einem roten umkreisten Kreuz markiert.

Eine Sprungantwort der synthetisierten ARF ist in Abbildung 4.11 dargestellt. Es ist ein leichtes Überschwingen im Einschwingvorgang der Stellgröße u zu erkennen. Ein Überschwingen der Istgröße der Position ist allerdings nicht zu erkennen. Das angenäherte Modell scheint die Dynamik des Denso Roboters geeignet zu modellieren.

Um die Reglergüte der ARF vergleichen zu können, wurde das gleiche Experiments mit einem Universal Robots UR5 wiederholt und die Soll- und Istgrößenverläufe der Sprungantwort in der Abbildung 4.12 mit den Verläufen einer roboterinternen (nativen) Sprungantwort verglichen. Das Robotermodell wurde zuvor durch einen zufällig initialisierten IEKF angenähert. Nachdem der Schätzfehler die Akzeptanzgrenze unterschritten hat, wurde die Synthese der ARF ausgelöst.

Wie in Abbildung 4.11 ist auch in Abbildung 4.12 ein Überschwingen der Stellgröße u erkennbar. Allerdings zeigt sich auch ein Überschwingen der Sollposition y und somit ein unerwünschtes Positionierverhalten und ein verlängerter Einschwingvorgang.

Für den Vergleich wurde derselbe Positionssprung mit der roboterinternen (nativen) Regelung durchgeführt und in die Abbildung als y_{native} und $\dot{y}_{\text{native}}$ eingefügt. Die Verläufe der roboter-

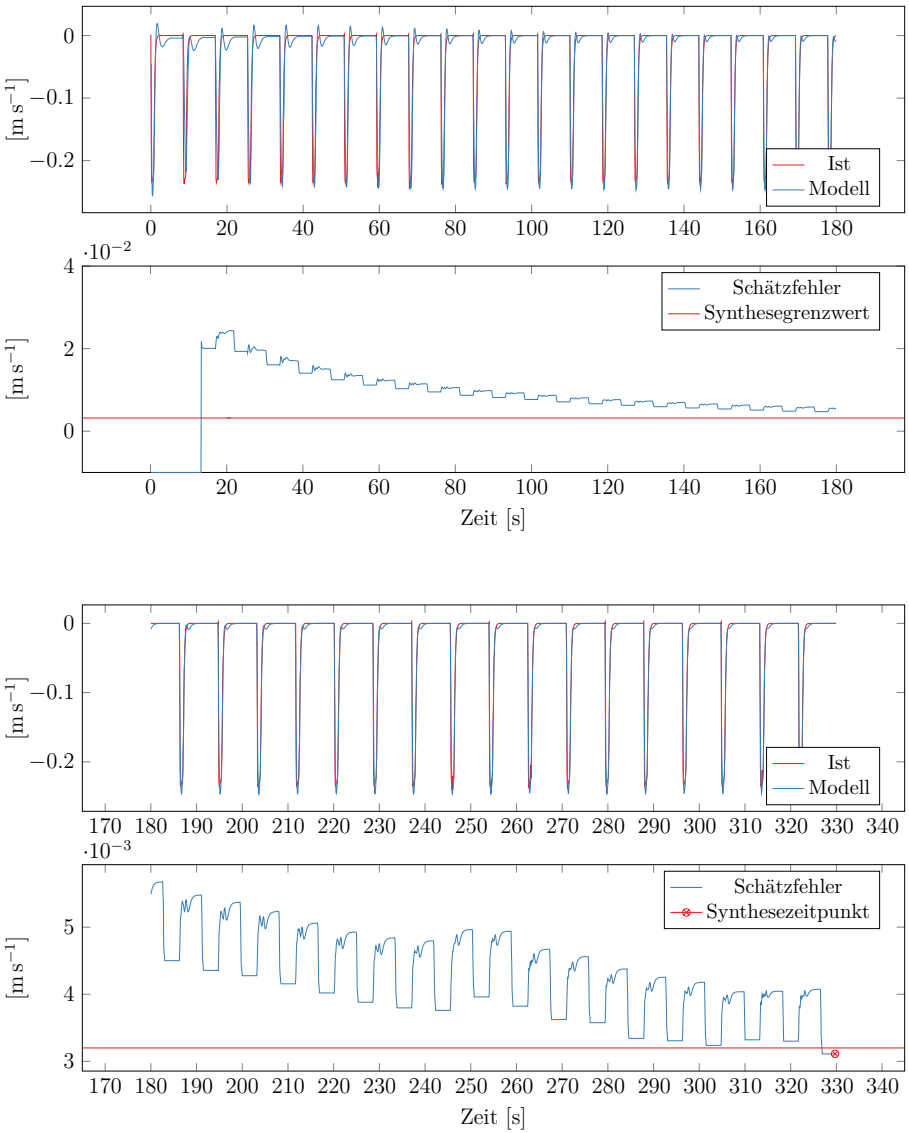


Abbildung 4.10: Modellanpassung eines Denso Roboters mit Auslösung einer Reglersynthese bei Unterschreitung eines maximalen Fehlers

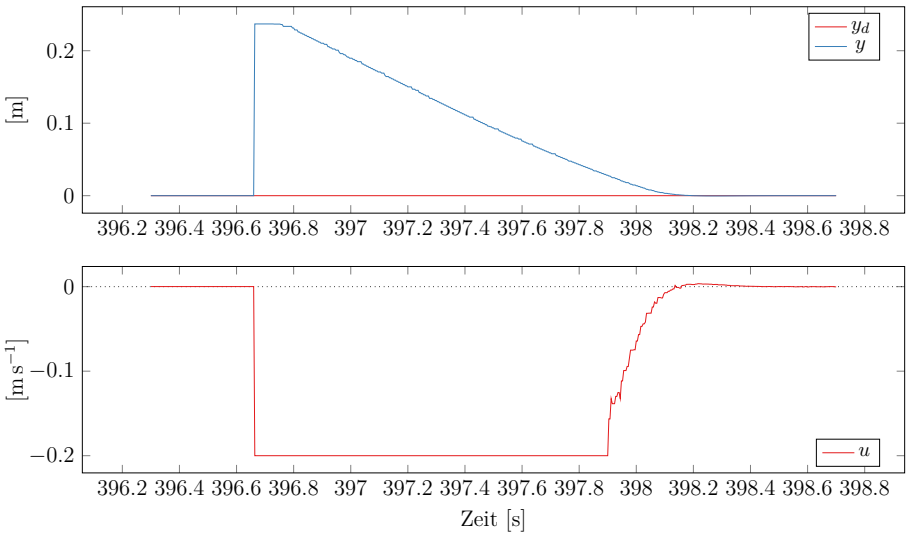


Abbildung 4.11: Darstellung einer Sprungantwort der synthetisierten Ausgangsrückführung mit einen Denso Roboter

internen Regelung wurde um die Totzeit der externen Regelung mittels ARF von ca. 50 ms verschoben. Hierdurch können die Verläufe direkt verglichen werden.

Der Einschwingvorgang y_{native} der roboterinternen Regelung beträgt in diesem Experiment ca. 0.5 s, wogegen der Einschwingvorgang y der ARF mit ca. 0.8 s etwas länger dauert. Beim Vergleich der Istgeschwindigkeiten $\dot{y}$ und $\dot{y}_{\text{native}}$ zeigt sich anfänglich ein sehr ähnlicher Verlauf. Jedoch wird durch die roboterinterne Regelung die Istgeschwindigkeit früher verringert und so ein Über- und Einschwingvorgang vermieden. Hierzu im Vergleich reagiert die ARF mit einem deutlichen Überschwingen bei Erreichen der Sollposition. Zwar erreicht die ARF das Ziel etwas eher, kann allerdings die Sollposition nur langsamer stabilisieren. Das weniger aggressive Verhalten der roboterinternen Regelung bedingt hier die kürzere Einschwingdauer.

Das in dem praktischen Experiment beobachtete Überschwingverhalten der ARF ist für die Positionsregelung eines Roboters unerwünscht und relativiert die erfolgversprechenden Simulationsergebnisse des Verfahrens. Der schwachen Robustheit der ARF gegenüber weichen Echtzeitbedingungen muss durch eine geeignete harte Echtzeitumgebung begegnet werden. Unter Anbetracht der fehlenden Kenntnisse der Roboterdynamik, der Modellierungszeit von ca. 5 min, der begrenzten Taktrate sowie der Totzeit ist die Reglergüte trotzdem zufriedenstellend.

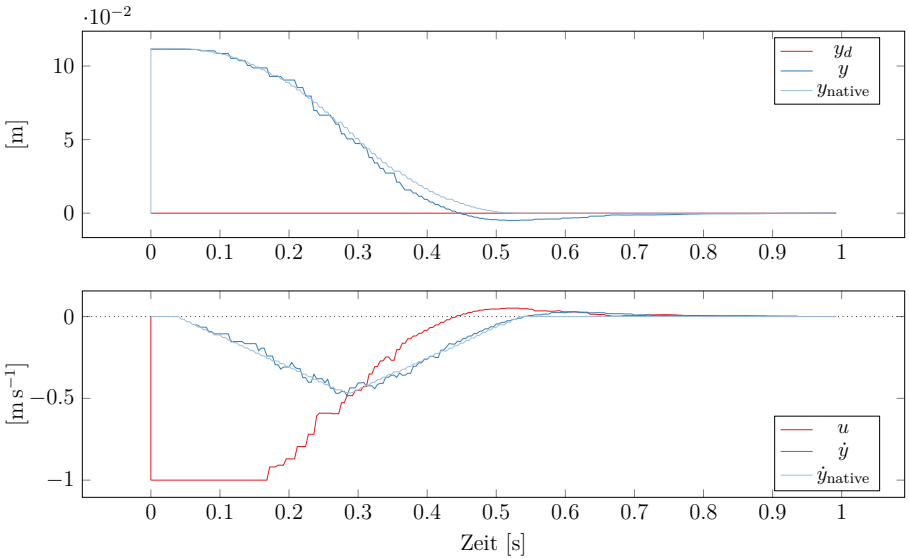


Abbildung 4.12: Darstellung einer Sprungantwort der synthetisierten Ausgangsrückführung auf einen UR5 Roboter

4.2.4 Kontaktkraftregelung mit vorgegebenem Performanzband

In diesen Experimenten wird das Verhalten der in Kapitel 3.3 vorgestellten PPC erprobt. Mit verschiedene Robotern und verschiedenen Umgebungssteifigkeiten werden verschiedene Anforderungsszenarien abgebildet und schließlich deren Abdeckung untersucht. Das Ziel der Experimente ist es das Regelungskonzept in der praktischen Anwendung in Betrachtung der Übertragbarkeit zwischen Robotern und Anwendungen zu verifizieren.

Der Aufbau des Experiments ist analog zu dem Aufbau der simulationsbasierten Untersuchungen in Kapitel 3.3.3. Lediglich die Roboter- und Sensorsimulation wird durch die entsprechende Anbindung an reale Roboter und Kraft-Momenten-Sensoren ersetzt.

Das Zustandsdiagramm in Abbildung 4.14 besteht aus einer kartesischen Bewegung zu einer definierten Pose über einen Oberflächenkontakt. Ist diese erreicht, wird mit konstanter Geschwindigkeit ein Kontakt aufgebaut. Übersteigt die gemessene Kraft eine signifikante Schwelle, so wird die eigentliche Stabilisierung bzw. Regelung der Kontaktkraft ausgeführt.

Abbildung 4.13: Bilder der Roboter Panda, Denso VS087, UR5 bei der Kontaktkraftregelung

Tabelle 4.3: Auflistung der verschiedenen Platten zur Variation der Umgebungssteifigkeit

Nummer	Steifigkeit [$\mathrm{N\,m^{-1}}$]	Material	E-Modul [$\mathrm{GN/m^2}$]	Maße l×b×h [$\mathrm{mm^3}$]
(1)	260	PVC	0.1	$100 \times 65 \times 1$
(2)	10800	PVC	0.1	$100 \times 100 \times 3$
(3)	18200	Aluminium	7	$100 \times 65 \times 1$
(4)	25584	PVC	0.1	$100 \times 100 \times 4$
(5)	54600	Stahl	21	$100 \times 65 \times 1$

Der Skill zur Regelung der Kontaktkraft setzt sich aus zwei priorisiert angeordneten Skills zusammen. Der eigentliche Kraftregelungsskill Push (PPC) hat höchste Priorität. In diesem Experiment wird genau eine Aufgabenraumkoordinate, nämlich die Kraft $f_z = 45\,\mathrm{N}$ in Richtung z auf den Oberflächenkontakt vorgegeben. Alle weiteren Freiheitsgrade sind zunächst durch den Kraftregelungsskill unbestimmt. Daher wird ein nieder priorisierter Skill Hold pose angefügt, der alle (bislang unbestimmten) Freiheitsgrade dazu bestimmt die Pose bei Eintritt in den Skill beizubehalten. Durch die hierarchische Anordnung überstimmt Push PPC in f_z Richtung die Sollvorgaben des Skills Hold pose.

Neben der Variation der eingesetzten Roboter werden Kontakte mit verschiedenen Umgebungssteifigkeiten untersucht. Die Umgebungssteifigkeiten werden mit einem mechanischen Aufbau nach Kapitel 4.1.4 variiert. Die fünf verschiedenen Platten sind in Tabelle 4.3 aufgeführt.

Zur Untersuchung des Verhaltens des PPC Reglers mit der Verschiebung der Performanzfunktion werden Kontaktexperimente mit dem UR5 Roboter auf der Aluminiumplatte (3) näher in Abbildung 4.15 dargestellt. Dargestellt ist hier der Regelfehler e und die entsprechenden Performanzgrenzen p.

Bei ausreichender Dynamik des Roboters verläuft der Regelfehler e vollständig innerhalb der Grenzen p. Wurden die Grenzen wie bei p_{init} zu eng gewählt, so verschieben sich die Grenzen p,

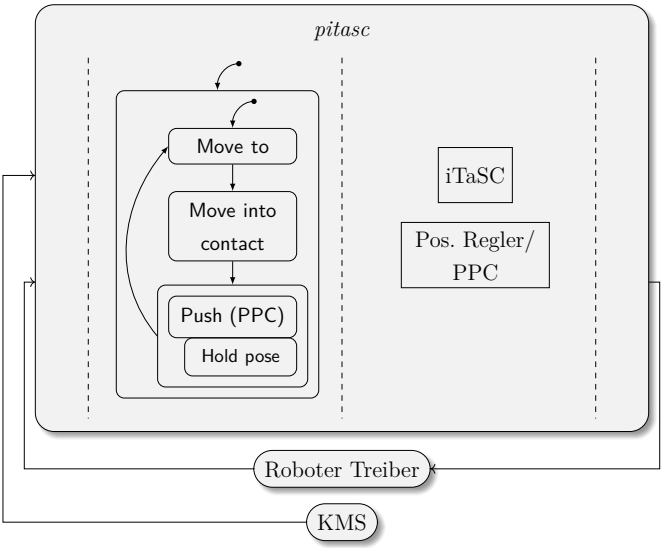


Abbildung 4.14: Darstellung des Experimentes zum Oberflächenkontaktaufbau

bis diese nicht mehr verletzt werden. Es ergibt sich eine verschobene effektive Performanzgrenze p_{eff}.

Weiter werden in Abbildung 4.16 die Regelfehlerverläufe e und die effektiven Performanzgrenzen p_{eff} für verschiedene Sollkraftvorgaben dargestellt. Mit identischen Parametern wird eine Kontaktkraft zwischen dem UR5 Roboter und einem Stahlblech (5) eingeregelt.

Jeder Roboter verhält sich entsprechend seines eigenen dynamischen Verhaltens, bspw. durch den Unterschied zwischen der bereits in Kapitel 4.2.1 erwähnten achspositionsbasierten Ansteuerung des Denso Roboters gegenüber der Achsgeschwindigkeitsvorgaben des Panda und UR5 Roboters. Neben der individuellen (zulässigen) maximalen Geschwindigkeit und Beschleunigungen variiert auch die intrinsische Steifigkeit der Roboterglieder und -achsen. Weiterhin bietet jeder der Roboter mehr oder weniger geeignete Softwareschnittstellen zur externen Steuerung an.

In Tabelle 4.4 sind die Istgrößenverläufe für verschiedene Roboter und verschiedene Umgebungssteifigkeiten aufgetragen. Die Parameter des PPC wurden wie in Kapitel 3.3.3 zu $a = 2 \cdot 10^{-4}$, $b = 2 \cdot 10^{-5}$ und $D = 3$ gewählt. Das starke Sensorrauschen des Kraft-Momenten-Sensors *Dyn pick WEF-6A200* lässt eine numerische Ableitung der Kraft nicht in ausreichender Berech-

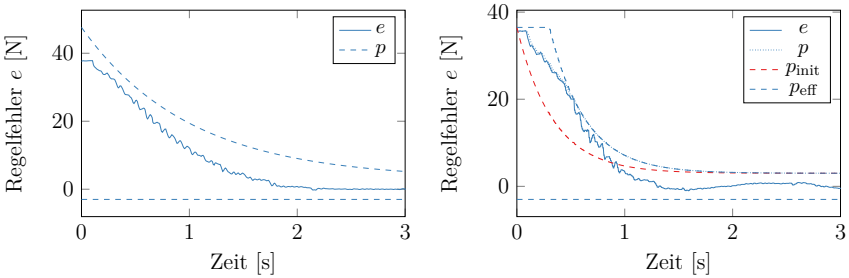


Abbildung 4.15: Darstellung des Regelfehlers e und der Performanzgrenzen p eines PPC Einregelvorgangs. Durch die Verschiebung der Performanzgrenzen p bei Verletzung der Grenzen ergibt sich eine effektive Performanzgrenze p_{eff} (Halt et al. 2019)

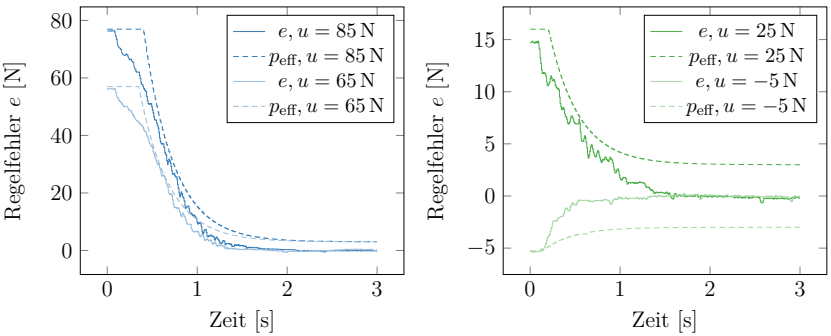


Abbildung 4.16: Darstellung des Regelfehlers e und der effektiven Performanzgrenze p_{eff} bei verschiedenen Sollkraftvorgaben u (Pan 2019)

nungszeit und Genauigkeit zu. Daher wurde für die Experimente mit dem Denso Roboter der Parameter $b = 0$ gesetzt und somit auf den differentiellen Anteil der PPC Regelung verzichtet.

Zunächst ist in den Verläufen aus Tabelle 4.4 zu erkennen, dass die Kontaktkraftregelung durch das PPC für alle Roboter und alle getesteten Umgebungssteifigkeiten zu einer schnellen Ausregelung auf die Führungsgröße sorgt, ohne dass ein Kontaktverlust auftritt. Die Einregelzeit wird erwartungsgemäß für höhere Steifigkeiten kürzer, da kleinere Bewegungen des Roboters zu größeren Auswirkungen auf die wirkende Kraft führen.

Die Istgrößenverläufe des UR5 und des Denso Roboters verlaufen bei den kleinen und mittleren Umgebungssteifigkeiten (1)-(4) vergleichbar. Durch die Vorgabe der identischen Performanzfunktion ergibt der jeweilige Verlauf und die Roboter sind in der Lage die Performanzgrenzen nicht zu verletzen. Eine Verschiebung der Performanzfunktion bzw. eine Anpassung der PPC Parameter ist bei dem Panda Roboter zu beobachten, da der Roboter etwas träger beschleunigt. Bei größeren Umgebungssteifigkeiten spielt die Steifigkeiten des Roboterarms eine entscheidende Rolle. Diese wird auch bei dem Panda Roboter sichtbar, welcher die größte intrinsische Nachgiebigkeit der hier betrachteten Roboterarme aufweist. Für große Umgebungssteifigkeiten (3)-(5) gleichen sich die Verläufe an, da die intrinsische Nachgiebigkeit der Roboterarme gravierend die Gesamtsteifigkeit des Systems bestimmt.

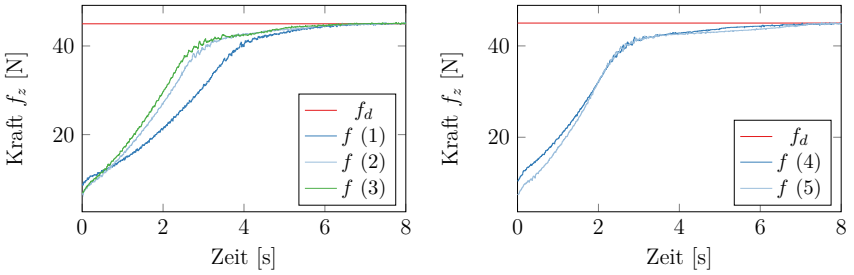
Insbesondere in dem Verläufen (1) und (2) des UR5 Roboters, aber auch im Verlauf (3) des Denso Roboters zeigen sich Auswirkungen von kurzzeitigem Verlust der Sensorkommunikation und somit das Verhalten des PPCs bei Messfehlern. Die gelesenen Sensordaten bleiben kurzzeitig auf einem konstanten numerischen Wert. Der Regler reagiert mit einer erhöhten Stellgrößenvorgabe, ohne dass ein unmittelbarer Effekt sichtbar wird. Werden nach einer kurzen Zeit wieder korrekte Sensordaten übermittelt, hat der Regelfehler seine Ideallinie innerhalb der Performanzfunktion überschritten, was an einem Überschwingen erkennbar ist. Das PPC reagiert entsprechend, um auf die Ideallinie innerhalb der Performanzfunktion zurückzukehren.

Für die größte Umgebungssteifigkeit (5) zeigt sich bei dem Verlauf des UR5 Roboters ein leichtes Überschwingen als Resultat der begrenzten Reaktionsfähigkeit des Reglers durch Totzeiten und Taktung, ausgelöst ebenfalls durch ein kurzzeitiges Ausbleiben der Sensordaten.

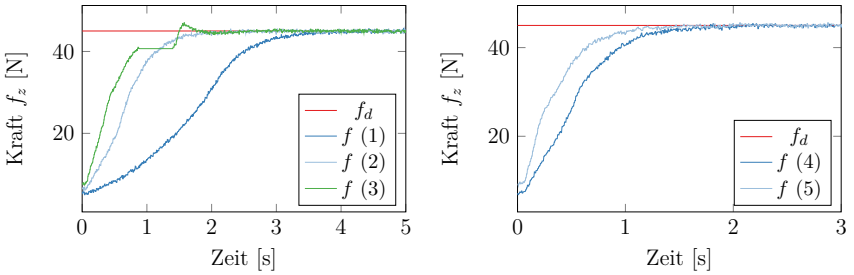
In Tabelle 4.5 wurden die Dauer der Stabilisierung für die verschiedenen Umgebungssteifigkeiten der Stahl-, Aluminium- und PVC-Platten mit den einzelnen Robotern zusammengetragen. Sowohl für verschiedene Umgebungssteifigkeiten mit dem gleichen Roboter, als auch mit verschiedenen Robotern für die gleichen Umgebungssteifigkeiten zeigen sich vergleichbare Zeiten um ca. 2 s. Für weiche Umgebungssteifigkeiten (bspw. durch die PVC-Platte) oder dem nachgiebigeren Armen (bspw. des Panda Roboters) fallen die Stabilisierungszeiten höher aus. Die

Tabelle 4.4: Kontaktkraftverläufe mittels PPC Regelung mit verschiedenen Robotern und verschiedenen Umgebungssteifigkeiten

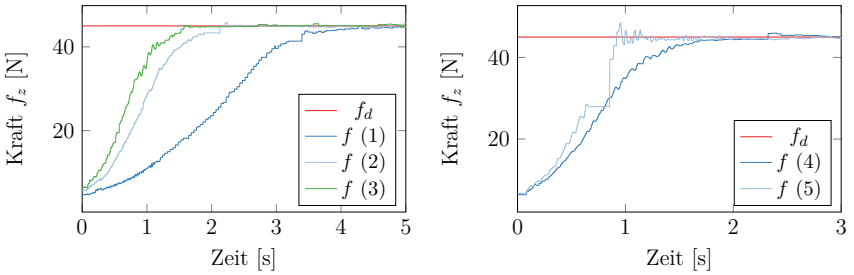
Panda



Denso



UR5



	Stahl	Aluminium	PVC	Mittelwert	Standardabw.
Panda	2.3 s	2.5 s	2.9 s	2.6 s	0.31 s
Denso	0.7 s	0.9 s	1.9 s	1.16 s	0.64 s
UR5	1.0 s	1.1 s	2.2 s	1.43 s	0.67 s
Mittelwert	1.37 s	1.47 s	2.33 s		
Standardabw.	0.68 s	0.8 s	0.26 s		

Tabelle 4.5: Dauer der Stabilisierung mit einer PPC Regelung

	Stahl	Aluminium	PVC	Mittelwert	Standardabw.
Panda	10.0 s	10.4 s	20.1	13.5 s	5.72 s
Denso	0.5 s	0.5 s	8.0 s	3.0 s	4.33 s
UR5	1.8 s	1.2 s	13.0 s	5.33 s	6.65 s
Mittelwert	4.1 s	4.03 s	13.7 s		
Standardabw.	5.15 s	5.52 s	6.08 s		

Tabelle 4.6: Dauer der Stabilisierung mit einem Impedanzregler

Standardabweichung bleibt unter dem maximalen Wert von 0.8 s. Dem PPC gelingt es ein vergleichbares Verhalten über die erprobten Variation von Roboter- und Umgebungssteifigkeiten aufzuprägen.

Um die Reglergüte des PPC vergleichen zu können wurde auf das in der Literatur verbreitete Verfahren der Impedanzregelung zurückgegriffen. Die gleichen Experimente wurden auch mit dem in Kapitel 4.1.3 vorgestellten Impedanzregler erster Ordnung durchgeführt. Zur Auslegung der Reglerparameter des Impedanzreglers wurde der UR5 mit der mittleren Umgebungssteifigkeit der Aluminiumplatte angenommen. Tabelle 4.6 fasst die Zeiten des Stabilisierungsvorgangs mit einem Impedanzregler zusammen. Sowohl die Mittelwerte als auch die Standardabweichungen sind deutlich höher und zeigen, dass ein fest ausgelegter Impedanzregler nicht dafür geeignet ist, vergleichbares Verhalten über große Variation von Roboterverhalten und Umgebungssteifigkeiten zu erreichen. Die Regelfehlerverläufe als Basis der Zeitmessung sind in Tabelle A.2 dargestellt.

Mit den durchgeführten Experimenten konnte die Eignung der PPC Regelung für reale Anwendungen verifiziert werden. Die Regelung zeigte sich übertragbar und robust für den Einsatz mit verschiedene Robotern und Umgebungssteifigkeiten. Die Standardabweichungen der Einregelzeiten über die verschiedenen Roboter und Steifigkeiten sind bedeutend kleiner als die zum Vergleich herangezogenen Zeiten eines Impedanzreglers. Damit kann die PPC Regelung als genereller Kraftregler eingesetzt werden.

Durch die Einbindung der PPC und der ARF Regelung in die entsprechenden Skills und deren Komposition zu Teil- und Gesamtapplikationen ergänzen sich die zuvor einzeln dargestellten Stärken und Einsatzgebiete zur praktischen Anwendung. In dem folgenden Abschnitt wird die Relevanz der bisherigen Untersuchung für praktische Anwendungsszenarien dargestellt.

4.3 Übertrag der Experimente auf praktische Anwendungsszenarien

Um das Kapitel abzuschließen werden im Folgenden die zuvor durchgeführten Experimente im Kontext von praktischen Anwendungen diskutiert. Als praktische Anwendungen werden die in Kapitel 2.2 eingeführten Anwendungen der Hutschienenmontage und der Türgriffmontage aufgegriffen.

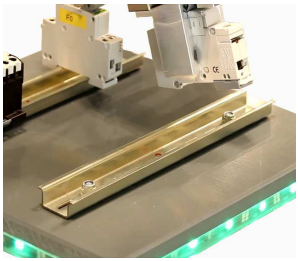
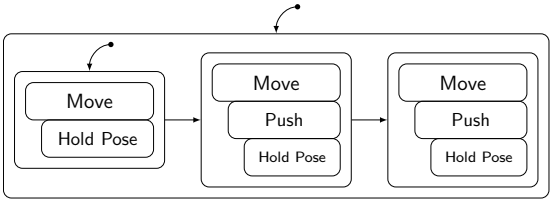
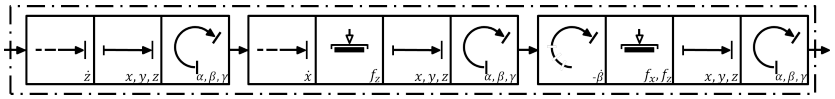
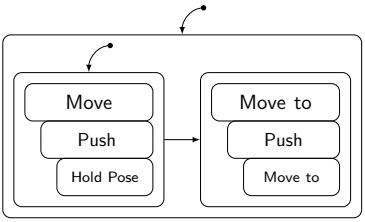
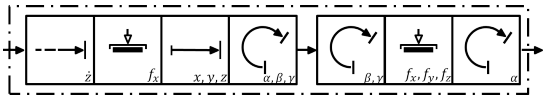
Die Prozessbeschreibung der Anwendungen auf Basis der VDI Richtlinie 2860 wurden in Kapitel 2 eingeführt und mithilfe von hierarchischen Kombinationen von elementaren Prozessen und zugehörigen Aufgabenraumkoordinaten in *Skills* überführt (Vgl. Tabelle 2.2). Ein *Skill* bezeichnet hierbei die Verkapselung von hierarchisch kombinierten Aufgabenraumkoordinaten $\boldsymbol{\chi}_{\mathrm{f}}$, deren Zwangsbedingungen, Abbruchbedingungen in Form von *Monitoren* sowie mögliche unterlagerte Kompositionen von *Subskills* (Vgl. Tabelle 2.1 für Kompositionen und Tabelle A.1 für Beispiele). Weiter beinhaltet ein *Skill* die Zuweisung welche Aufgabenraumkoordinate mithilfe von welchen Regler eingestellt werden soll.

In den vorherigen Unterkapiteln wurden die Abläufe der Experimente in Form von Zustandsmaschinen dargestellt. Diese Darstellung betont den prozeduralen Ablauf der Robotersteuerung, jedoch nicht die Geräteunabhängigkeit. Trotz der Unterschiede zwischen Prozessbeschreibung nach VDI Richtlinie 2860 und Zustandsmaschine sind beide Darstellungen qualitativ gleichwertig und können ineinander überführt werden.

In Tabelle 4.7 sind beide Darstellungsformen für die Hutschienenmontage und die Türgriffmontage nebeneinander gestellt. Auf die vollständige Darstellung des *pitasc* Knoten wurde hierbei verzichtet. Die Prozesse[7] Move to und Move to (rotational) wurden zu dem generellen Skill Move to in sechs Freiheitsgraden zusammengefasst. Dieser wurde weiter mit der Zwangsbedingung der Aufgabenraumkoordinaten $\boldsymbol{\chi}_{\mathrm{f}} = \mathbf{0}$ erweitert und als Hold Pose benannt. Beide Anwendungen dienen als repräsentative Beispiele für Kategorien des *Fügen durch Zusammensetzen* mit den Untergruppen *Einlegen*, *Ineinanderschieben* und *Einhängen* (DIN 8593-1 2003) sowie *Fügen*

[7] Der Begriff *Prozess* wird hier äquivalent für den Begriff *Skill* genutzt, um den Bezug auf die VDI Richtlinie 2860 zu betonen.

Tabelle 4.7: Nebeneinanderstellung der Prozessbeschreibung nach der VDI Richtlinie 2860 und der entsprechenden Zustandsmaschinendarstellung

Hutschienenmontage
  
Türgriffmontage
  

durch An- und Einpressen mit den Untergruppen *Schrauben*, *Verstiften*, *Verkeilen* und *Verspannen* (DIN 8593-3 2003). Somit wird ein Großteil der in DIN 8593 beschriebenen Vorgänge durch die beiden Beispiele abgedeckt.

Jeder Subskill weist dem iTaSC Algorithmus spezifische Aufgabenraumkoordinaten, Zwangsbedingungen und einen spezifischen Regler zu. Das bedeutet, dass der Skill Push Kraftregler und die Skills Move to bzw. Hold Pose Positionsregler zuweisen. Mit den erarbeiteten Regleransätzen der dynamischen Ausgangsrückführung ARF zur Positionsregelung und dem Regler mit vorgegebenem Performanzband PPC zur Kraftregelung, wirken sich die Eigenschaften der Regelung direkt auf den Anwendungsablauf aus.

Die PPC Regelung zeigte bei den untersuchten Robotern und Umgebungssteifigkeiten vergleichbares Abklingverhalten (Vgl. Tabelle 4.5). Die entwickelte Regelung bewirkt, dass sich das System vergleichbar verhält, weshalb sich das Roboterprogramm zwischen verschiedenen Anwendungen übertragen lässt. Als Subskill eines hierarchisch kombinierten Skills wird diese Übertragbarkeit an den zusammengesetzten Skill weitergegeben. Somit lässt sich auch der zusammengesetzte Skill zwischen Robotern und Anwendungen übertragen. Durch die Vorgabe eines Performanzbands kann die Reglerantwort im großen Maße angepasst werden. Kann das Performanzband nicht eingehalten werden und greifen die in Kapitel 3.3.2 beschriebenen Maßnahmen, ist eine höhere Reglerperformanz mit bspw. einem entsprechen eingestellten Impedanzregler zu erwarten.

Die PPC Regelung zeigt im Mittel mindestens eine Halbierung der Stabilisierungszeiten im Vergleich zu der Impedanzregelung unter den hier angenommenen Bedingungen der variablen Anwendungen und ausführenden Roboter. Das Applikationsbeispiel der Türgriffmontage und die zwei der drei Skills der Hutschienenmontage können durch den Einsatz des PPC in ihrer Ausführgeschwindigkeit mindestens verdoppelt werden. Eine genauere Einschätzung der Verbesserung hängt von dem eingesetzten Roboter und der Nachgiebigkeit der Klipsfeder des Hutschienenelementes bzw. Nachgiebigkeit der Aufhängung des Türschließmechanismus ab.

Neben der Übertragbarkeit bestimmt auch eine weitestgehend automatisierte Synthese für ein möglichst zeitoptimales Einregelverhalten das Entwurfsziel der ARF. Im gemeinsamen Einsatz mit der automatischen Modellapproximation mittels IEKF wurde zusätzlich die Modellbildung und somit der gesamte Entwurf automatisiert. Die ARF mittels IEKF ermöglicht einen sehr schnelles Einregelverhalten mit automatischer Synthese.

Der Einsatz der ARF in den Skills Hold Pose oder Move to in hierarchisch kombinierten Skills ist zwar möglich, jedoch handelt es sich meistens um sehr kleine Auslenkungen. Deshalb rechtfertigt der Aufwand einer Modellbildung den Zeitgewinn bei der Einregelung nicht. Der vorrangige Einsatz des Reglers bezieht sich auf reine Positions-Skills, in denen alle kartesischen

Freiheitsgrade auf eine Zielpose eingeregelt werden sollen. Sequenzen solcher Transferbewegungen bilden typischerweise einen großen Anteil der Roboterapplikationen. Die ARF bietet die Möglichkeit mittels externer Regelung diese Transferbewegungen mit vergleichbarer Performanz zur roboterinternen (nativen) Positionsregelung auszuführen. Die nativen Positionsregelungen für jeden Roboter in den jeweiligen Bezeichnern und Parametern unterschiedlich. Daher ist für die Übertragbarkeit der Programme der Einsatz eines generellen Reglers, wie bspw. die ARF, zu bevorzugen.

5 Zusammenfassung und Ausblick

Das Ziel der vorliegenden Arbeit war es ein performantes sowie roboter- und kinematikneutrales Steuerungsrahmenwerk für roboterbasierte Montage zu schaffen. Dieses Rahmenwerk sollte einfach zu programmieren sein und die Möglichkeit aufweisen erstellte Programme zwischen Robotern zu übertragen. Um dies zu erreichen wurden Reglerstrukturen entwickelt und untersucht, welche sich ohne Zutun des Programmierers eigenständig parametrisieren.

Die Durchführung dieser Arbeit basiert auf dem *pitasc* Robotersteuerungsrahmenwerk. *pitasc* besteht aus einer roboterneutralen Programmierschicht, einer kinematischen Abstraktionsschicht als Realisierung des iTaSC Formalismus und einer Regelungskomponente, welche in einem allgemeinen Aufgabenraum betrieben wird.

Ein herausstechendes Merkmal der Programmierschicht liegt in der hierarchischen Kombinierbarkeit (sog. Hierarchien) von einzeln spezifizierbaren Rahmenbedingungen und deren Ausführung im jeweiligen Aufgabenraum, sog. Skills. Durch die priorisierte Anordnung der Skills entstehen komplexe Roboterbewegungen als Ergebnis der bestmöglichen Einhaltung der Rahmenbedingungen. Jeder Einzelskill bewirkt die Einhaltung der für die Montage relevanten Beziehungen zwischen dem zu montierenden Bauteil und dem Montageziel. Solche Beziehung können bspw. Kontaktkräfte oder relative Bewegungen oder Verdrehungen sein. Der Applikationsprogrammierer beschreibt direkt die notwendigen Beziehungen und priorisiert diese zueinander in der Wichtigkeit der Einhaltung um etwaige Konflikte auflösen zu können. Das unterliegende System wandelt diese in Roboterbewegungen um, um die gegebenen Beziehungen einzuhalten.

Unter Betrachtung der VDI Richtlinie 2860 “Montage- und Handhabungstechnik, [...]” wurde dargelegt, dass durch entsprechende hierarchische Kombination von geschwindigkeits-, positions- und kraftgeregelten Skills eine weite Abdeckung relevanter Montageaufgaben abgedeckt werden kann. Durch den Einsatz des iTaSC Formalismus in der kinematischen Abstraktionsschicht wird eine allgemeine Transformation von beliebigen Aufgabenräumen in den Achswinkelraum des zu steuernden Roboters ermöglicht. Da die Kinematik in jeden Berechnungsschritt einbezogen wird muss diese zur Programmierzeit nicht betrachtet werden.

Zur Einregelung der geforderten Rahmenbedingungen im Aufgabenraum müssen die entsprechend transformierten Messwerte vorliegen. Notwendigerweise spezifizieren sich die Sollwerte

im Aufgabenraum direkt aus den Rahmenbedingungen. Der eingesetzte Regler erhält grundsätzlich keine weiteren Kenntnisse des zugrundeliegenden Aufgabenraums oder der vor- bzw. nachgelagerten Transformationen.

In den meisten Fällen liegt der relevante Messwert bereits in einem Darstellungsraum vor, welcher durch einfache lineare Transformation in dem gewünschten Aufgabenraum übertragen werden kann. Dies ist bspw. der Fall bei einer kartesischen Kraftregelung unter dem Einsatz eines 6D Kraft-Momenten-Sensors. Hierbei liegen die Messdaten bereits im kartesischen Kraftraum vor. Mit linearen Transformationen werden die Messdaten von dem Basiskoordinatensystem des Sensors auf einen (virtuellen) Messpunkt im kartesischen Aufgabenraum der Bewegung verschoben.

Der Fokus dieser Arbeit richtet sich auf die Untersuchung und Auslegung der Regelungskomponente unterhalb und im Zusammenspiel mit den zuvor beschriebenen Abstraktionsschichten. Hierbei wird ein besonderes Augenmerk darauf gerichtet, dass die Regelung hoch performant und durch einen weitestgehend automatischen Entwurf oder sich selbstständig anpassende Ansätzen zwischen verschiedenen Robotern übertragbar ist.

Zunächst wurde eine Modellidentifikation innerhalb des Aufgabenraums untersucht. Hierfür wurde ein identifizierendes erweitertes Kalman Filter (IEKF) eingesetzt. Der Ansatz dieses Filters erlaubt die Annäherung eines Zustandsraummodells unter der Vorgabe der Form der Systemmatrix $\boldsymbol{A}$. Daher eignet sich der Ansatz besonders zur Identifikation eines Modells für die Reglersynthese.

Es wurde eine modellflexible und performante Realisierung durch eine gemischte symbolisch-numerische Implementierung geschaffen. Die Konvergenz des Filters wurde anhand des geglätteten Modellfehlers angegeben. Zusätzlich zu dem Modellfehler wurde die Frobeniusnorm der Fehlerkovarianzmatrix herangezogen um sicherzustellen, dass der Modellfehler nicht aufgrund eines Stillstands des Systems abklingt. Der Einfluss von Rauschen, der Sequenzierung der Eingangsdatenpakete sowie der Einstellparameter wurde untersucht, wobei sich das Filter robust gegenüber den Störeinflüssen zeigte. Ein alternativer Ansatz zur Annäherung eines zeitverzögerten PT2-Systems wurde präsentiert und als Initialisierung des IEKF eingesetzt.

Unter Einsatz des identifizierten Modells wurde ein modellbasierter Regelungsentwurfsansatz vorgestellt. Die entsprechende Ausgangsrückführung (ARF) ergibt sich als Ergebnis eines quasikonvexen LMI Optimierungsproblems. Verfügbare Solver für LMI Probleme erlauben ein effizientes Finden einer globalen Lösung unabhängig der im Solver eingesetzten Lösungsmethode.

Die grundsätzliche Idee der ARF basiert auf der Synthese einer Zustandsraumregler-Beobachter Struktur. Die strukturellen Vorgaben wurden zur Erweiterung des Suchraums und für eine

konvexe Formulierung aufgelöst. Die sich ergebende interne Struktur ist somit ebenfalls Teil der Optimierungsaufgabe. Durch die in der Optimierung einbezogenen Randbedingungen wird ein Regler gefunden, welcher trotz des Einflusses einer Stellgrößensättigung stabil im Sinne von Ljapunov ist. Die ARF erreicht in den Simulationen nahezu zeitoptimale Regelverläufe. Die ARF passt sich gut in die Anforderungen der roboterneutralen Programmierung ein, da sehr gute Regelungsergebnisse bei nahezu vollständig automatischem Entwurf möglich sind. Eine synthetisierte ARF kann allerdings nicht auf einen anderen Roboter übertragen werden.

Die Einführung eines Vorfilters und die Zeitdiskretisierung des Reglers waren zum praktischen Einsatz notwendig. Ebenso wurden die Einflüsse der freien Parameter untersucht und praktische Handhabungsregeln abgeleitet. In Experimenten an verschiedenen Industrierobotern wurde zum einen die ARF mit gegebenen Modell und zum anderen im direkten Zusammenspiel mit einer kontinuierlichen Modellschätzung mittels IEKF erprobt. Die ARF eignet sich bevorzugt zur reinen Positionsregelung und bietet hierbei eine Alternative zu den roboterinternen Positionsreglern. Für eine reine Anwendung der Regelung zur linearen Roboterbewegung sollte dem ARF eine Vorsteuerung mit spezifischen Beschleunigungs- und Bremsrampen vorgelagert werden. Da eine solche Vorsteuerung die Brücke zwischen Roboterwegplanung und rein reaktiver Regelung spannt, wurde hiervon in dieser Arbeit abgesehen.

Die ARF liefert in der Simulation erfolgversprechende Ergebnisse. Bei der Umsetzung auf dem realen Roboter bleiben die Ergebnisse allerdings durch die Neigung zum Überschwingen hinter den Erwartungen zurück. Darüber hinaus wird der Nutzen durch den vergleichsweise hohen Zeitaufwand der Modellbildung deutlich relativiert. Für die Übertragbarkeit von Programmen ist es allerdings vorteilhaft nicht auf interne roboterspezifische Regelungen zurückzugreifen, sondern einen für alle Roboter gleichen Regelungsansatz zu verfolgen. Aus dieser Perspektive kann sich die ARF gut gegenüber anderen Regelungskonzepten im Hinblick auf Regelungsgüte, automatischer Synthese und dem Einbezug von Stellgrößensättigung positionieren.

Zur Regelung von kraftbasierten Skills wurde ein anderer Ansatz verfolgt. Da Kräfte nur bei Kontakt zwischen dem zu steuernden Robotern und der Umgebung entstehen, bergen Experimente zur Ermittlung eines Modells ein großes Risiko, dass das Robotersystem und das Produkt beschädigt werden. Daher wurde für die Regelung von Kräften ein modellfreier Ansatz verfolgt. Ein *Regler mit vorgegebenem Performanzband* (PPC) bezeichnet einen P, bzw. PD Regleransatz mit regelungsfehlerabhängigen Verstärkungsfaktoren. Verlässt der zeitliche Verlauf des Regelungsfehlers die Ideallinie eines vorgegebenem Bands, so werden die Verstärkungsfaktoren derart angepasst, dass zum Idealverlauf zurückgekehrt wird.

Durch die Einpassung des PPC in den Aufgabenraum ergeben sich strukturelle Veränderungen gegenüber der in (Kanakis et al. 2018) vorgeschlagenen Form. Für die veränderte Form wurde ein

Stabilitätsnachweis durchgeführt. Kann die PPC Regelung die vorgegebenen Performanzgrenzen nicht einhalten, so entsteht ein Widerspruch in der Voraussetzung der PPC Transformation und die Regelung muss gestoppt werden. Daher wurden Maßnahmen ergriffen, um den PPC gegen die möglichen Gründe einer Verletzung der Performanzgrenzen abzusichern. Zu schwach eingestellte Verstärkungsfaktoren werden automatisch erhöht und die Performanzgrenze wird so verschoben, dass die Regelung weiterhin betrieben werden kann. Obgleich die Notwendigkeit der Maßnahmen auf eine fehlerhafte Bedienung hinweist kann der PPC praktisch nicht ohne die Absicherungen betrieben werden. Eine Benachrichtigung bei Auslösen der Maßnahmen dient dem Bediener als Anhaltspunkt für weitere Roboterprogrammanpassungen.

Die verschiedenen Regelungsansätze wurden jeweils anhand von Simulationen erprobt und folgend in Experimenten mit unterschiedlichen Industrierobotern verifiziert. Die aus den Simulationen hervorgegangen Erkenntnisse spiegeln heuristische Umgangs- und Einstellregeln wider, welche in den Experimenten Anwendung fanden. Alle präsentierten Ansätze beweisen sowohl in Verbindung mit *pitasc*, als auch als Einzeltechnologien in anderen Anwendungsgebieten der Roboter hohes Potenzial. Daher sollten in weiterführenden Arbeiten diese Technologien aufgegriffen und erweitert werden.

Die LMI Formulierung des Syntheseproblems der ARF bietet die Möglichkeit, weitere Randbedingungen einzuführen und so höhere Robustheit gegenüber Schwankungen der Taktzeit (engl.: jitter) oder Quantifizierung einzuführen. Eine LMI Formulierung zum Entwurf einer diesbezüglich robusten Regelung wurde bspw. in (Fridman et al. 2009) vorgestellt. Eine stärkere Unterdrückung des Überschwingverhaltens kann mit anderen Bedingungen zur Begrenzung der platzierten Eigenwerte, wie bspw. in (Jasniewicz 2010) erreicht werden.

Im Falle des PPC sollten weitere Untersuchungen das Verhalten bei Verletzung der Performanzgrenzen analysieren. Zur praktischen Anwendung sollte der Entwurf für zeitdiskrete Regelungen erweitert werden (Vgl. Rovithakis 2018). Im Hinblick auf das Gesamtsystem und Zusammenspiel verschiedener Skills sollte der gegenseitige Einfluss einzelner Regler innerhalb einer hierarchischen Skillkomposition untersucht werden.

A Anhang

A.1 Modale Transformation

Da die modale Transformation in der Regelungstechnik nicht verbreitet ist, soll diese im Folgenden kurz dargestellt werden.

Ziel der modalen Transformation ist es, die Systemmatrix $\boldsymbol{A}$ einer Zustandsraumdarstellung,

$$\begin{cases} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \ , \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} \ , \end{cases} \tag{A.1}$$

in eine diagonale, bzw. blockdiagonale Form $\boldsymbol{\Lambda}$ zu transformieren.

Die Hauptdiagonale von $\boldsymbol{\Lambda}$ beinhaltet die reellen Eigenwerte von $\boldsymbol{A}$. Bei konjugiert komplexen Eigenwerten, bilden die um ± 1 verschobenen Nebendiagonalen von $\boldsymbol{\Lambda}$ die imaginären Anteile der Eigenwerte ab.

Eine entsprechende invertierbare Transformationsmatrix $\boldsymbol{E}$ sodass,

$$\boldsymbol{\Lambda} = \boldsymbol{E}^{-1}\boldsymbol{A}\boldsymbol{E} \ , \tag{A.2}$$

wird für rein reelle Eigenwerte mit der Matrix $\boldsymbol{E}$ gefunden, welche reihenweise aus den Eigenvektoren der Matrix $\boldsymbol{A}$ zusammengesetzt wird. Die Reihenfolge der Eigenvektoren bestimmt die Reihenfolge der Eigenwerte auf der Hauptdiagonalen der Matrix $\boldsymbol{\Lambda}$. Da die Lösungsverfahren der Eigenwerte und Eigenvektoren im Allgemeinen keine geordnete Reihenfolge vorgeben, kann es sinnvoll sein die Eigenvektoren entsprechend der Größe der Eigenwerte zu sortieren.

Für konjugiert komplexe Eigenwertpaare werden die jeweils komplexen Eigenvektoren in einen Vektor des Realteils und einen Vektor des Imaginärteils aufgeteilt. Die Transformationsmatrix $\boldsymbol{E}$ wird um zwei Reihen mit den Einträgen der aufgeteilten Eigenvektoren zusammengesetzt. Da die zwei zugefügte Reihen einem Eigenwertpaar entsprechen, bleibt die Dimension von $\boldsymbol{E}$ erhalten.

Die transformierte Zustandsraumdarstellung mit dem Zustandsvektor $\boldsymbol{z} = \boldsymbol{E}\boldsymbol{x}$ ergibt sich zu,

$$\begin{cases} \dot{\boldsymbol{z}} &= \boldsymbol{E}^{-1}\boldsymbol{A}\boldsymbol{E}\boldsymbol{z} + \boldsymbol{E}^{-1}\boldsymbol{B}\boldsymbol{u} \ , \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{E}\boldsymbol{z} + \boldsymbol{D}\boldsymbol{u} \ . \end{cases} \tag{A.3}$$

A.2 Diskretisierung der Zustandsraumstruktur der Ausgangsrückführung

Die Diskretisierung der Zustandsraumstruktur einer Ausgangsrückführung folgt prinzipiell dem grundsätzlichen Vorgehen der Überführung einer zeitkontinuierlichen Zustandsraumdarstellung in eine zeitdiskreten Zustandsraumdarstellung. Da die Systemstruktur der Ausgangsrückführung von einer üblichen Zustandsraumdarstellung abweicht, sei im Folgenden die Diskretisierung nochmals gezeigt.

Zugrunde liegt das zeitkontinuierliche Regelungssystem[8] K,

$$K : \begin{cases} \dot{\boldsymbol{z}}(t) &= \boldsymbol{A}\boldsymbol{z}(t) + \boldsymbol{B}\boldsymbol{y}(t) + \boldsymbol{E}\boldsymbol{u}(t) \ , \\ \boldsymbol{u}(t) &= \boldsymbol{C}\boldsymbol{z}(t) + \boldsymbol{D}\boldsymbol{y}(t) \ . \end{cases} \tag{A.4}$$

Gesucht wird das entsprechende zeitdiskrete Zustandsraumsystem $\overline{K}$ mit den zu bestimmenden Matrizen $\overline{\boldsymbol{A}}$, $\overline{\boldsymbol{B}}$, und $\overline{\boldsymbol{E}}$,

$$\overline{K} : \begin{cases} \boldsymbol{z}_{r+1} &= \overline{\boldsymbol{A}}\boldsymbol{z}_r + \overline{\boldsymbol{B}}\boldsymbol{y}_r + \overline{\boldsymbol{E}}\boldsymbol{u}_r \ , \\ \boldsymbol{u}_r &= \boldsymbol{C}\boldsymbol{z}_r + \boldsymbol{D}\boldsymbol{y}_r \ . \end{cases} \tag{A.5}$$

Es werden die Annahmen getroffen, dass die Größen $\boldsymbol{y}(t)$ und $\boldsymbol{u}(t)$ zwischen jeden Zeitschritt r konstant verlaufen. Es gilt somit,

$$\boldsymbol{y}(t) = \boldsymbol{y}_r \, , \ \text{für} \ \ rt \leq t \leq (r+1)t \ , \tag{A.6}$$

$$\boldsymbol{u}(t) = \boldsymbol{u}_r \, , \ \text{für} \ \ rt \leq t \leq (r+1)t \ . \tag{A.7}$$

[8] Zur besseren Lesbarkeit wurde, im Vergleich zu Gleichung (3.26), der Index der Größe $\boldsymbol{u}_c$, sowie der Index K und das Sättigungsglied ausgelassen.

Die Zustandsübergangsgleichung von K wird mit dem Matrixexponential $\boldsymbol{e}^{-\boldsymbol{A}t}$ beidseitig erweitert und umgestellt,

$$\begin{aligned}
\boldsymbol{e}^{-\boldsymbol{A}t}\dot{\boldsymbol{z}}(t) &= \boldsymbol{e}^{-\boldsymbol{A}t}\boldsymbol{A}\boldsymbol{z}(t) + \boldsymbol{e}^{-\boldsymbol{A}t}\boldsymbol{B}\boldsymbol{y}(t) + \boldsymbol{e}^{-\boldsymbol{A}t}\boldsymbol{E}\boldsymbol{u}(t) \ , \\
\boldsymbol{e}^{-\boldsymbol{A}t}\dot{\boldsymbol{z}}(t) - \boldsymbol{e}^{-\boldsymbol{A}t}\boldsymbol{A}\boldsymbol{z}(t) &= \boldsymbol{e}^{-\boldsymbol{A}t}\boldsymbol{B}\boldsymbol{y}(t) + \boldsymbol{e}^{-\boldsymbol{A}t}\boldsymbol{E}\boldsymbol{u}(t) \ , \\
\frac{d}{dt}\left(\boldsymbol{e}^{-\boldsymbol{A}t}\boldsymbol{z}(t)\right) &= \boldsymbol{e}^{-\boldsymbol{A}t}\boldsymbol{B}\boldsymbol{y}(t) + \boldsymbol{e}^{-\boldsymbol{A}t}\boldsymbol{E}\boldsymbol{u}(t) \ . \qquad \text{(A.8)}
\end{aligned}$$

Integration von Gleichung (A.8) von Zeitpunkt t_0 bis t ergibt,

$$\boldsymbol{e}^{-\boldsymbol{A}t}\boldsymbol{z}(t) - \boldsymbol{e}^{-\boldsymbol{A}t_0}\boldsymbol{z}(t_0) = \int_{t_0}^{t} \boldsymbol{e}^{-\boldsymbol{A}\tau}\boldsymbol{B}\boldsymbol{y}(\tau)d\tau + \int_{t_0}^{t} \boldsymbol{e}^{-\boldsymbol{A}\tau}\boldsymbol{E}\boldsymbol{u}(\tau)d\tau \ . \qquad \text{(A.9)}$$

Durch sukzessives Zusammenfassen und Umstellung der Gleichung (A.9) nach $\boldsymbol{z}(t)$ ergibt sich

$$\begin{aligned}
\boldsymbol{z}(t) - \boldsymbol{e}^{\boldsymbol{A}(t-t_0)}\boldsymbol{z}(t_0) &= \int_{t_0}^{t} \boldsymbol{e}^{\boldsymbol{A}(t-\tau)}\boldsymbol{B}\boldsymbol{y}(\tau)d\tau + \int_{t_0}^{t} \boldsymbol{e}^{\boldsymbol{A}(t-\tau)}\boldsymbol{E}\boldsymbol{u}(\tau)d\tau \ , \\
\boldsymbol{z}(t) &= \boldsymbol{e}^{\boldsymbol{A}(t-t_0)}\boldsymbol{z}(t_0) + \int_{t_0}^{t} \boldsymbol{e}^{\boldsymbol{A}(t-\tau)}\boldsymbol{B}\boldsymbol{y}(\tau)d\tau + \int_{t_0}^{t} \boldsymbol{e}^{\boldsymbol{A}(t-\tau)}\boldsymbol{E}\boldsymbol{u}(\tau)d\tau \ . \qquad \text{(A.10)}
\end{aligned}$$

Mit Gleichung (A.10), den Annahmen (A.6) und (A.7), sowie der Spezifizierung des Zeitpunktes t_0 als aktueller Zeitschritt r und des Zeitschrittes $t = T$ als nächster Zeitschritt $r + 1$ mit der Abtastzeit T, ergibt sich,

$$\boldsymbol{z}_r = \boldsymbol{z}(t_0) \ , \qquad \text{(A.11)}$$

$$\boldsymbol{z}_{r+1} = \boldsymbol{z}(t = T) \ , \qquad \text{(A.12)}$$

$$\boldsymbol{z}_{r+1} = \boldsymbol{e}^{\boldsymbol{A}T}\boldsymbol{z}_r + \int_{rT}^{(r+1)T} \boldsymbol{e}^{\boldsymbol{A}((r+1)T-\tau)}d\tau\boldsymbol{B}\boldsymbol{y}_r + \int_{rT}^{(r+1)T} \boldsymbol{e}^{\boldsymbol{A}((r+1)T-\tau)}d\tau\boldsymbol{E}\boldsymbol{u}_r \ . \qquad \text{(A.13)}$$

Die zeitdiskrete Zustandsraumdarstellung $\overline{K}$ kann mit der Substitution

$$\nu = (r+1)T - \tau \ , \ d\tau = -d\nu$$

und Tauschen der Integrationsgrenzen wie folgt gewonnen werden,

$$\boldsymbol{z}_{r+1} = \underbrace{\boldsymbol{e}^{\boldsymbol{A}T}}_{\overline{\boldsymbol{A}}}\boldsymbol{z}_r + \underbrace{\int_0^T \boldsymbol{e}^{\boldsymbol{A}\nu}d\nu\boldsymbol{B}}_{\overline{\boldsymbol{B}}}\boldsymbol{y}_r + \underbrace{\int_0^T \boldsymbol{e}^{\boldsymbol{A}\nu}d\nu\boldsymbol{E}}_{\overline{\boldsymbol{E}}}\boldsymbol{u}_r \ , \qquad \text{(A.14)}$$

mit den jeweiligen Lösungen der Integrale und der Einheitsmatrix $\boldsymbol{I}$,

$$\overline{\boldsymbol{A}} = \boldsymbol{e}^{\boldsymbol{A}T} \ , \tag{A.15}$$

$$\overline{\boldsymbol{B}} = \boldsymbol{A}^{-1} \left(\boldsymbol{e}^{\boldsymbol{A}T} - \boldsymbol{I} \right) \boldsymbol{B} \ , \tag{A.16}$$

$$\overline{\boldsymbol{E}} = \boldsymbol{A}^{-1} \left(\boldsymbol{e}^{\boldsymbol{A}T} - \boldsymbol{I} \right) \boldsymbol{E} \ . \tag{A.17}$$

Ergebnis A.15 und A.16 sind identisch zu der klassischen Diskretisierung eines Zustandsraumsystems und Ergebnis A.16 analog zu A.17. Daher können vorhandene Softwarewerkzeuge verwendet werden und bspw. eine Funktion zu Diskretisierung `Ad, Bd = discrete(A, B)` mit `Ad, Bd = discrete(A, B)` und `Ad, Ed = discrete(A, E)` aufgerufen werden.

A.3 Pseudocode

Das folgende Kapitel besteht aus Algorithmen, welche in Pseudocode dargestellt wurden. Die Algorithmen beziehen sich hierbei auf die relevanten Kapitel und stehen ohne weitere Erläuterungen.

Algorithm 1 Bisektionsalgorithmus (für Maxima)

```
1: procedure BISECT MAX
2:     problem(p) ← Optimierungsproblem mit Bisektionsvariable p
3:     l ← Initialwerte der unteren Grenze
4:     u ← Initialwerte der oberen Grenze
5:     bisection_tol ← Toleranzwert zum Abbruch
6:
7: verify upper bound:
8:     while Solve(problem(p = u)).status == 'feasible' do
9:         l ← u
10:        u ← 2 ∗ u
11:    end while
12:
13: bisect:
14:    while u − l >= bisection_tol do
15:        p ← (l + u)/2
16:        sol ← Solve(problem(p))
17:        if sol.status == 'feasible' then
18:            l ← p
19:            solution ← sol
20:        else
21:            u ← p
22:        end if
23:    end while
24:    return solution
25: end procedure
```

Algorithm 2 Identifying Extended Kalman Filter

procedure IEKF
    $\mathrm{s}, \mathrm{w} \leftarrow$ Anfangswerte der (konjugiert komplexen) System Polpaare

init:
    $(\sigma_i, \omega_i) \leftarrow$ Initialisiere symbolische Repräsentation
    $\boldsymbol{A}, \mathrm{A0} \leftarrow$ Initialisiere symbolische und numerische Repräsentation
    $\boldsymbol{B}, \mathrm{B0} \leftarrow$ Initialisiere symbolische und numerische Repräsentation
    $\boldsymbol{C}, \mathrm{C0} \leftarrow$ Initialisiere symbolische und numerische Repräsentation
    $\boldsymbol{D}, \mathrm{D0} \leftarrow$ Initialisiere symbolische und numerische Repräsentation
    $\boldsymbol{x}, \mathrm{x0} \leftarrow$ Initialisiere symbolische und numerische Repräsentation
    $\boldsymbol{u} \leftarrow$ Initialisiere symbolische Repräsentation
    $\mathrm{P} \leftarrow$ Initialisiere numerische Repräsentation
    $\mathrm{Q} \leftarrow$ Initialisiere numerische Repräsentation
    $\mathrm{z}_k \leftarrow$ x0, s, w, B0, C0, D0

prepare symbols:
    $\boldsymbol{z} \leftarrow \boldsymbol{x}, \sigma_i, \omega_i, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$
    $\boldsymbol{f} = [\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}, 0, \cdots, 0]$
    $\boldsymbol{h} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u}$
    $\boldsymbol{F} = \dfrac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}}$
    $\boldsymbol{H} = \dfrac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}}$

prepare numeric handles:
    $\mathrm{f}() \leftarrow \mathrm{lambdify}(\boldsymbol{f}, \boldsymbol{z})$
    $\mathrm{F}() \leftarrow \mathrm{lambdify}(\boldsymbol{F}, \boldsymbol{z})$
    $\mathrm{h}() \leftarrow \mathrm{lambdify}(\boldsymbol{h}, \boldsymbol{z})$
    $\mathrm{H}() \leftarrow \mathrm{lambdify}(\boldsymbol{H}, \boldsymbol{z})$
    $\mathrm{x}() \leftarrow \mathrm{lambdify}(\boldsymbol{x}, \boldsymbol{z})$
    $\mathrm{C}() \leftarrow \mathrm{lambdify}(\boldsymbol{C}, \boldsymbol{z})$
    $\mathrm{D}() \leftarrow \mathrm{lambdify}(\boldsymbol{D}, \boldsymbol{z})$

estimation:
    while estimate **do**
        $\mathrm{y} \leftarrow \mathrm{C}(\mathrm{z}_k)\mathrm{x}(\mathrm{z}_k) + \mathrm{D}(\mathrm{z}_k)\mathrm{u}_k$
        $\mathrm{K} \leftarrow \mathrm{PH}(\mathrm{z}_k)^\intercal[\mathrm{H}(\mathrm{z}_k)\mathrm{P}]^{-1}\mathrm{H}(\mathrm{z}_k)^\intercal + \mathrm{R}$
        $\mathrm{P} \leftarrow [\mathrm{I} - \mathrm{KH}(\mathrm{z}_k)]\mathrm{P}$
        $\mathrm{P} \leftarrow \mathrm{P} + T \cdot [\mathrm{F}(\mathrm{z}_k)\mathrm{P} + \mathrm{PF}(\mathrm{z}_k{}^\intercal) + \mathrm{Q}]$
        $\mathrm{z}_k \leftarrow \mathrm{z}_k + T \cdot \mathrm{f}(\mathrm{z}_k) + \mathrm{K}(\mathrm{y} - \mathrm{h}(\mathrm{z}_k))$
    end while
end procedure

A.4 Tabellen und Graphen

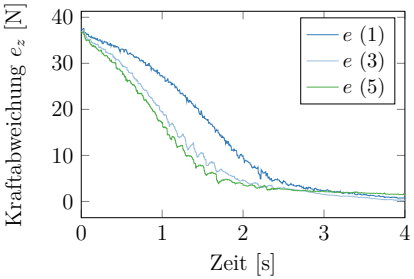
Tabelle A.1: Liste aus elementaren und zusammengesetzten Skills nach (Halt et al. 2018b)

Skill-Name	Subskills	Monitore	Verhalten
Move	-	-	Lineare kartesische Bewegung mit gegebener Geschwindigkeit
Hold pose	-	-	Beibehalten der aktuellen Position
Push	-	-	Aufbringen einer gegebenen Kraft
Move to	Move	Position Monitor	Lineare kartesische Bewegung zu einer gegebenen Zielposition
Move into contact	Move, Hold pose	Kraft Monitor	Lineare kartesische Bewegung bis eine Kraftschwelle überschritten wird
Slide into contact	Move into contact, Push, Hold pose	Kraft Monitor	Entlanggleiten auf der Oberfläche eines Objektes mit gegebener Anpresskraft bis eine Kraftschwelle in der Bewegungsrichtung überschritten wird
Rotational snap	Move (rotational), Push, Hold pose	Kraft Monitor	Drehen eines gegriffenen Objektes unter gleichzeitigen Einhalten von Kontaktkräften mit bspw. einer Hutschiene. Die Bewegung endet, wenn eine Schwelle für das aufgebrachte Moment in Drehrichtung überschritten wird

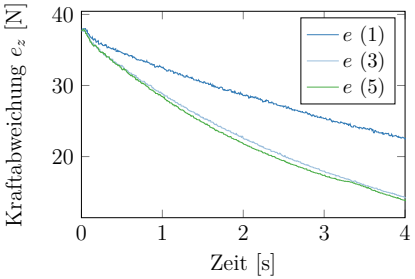
Tabelle A.2: Gegenüberstellung des Regelfehlerverlaufs einer PPC Regelung und einer Impedanzregelung mit verschiedenen Robotern und verschiedenen Umgebungssteifigkeiten

Panda

PPC

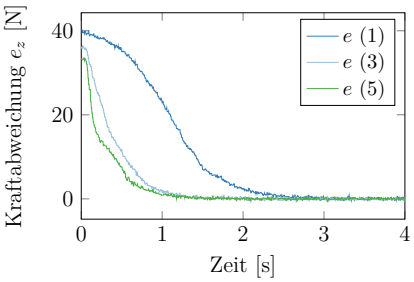


Impedanzregelung

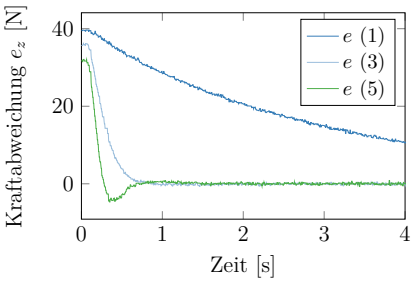


Denso

PPC

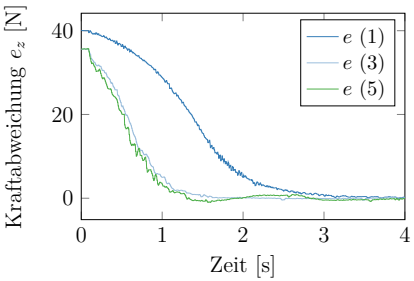


Impedanzregelung

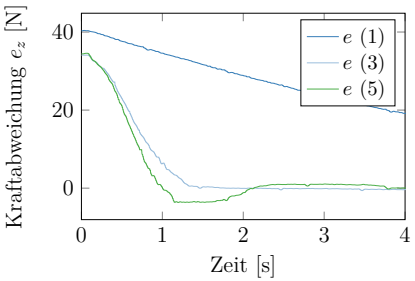


UR5

PPC



Impedanzregelung



Literatur

Abid et al. 2013

Abid, Fatma; Chevallier, G.; Blanchard, J. L.; Dion, J. L.; Dauchez, N., 2013.
System Identification Using Kalman Filters.
In: Allemang, Randall J.; Simmermacher, T.; Catbas, F. Necati; Cunha, Alvaro; Mayes, R.; Carrella, Alex; Adams, Douglas E.; Kerschen, Gaetan (Hrsg.): *Proceedings of the 31st IMAC, a conference on structural dynamics*,
S. 561–573.
DOI: 10.1007/978-1-4614-6585-0_55

Aertbeliën et al. 2014

Aertbeliën, Erwin; Schutter, Joris De, 2014.
eTaSL/eTC: A constraint-based task specification language and robot controller using expression graphs.
In: *2014 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*,
S. 1540–1546.
DOI: 10.1109/IROS.2014.6942760

Akan et al. 2011

Akan, Batu; Ameri, Afshin; Curuklu, Baran; Asplund, Lars, 2011.
Intuitive industrial robot programming through incremental multimodal language and augmented reality.
In: *2011 IEEE Int. Conf. on Robotics and Automation (ICRA)*,
S. 3934–3939.
DOI: 10.1109/ICRA.2011.5979887

Artiminds 2019

Artiminds, 2019.
Robot Programming Suite,
Zugriff am: 25. 06. 2019.
Verfügbar unter: https://www.artiminds.com/artiminds-rps/

Bechlioulis et al. 2010
Bechlioulis, Charalampos P.; Doulgeri, Zoe; Rovithakis, George A., 2010.
Neuro-Adaptive Force/Position Control with Prescribed Performance and Guaranteed Contact Maintenance.
2010 IEEE Transactions on Neural Networks, **21** (12), S. 1857–1868.
DOI: 10.1109/TNN.2010.2076302

Bechlioulis et al. 2008
Bechlioulis, Charalampos P.; Rovithakis, George A., 2008.
Robust Adaptive Control of Feedback Linearizable MIMO Nonlinear Systems With Prescribed Performance.
2008 IEEE Transactions on Automatic Control, **53** (9), S. 2090–2099.
DOI: 10.1109/TAC.2008.929402

Beetz et al. 2018
Beetz, Michael; Bessler, Daniel; Haidu, Andrei; Pomarlan, Mihai; Bozcuoglu, Asil Kaan; Bartels, Georg, 2018.
Know Rob 2.0 — A 2nd Generation Knowledge Processing Framework for Cognition-Enabled Robotic Agents.
In: *2018 IEEE Int. Conf. on Robotics and Automation (ICRA)*,
S. 512–519.
DOI: 10.1109/ICRA.2018.8460964

Behrens et al. 2018
Behrens, Rol; Belov, Anton; Poggendorf, Maik; Penzlin, Felix; Hanses, Magnus; Jantz, Emily; Elkmann, Norbert, 2018.
Performance Indicator for Benchmarking Force-Controlled Robots.
In: *2018 IEEE Int. Conf. on Robotics and Automation (ICRA)*,
S. 1653–1660.
DOI: 10.1109/ICRA.2018.8460858

Best et al. 2017
Best, Matthew C.; Bogdanski, Karol, 2017.
Extending the Kalman filter for structured identification of linear and nonlinear systems.
International Journal of Modelling, Identification and Control, **27** (2), S. 114–124.
DOI: 10.1504/IJMIC.2017.082952

Best et al. 2007
Best, Matthew C.; Newton, A. P.; Tuplin, S., 2007.
The identifying extended Kalman filter: Parametric system identification of a vehicle handling model.
Proceedings of the Institution of Mechanical Engineers, Part K: Journal of Multi-body Dynamics, **221** (1), S. 87–98.
DOI: 10.1243/14644193JMBD68

Bischoff et al. 2010
Bischoff, Rainer et al., 2010.
The KUKA-DLR Lightweight Robot arm-a new reference platform for robotics research and manufacturing.
In: *2010 Int. Symposium on Robotics (ISR)*,
S. 1–8

Bischoff et al. 2002
Bischoff, Rainer; Kazi, Arif.; Seyfarth, Markus, 2002.
The MORPHA style guide for icon-based programming.
In: *2002 IEEE Int. Workshop on Robot and Human Interactive Communication (ROMAN)*,
S. 482–487.
DOI: 10.1109/ROMAN.2002.1045668

Borghesan et al. 2012
Borghesan, Gianni; Willaert, Bert; Schutter, Joris De, 2012.
A constraint-based programming approach to physical human-robot interaction.
In: *2012 IEEE Int. Conf. on Robotics and Automation (ICRA)*,
S. 3890–3896.
DOI: 10.1109/ICRA.2012.6224943

Boyd et al. 1994
Boyd, Stephen P.; El Ghaoui, Laurent; Feron, Eric; Balakrishnan, Venkataramanan, 1994.
Linear Matrix Inequalities in System and Control Theory.
Society for Industrial and Applied Mathematics.
DOI: 10.1137/1.9781611970777

Boyd et al. 2015
Boyd, Stephen P.; Vandenberghe, Lieven, 2015.
Convex optimization.
Cambridge: University Press.
ISBN 978-0-521-83378-3

Bruyninckx et al. 2001
Bruyninckx, Herman; Lefebvre, Tine; Mihaylova, L.; Staffetti, E.; Schutter, Joris De; Xiao, Jing, 2001.
A roadmap for autonomous robotic assembly.
In: *2001 IEEE Int. Symposium on Assembly and Task Planning (ISATP2001)*,
S. 49–54.
DOI: 10.1109/ISATP.2001.928965

Bruyninckx et al. 1996
Bruyninckx, Herman; Schutter, Joris De, 1996.
Specification of force-controlled actions in the "task frame formalism" - a synthesis.
IEEE Transactions on Robotics and Automation, **12** (4), S. 581–589.
DOI: 10.1109/70.508440

Caine et al. 1989
Caine, Michael E.; Lozano-Perez, Tomás; Seering, Warren P., 1989.
Assembly strategies for chamferless parts.
In: *1989 IEEE Int. Conf. on Robotics and Automation (ICRA)*,
S. 472–477.
DOI: 10.1109/ROBOT.1989.100031

Chiaverini et al. 1997
Chiaverini, S.; Siciliano, Bruno; Villani, Luigi, 1997.
Parallel force/position control with stiffness adaptation.
In: *1997 IEEE Int. Conf. on Robotics and Automation (ICRA)*,
S. 1136–1141.
DOI: 10.1109/ROBOT.1997.614290

Chilali et al. 1996
Chilali, Mahmoud; Gahinet, Pascal; Scherer, Carsten, 1996.
Multi-Objective Output-Feedback Control via LMI Optimization.
IFAC Proceedings Volumes, **29** (1), S. 1691–1696.
DOI: 10.1016/S1474-6670(17)57912-4

Costa et al. 2018
Costa, Carlos M.; Veiga, Germano; Sousa, Armando; Rocha, Luis; Oliveira, Eugenio; Cardoso, Henrique Lopes; Thomas, Ulrike, 2018.
Automatic generation of disassembly sequences and exploded views from solidworks symbolic geometric relationships.
In: *2018 IEEE Int. Conf. on Autonomous Robot Systems and Competitions (ICARSC)*,
S. 211–218.
DOI: 10.1109/ICARSC.2018.8374185

Csiszar et al. 2017
Csiszar, Akos; Hoppe, Martin; Khader, Shahbaz A.; Verl, Alexander, 2017.
Behavior Trees for Task-Level Programming of Industrial Robots.
In: Schüppstuhl, Thorsten; Franke, Jörg; Tracht, Kirsten (Hrsg.): *Tagungsband des 2. Kongresses Montage Handhabung Industrieroboter*,
S. 169–180.
DOI: 10.1007/978-3-662-54441-9_18

Decré et al. 2013
Decré, Wilm; Smits, Ruben; Bruyninckx, Herman; Schutter, Joris De, 2013.
Extending iTaSC to support inequality constraints and non-instantaneous task specification.
In: *2013 IEEE Int. Conf. on Robotics and Automation (ICRA)*,
S. 964–971.
DOI: 10.1109/ROBOT.2009.5152477

Diamond et al. 2016
Diamond, Steven; Boyd, Stephen P., 2016.
CVXPY: A Python-Embedded Modeling Language for Convex Optimization.
Journal of Machine Learning Research, **17** (83), S. 1–5

DIN 8593-1 2003
DIN 8593-1, 2003.
DIN 8593-1:2003-09; Fertigungsverfahren Fügen - Teil 1: Zusammensetzen; Einordnung, Unterteilung, Begriffe

DIN 8593-3 2003
DIN 8593-3, 2003.
DIN 8593-3:2003-09; Fertigungsverfahren Fügen - Teil 3: Anpressen, Einpressen; Einordnung, Unterteilung, Begriffe

Doerr et al. 2017
Doerr, Andreas; Daniel, Christian; Nguyen-Tuong, Duy; Marco, Alonso; Schaal, Stefan; Marc, Toussaint; Trimpe, Sebastian, 2017a.
Optimizing Long-term Predictions for Model-based Policy Search.
In: *2017 Proceedings of Machine Learning Research*,
S. 227–238

Doerr et al. 2017
Doerr, Andreas; Nguyen-Tuong, Duy; Marco, Alonso; Schaal, Stefan; Trimpe, Sebastian, 2017b.
Model-based policy search for automatic tuning of multivariate PID controllers.
In: *2017 IEEE Int. Conf. on Robotics and Automation (ICRA)*,
S. 5295–5301.
DOI: 10.1109/ICRA.2017.7989622

Dominick et al. 2014
Dominick, Vanthienen; Klotzbücher, Markus; Bruyninckx, Herman, 2014.
The 5C-based architectural Composition Pattern: Lessons learned from re-developing the iTaSC framework for constraint-based robot programming,
S. 17–35.
DOI: 10.6092/JOSER_2014_05_01_p17

Doty et al. 1993
Doty, Keith. L.; Melchiorri, Claudio; Bonivento, Claudio, 1993.
A Theory of Generalized Inverses Applied to Robotics.
The International Journal of Robotics Research, **12** (1), S. 1–19.
DOI: 10.1177/027836499301200101

European Commission 2013
European Commission, 2013.
Robotics 2020; Strategic Research Agenda for Robotics in Europe,
Zugriff am: 25. 06. 2019.
Verfügbar unter: https://ec.europa.eu/research/industrial_technologies/pdf/robotics-ppp-roadmap_en.pdf

Finkemeyer 2004
Finkemeyer, Bernd, 2004.
Robotersteuerungsarchitektur auf der Basis von Aktionsprimitiven.
ISBN 978-3-8322-2893-4.
Braunschweig: Shaker; Aachen, RWTH, Diss., 2004

Finkemeyer et al. 2005
Finkemeyer, Bernd; Kröger, Torsten; Wahl, Friedrich M., 2005.
Executing assembly tasks specified by manipulation primitive nets.
Advanced Robotics, **19** (5), S. 591–611.
DOI: 10.1163/156855305323383811

Föllinger 1998
Föllinger, Otto, 1998.
Nichtlineare Regelungen I: Grundbegriffe, Anwendungen der Zustandsebene, Direkte Methode.
8. Aufl.
Oldenbourg: De Gruyter.
Methoden der Regelungs- und Automatisierungstechnik.
DOI: 10.1524/9783110406153

Franka Emika 2019
Franka Emika, 2019a.
Franka Emika Panda,
Zugriff am: 25. 06. 2019.
Verfügbar unter: https://www.franka.de

Franka Emika 2019
Franka Emika, 2019b.
Panda Technical Data,
Zugriff am: 25. 06. 2019.
Verfügbar unter: https://s3-eu-central-1.amazonaws.com/franka-de-uploads-staging/uploads/2018/05/2018-05-datasheet-panda.pdf

Fridman et al. 2009
Fridman, Emilia; Dambrine, Michel, 2009.
Control under quantization, saturation and delay: An LMI approach.
Automatica, **45** (10), S. 2258–2264.
DOI: 10.1016/j.automatica.2009.05.020

Grant et al. 2014
Grant, Michael; Boyd, Stephen P., 2014.
CVX: Matlab Software for Disciplined Convex Programming, version 2.1

Grant et al. 2006
Grant, Michael; Boyd, Stephen P.; Ye, Yinyu, 2006.
Disciplined Convex Programming.
In: Liberti, Leo; Maculan, Nelson (Hrsg.): *Global Optimization*,
S. 155–210.
DOI: 10.1007/0-387-30528-9_7

Halt et al. 2018
Halt, Lorenz; Nägele, Frank; Tenbrock, Philipp; Pott, Andreas, 2018a.
Intuitive constraint-based robot programming for robotic assembly tasks.
In: *2018 IEEE Int. Conf. on Robotics and Automation (ICRA)*,
S. 520–526.
DOI: 10.1109/ICRA.2018.8462882

Halt et al. 2019
Halt, Lorenz; Pan, Fengjunjie; Tenbrock, Philipp; Pott, Andreas; Seel, Thomas, 2019.
A Transferable Force Controller based on Prescribed Performance for Contact Establishment in Robotic Assembly Tasks.
In: *2019 IEEE Int. Conf. on Automation Science and Engineering (CASE)*,
S. 830–835.
DOI: 10.1109/COASE.2019.8843020

Halt et al. 2018
Halt, Lorenz; Tenbrock, Philipp; Nägele, Frank; Pott, Andreas, 2018b.
On the implementation of transferable assembly applications for industrial robots.
In: *2018 IEEE . Symposium on Robotics (ISR)*,
S. 1–7.
ISBN 978-3-8007-4699-6

Hasegawa et al. 1992
Hasegawa, T.; Suehiro, T.; Takase, K., 1992.
A model-based manipulation system with skill-based execution.
IEEE Transactions on Robotics and Automation, **8** (5), S. 535–544.
DOI: 10.1109/70.163779

Herrero et al. 2017
Herrero, Héctor; Moughlbay, Amine Abou; Outón, Jose Luis; Sallé, Damien; de Ipiña, Karmele López, 2017.
Skill based robot programming: Assembly, vision and Workspace Monitoring skill interaction.
Neurocomputing, **255**, S. 61–70.
DOI: 10.1016/j.neucom.2016.09.133

Hoffman et al. 2018
Hoffman, Enrico Mingo; Laurenzi, Arturo; Muratore, Luca; Tsagarakis, Nikos G.; Caldwell, Darwin G., 2018.
Multi-Priority Cartesian Impedance Control based on Quadratic Programming Optimization.
In: *2018 IEEE Int. Conf. on Robotics and Automation (ICRA)*,
S. 309–315.
DOI: 10.1109/ICRA.2018.8462877

Hogan 1984
Hogan, Neville, 1984.
Impedance Control: An Approach to Manipulation.
In: *1984 American Control Conference (ACC)*,
S. 304–313.
DOI: 10.23919/ACC.1984.4788393

IFR 2018
IFR, 2018.
Executive Summary World Robotics 2018 Industrial Robots.
International Federation of Robotics (Frankfurt am Main),
Zugriff am: 14. 03. 2022.
Verfügbar unter: https://ifr.org/downloads/press2018/Executive_Summary_WR_2018_Industrial_Robots.pdf

ISO/IEC 25010 2011
ISO/IEC 25010, 2011.
ISO/IEC 25010:2011; Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models,
Zugriff am: 14. 03. 2022.
Verfügbar unter: https://www.iso.org/standard/35733.html

Jäkel et al. 2010
Jäkel, Rainer; Schmidt-Rohr, Sven R.; Lösch, Martin; Dillmann, Rüdiger, 2010.
Representation and constrained planning of manipulation strategies in the context of Programming by Demonstration.
In: *2010 IEEE Int. Conf. on Robotics and Automation (ICRA)*,
S. 162–169.
DOI: 10.1109/ROBOT.2010.5509959

Jasniewicz 2010
Jasniewicz, Boris, 2010.
Über weiche strukturvariable Regelung mittels impliziter Lyapunov-Funktionen: Von der impliziten zur expliziten Regelung.
ISBN 978-3-18-516908-3.
Düsseldorf: VDI-Verlag; Darmstadt, TU, Diss., 2009.
urn:nbn:de:tuda-tuprints-20915

Jiménez 2013
Jiménez, P., 2013.
Survey on assembly sequencing: A combinatorial and geometrical perspective.
Journal of Intelligent Manufacturing, **24** (2), S. 235–250.
DOI: 10.1007/s10845-011-0578-5

Kanakis et al. 2018
Kanakis, George S.; Dimeas, Fotios; Rovithakis, George A.; Doulgeri, Zoe, 2018.
Prescribed contact establishment of a robot with a planar surface under position and stiffness uncertainties.
Robotics and Autonomous Systems, **2018** (104), S. 99–108.
DOI: 10.1016/j.robot.2018.03.005

Kanakis et al. 2017
Kanakis, George S.; Rovithakis, George A.; Doulgeri, Zoe, 2017.
On Prescribed Contact Establishment.
IFAC-PapersOnLine, **50** (1), S. 4831–4836.
DOI: 10.1016/j.ifacol.2017.08.970

Khalil 2002
Khalil, Hassan K., 2002.
Nonlinear systems.
3. ed.
Prentice Hall.
ISBN 0-13-122740-8

Klotzbücher et al. 2012
Klotzbücher, Markus; Bruyninckx, Herman, 2012.
Coordinating Robotic Tasks and Systems with rFSM Statecharts.
DOI: 10.6092/JOSER_2012_03_01_p28

Klotzbücher et al. 2011
Klotzbücher, Markus; Smits, Ruben; Bruyninckx, Herman; Schutter, Joris De, 2011.
Reusable hybrid force-velocity controlled motion specifications with executable Domain Specific Languages.
In: *2011 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*,
S. 4684–4689.
DOI: 10.1109/IROS.2011.6094782

Kresse et al. 2012
Kresse, Ingo; Beetz, Michael, 2012.
Movement-aware action control — Integrating symbolic and control-theoretic action execution.
In: *2012 IEEE Int. Conf. on Robotics and Automation (ICRA)*,
S. 3245–3251.
DOI: 10.1109/ICRA.2012.6225119

Kröger et al. 2011
Kröger, Torsten; Finkemeyer, Bernd, 2011.
Robot motion control during abrupt switchings between manipulation primitives.
In: *2011 ICRA workshop on mobile manipulation*
Zugriff am: 14. 03. 2022.
Verfügbar unter: http://mobilemanipulation.org/icra2011/images/8/80/ICRA_2011_Kroger.pdf

Kröger et al. 2010
Kröger, Torsten; Finkemeyer, Bernd; Wahl, Friedrich M., 2010.
Manipulation Primitives - A Universal Interface between Sensor-Based Motion Control and Robot Programming.
In: Schütz, Daniel; Wahl, Friedrich M. (Hrsg.): *Robotic systems for handling and assembly*,
S. 293–313.
Springer Tracts in Advanced Robotics.
DOI: 10.1007/978-3-642-16785-0_17

Labbe 2014
Labbe, Roger, 2014.
Kalman and Bayesian Filters in Python,
Zugriff am: 25. 01. 2019.
Verfügbar unter: https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python

Laet et al. 2012
Laet, Tinne De; Smits, Ruben; Bruyninckx, Herman; Schutter, Joris De, 2012.
Constraint-Based Task Specification and Control for Visual Servoing Application Scenarios.
at - Automatisierungstechnik, **60** (5), S. 260–269.
DOI: 10.1524/auto.2012.0996

Lambrecht et al. 2011
Lambrecht, Jens; Chemnitz, Moritz; Krüger, Jörg, 2011.
Control layer for multi-vendor industrial robot interaction providing integration of supervisory process control and multifunctional control units.
In: *2011 IEEE Conf. on Technologies for Practical Robot Applications (TePRA)*,
S. 115–120.
DOI: 10.1109/TEPRA.2011.5753492

Lefebvre et al. 2012
Lefebvre, Tine; Xiao, Jing; Bruyninckx, Herman; de Gersem, Gudrun, 2012.
Active compliant motion: A survey.
Advanced Robotics, **19** (5), S. 479–499.
DOI: 10.1163/156855305323383767

Lens 2009
Lens, Hendrik, 2009.
Fast robust stabilization by saturating output feedback of uncertain linear systems with input constraints.
In: *2009 IEEE Conf. on Decision and Control (CDC) and Chinese Control Conference (CCC)*,
S. 6935–6940.
ISBN 978-1-4244-3871-6.
DOI: 10.1109/CDC.2009.5400601

Lens 2010
Lens, Hendrik, 2010.
Schnelle Regelung mit Ausgangsrückführung für Systeme mit Stellgrößenbeschränkung.
ISBN 978-3-18-516708-9.
Düsseldorf: VDI-Verlag; Darmstadt, TU, Diss., 2009.
urn:nbn:de:tuda-tuprints-22774

Lens et al. 2008
Lens, Hendrik; Adamy, Jürgen, 2008.
Observer Based Controller Design for Linear Systems with Input Constraints.
IFAC Proceedings Volumes, **41** (2), S. 9916–9921.
DOI: 10.3182/20080706-5-KR-1001.01678

Löfberg 2004
Löfberg, Johan, 2004.
YALMIP: A Toolbox for Modeling and Optimization in MATLAB.
In: *2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No.04CH37508)*,
S. 284–289.
DOI: 10.1109/CACSD.2004.1393890

Maderna et al. 2018
Maderna, Riccardo; Casalino, Andrea; Zanchettin, Andrea Maria; Rocco, Paolo, 2018.
Robotic handling of liquids with spilling avoidance: a constraint-based control approach.
In: *2018 IEEE Int. Conf. on Robotics and Automation (ICRA)*.
DOI: 10.1109/ICRA.2018.8460927

Maeda et al. 2008
Maeda, Yusuke; Ushioda, Tatsuya; Makita, Satoshi, 2008.
Easy robot programming for industrial manipulators by manual volume sweeping.
In: *2008 IEEE Int. Conf. on Robotics and Automation (ICRA)*,
S. 2234–2239.
DOI: 10.1109/ROBOT.2008.4543546

Makris et al. 2014
Makris, Sotiris; Tsarouchi, Panagiota; Surdilovic, Dragoljub; Krüger, Jörg, 2014.
Intuitive dual arm robot programming for assembly operations.
CIRP Annals - Manufacturing Technology, **63** (1), S. 13–16.
DOI: 10.1016/j.cirp.2014.03.017

Malhan et al. 2018
Malhan, Rishi K.; Shahapurkar, Yash; Kabir, Ariyan M.; Shah, Brual; Gupta, Satyandra K., 2018.
Integrating Impedance Control and Learning Based Search Scheme for Robotic Assemblies Under Uncertainty.
In: *2018 ASME Int. Conf. on Manufacturing Science and Engineering*,
V003T02A006.
DOI: 10.1115/MSEC2018-6626

Mansard et al. 2009
Mansard, Nicolas; Stasse, Olivier; Evrard, Paul; Kheddar, Abderrahmane, 2009.
A versatile Generalized Inverted Kinematics implementation for collaborative working humanoid robots: The Stack Of Tasks.
In: *2009 Int. Conf. on Advanced Robotics (ICAR)*,
S. 1–6

Marco et al. 2016
Marco, Alonso; Hennig, Philipp; Bohg, Jeannette; Schaal, Stefan; Trimpe, Sebastian, 2016.
Automatic LQR tuning based on Gaussian process global optimization.
In: *2016 IEEE Int. Conf. on Robotics and Automation (ICRA)*,
S. 270–277.
DOI: 10.1109/ICRA.2016.7487144

Mason 1981
Mason, Matthew T., 1981.
Compliance and Force Control for Computer Controlled Manipulators.
IEEE Transactions on Systems, Man, and Cybernetics, **11** (6), S. 418–432.
DOI: 10.1109/TSMC.1981.4308708

Meurer et al. 2017
Meurer, Aaron et al., 2017.
SymPy: Symbolic computing in Python.
PeerJ Computer Science, **3**, S. e103.
DOI: 10.7717/peerj-cs.103

Mittelmann 2012
Mittelmann, Hans D., 2012.
The State-of-the-Art in Conic Optimization Software.
In: Anjos, Miguel F. (Hrsg.): *Handbook on semidefinite, conic and polynomial optimization*,
S. 671–686.
International series in operations research & management science.
ISBN 978-1-4614-0769-0.
DOI: 10.1007/978-1-4614-0769-0_23

Mosemann et al. 2001
Mosemann, Heiko; Wahl, Friedrich M., 2001.
Automatic decomposition of planned assembly sequences into skill primitives.
IEEE Transactions on Robotics and Automation, **17** (5), S. 709–718.
DOI: 10.1109/70.964670

Müller et al. 2017
Müller, Nico; Wagner, Maximilian; Heß, Peter, 2017.
Intuitive Programmierung für die Mensch-Roboter-Kollaboration.
ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb, **112** (7-8), S. 477–480.
DOI: 10.3139/104.111762

Nägele et al. 2018
Nägele, Frank; Halt, Lorenz; Tenbrock, Philipp; Pott, Andreas, 2018.
A prototype-based skill model for specifying robotic assembly tasks.
In: *2018 IEEE Int. Conf. on Robotics and Automation (ICRA)*.
DOI: 10.1109/ICRA.2018.8462885

Nägele et al. 2019
Nägele, Frank; Halt, Lorenz; Tenbrock, Philipp; Pott, Andreas, 2019.
Composition and Incremental Refinement of Skill Models for Robotic Assembly Tasks.
In: *2019 IEEE Int. Conf. on Robotic Computing (IRC)*, S. 177–182.
DOI: 10.1109/IRC.2019.00034

Nakamura et al. 1987
Nakamura, Yoshihiko; Hanafusa, Hideo; Yoshikawa, Taizo, 1987.
Task-Priority Based Redundancy Control of Robot Manipulators.
The International Journal of Robotics Research, **6** (2), S. 3–15.
DOI: 10.1177/027836498700600201

Oliphant 2006
Oliphant, Travis, 2006.
NumPy: A guide to NumPy,
Zugriff am: 22. 10. 2019.
Verfügbar unter: http://www.numpy.org/

Pan 2019
Pan, Fengjunjie, 2019.
Entwicklung und experimentelle Erprobung eines modellfreien „prescribed performance controllers“ für schnellen Kontaktaufbau bei Fügeprozessen mit Industrierobotern.
Berlin, TU, Masterarbeit, 2019

Pedersen et al. 2016
Pedersen, Mikkel Rath; Nalpantidis, Lazaros; Andersen, Rasmus Skovgaard; Schou, Casper; Bøgh, Simon; Krüger, Volker; Madsen, Ole, 2016.
Robot skills for manufacturing: From concept to industrial deployment.
Robotics and Computer-Integrated Manufacturing, **37**, S. 282–291.
DOI: 10.1016/j.rcim.2015.04.002

Perzylo et al. 2016
Perzylo, Alexander; Somani, Nikhil; Profanter, Stefan; Kessler, Ingmar; Rickert, Markus; Knoll, Alois, 2016.
Intuitive instruction of industrial robots: Semantic process descriptions for small lot production.
In: *2016 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*,
S. 2293–2300.
DOI: 10.1109/IROS.2016.7759358

Perzylo et al. 2015
Perzylo, Alexander; Somani, Nikhil; Rickert, Markus; Knoll, Alois, 2015.
An ontology for CAD data and geometric constraints as a link between product models and semantic robot task descriptions.
In: *2015 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*,
S. 4197–4203.
DOI: 10.1109/IROS.2015.7353971

Peternel et al. 2015
Peternel, Luka; Petric, Tadej; Babic, Jan, 2015.
Human-in-the-loop approach for teaching robot assembly tasks using impedance control interface.
In: *2015 IEEE Int. Conf. on Robotics and Automation (ICRA)*,
S. 1497–1502.
DOI: 10.1109/ICRA.2015.7139387

Pott et al. 2019
Pott, Andreas; Dietz, Thomas, 2019.
Industrielle Robotersysteme.
iesbaden: Springer Vieweg.
DOI: 10.1007/978-3-658-25345-5

Quigley et al. 2009
Quigley, Morgan; Conley, Ken; Gerkey, Brian; Faust, Josh; Foote, Tully; Leibs, Jeremy; Wheeler, Rob; Ng, Andrew Y., 2009.
ROS: an open-source Robot Operating System.
In: *2009 ICRA workshop on open source software*,
S. 5

Raibert et al. 1981
Raibert, M. H.; Craig, John J., 1981.
Hybrid Position/Force Control of Manipulators.
Journal of Dynamic Systems, Measurement, and Control, **103** (2), S. 126.
DOI: 10.1115/1.3139652

Rovithakis 2018
Rovithakis, George A., 2018.
Prescribed Performance Adaptive Control of Uncertain Nonlinear Systems: State-of-the-art and Open Issues.
PAMM, **18** (1), S. e201800134.
DOI: 10.1002/pamm.201800134

Sagnol 2012
Sagnol, Guillaume, 2012.
Picos Documentation. Release 0.1.1.
Verfügbar unter: urn:nbn:de:0297-zib-17396

Samson et al. 1991
Samson, Claude; Le Borgne, Michel; Espiau, Bernard, 1991.
Robot control: The task function approach.
Robotica, (4), S. 447–448.
DOI: 10.1017/S0263574700000643

Schmitt et al. 2017
Schmitt, Philipp S.; Neubauer, Werner; Feiten, Wendelin; Wurm, Kai M.; Wichert, Georg V.; Burgard, Wolfram, 2017.
Optimal, sampling-based manipulation planning.
In: *2017 IEEE Int. Conf. on Robotics and Automation (ICRA)*,
S. 3426–3432.
DOI: 10.1109/ICRA.2017.7989390

Schutter et al. 2005
Schutter, Joris De; Rutgeerts, Johan; Aertbeliën, Erwin; de Groote, F.; Laet, Tinne De; Lefebvre, Tine; Verdonck, W.; Bruyninckx, Herman, 2005.
Unified Constraint-Based Task Specification for Complex Sensor-Based Robot Systems.
In: *2005 IEEE Int. Conf. on Robotics and Automation (ICRA)*,
S. 3607–3612.
DOI: 10.1109/ROBOT.2005.1570669

Siciliano et al. 1999
Siciliano, Bruno; Villani, Luigi, 1999.
Robot force control.
Kluwer Academic Publishers.
ISBN 0-7923-7733-8

Smits 2010
Smits, Ruben, 2010.
Robot Skills: Design of a Constraint-Based Methodology and Software Support.
Leuven, KU, Diss., 2010.
https://lirias.kuleuven.be/1745778?limo=0

Smits et al. 2011
Smits, Ruben; Bruyninckx, Herman, 2011.
Composition of complex robot applications via data flow integration.
In: *2011 IEEE Int. Conf. on Robotics and Automation (ICRA)*,
S. 5576–5580.
DOI: 10.1109/ICRA.2011.5979958

Smits et al. 2008
Smits, Ruben; Laet, Tinne De; Claes, Kasper; Bruyninckx, Herman; Schutter, Joris De, 2008.
iTASC: a tool for multi-sensor integration in robot manipulation.
In: *2008 IEEE Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems*,
S. 426–433.
DOI: 10.1109/MFI.2008.4648032

Somani et al. 2015
Somani, Nikhil; Gaschler, Andre; Rickert, Markus; Perzylo, Alexander; Knoll, Alois, 2015.
Constraint-based task programming with CAD semantics: From intuitive specification to real-time control.
In: *2015 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*,
S. 2854–2859.
DOI: 10.1109/IROS.2015.7353770

Somani et al. 2016
Somani, Nikhil; Rickert, Markus; Gaschler, Andre; Cai, Caixia; Perzylo, Alexander; Knoll, Alois, 2016.
Task level robot programming using prioritized non-linear inequality constraints.
In: *2016 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*,
S. 430–437.
DOI: 10.1109/IROS.2016.7759090

Stenmark et al. 2017
Stenmark, Maj; Haage, Mathias; Topp, Elin Anna, 2017.
Simplified Programming of Re-usable Skills on a Safe Industrial Robot.
In: *2017 ACMIEEE Int. Conf. on Human-Robot Interaction (HRI)*,
S. 463–472.
DOI: 10.1145/2909824.3020227

Stolt 2012
Stolt, Andreas, 2012.
Robotic Assembly and Contact Force Control.
Lund, TU, Diss., 2012.
https://portal.research.lu.se/files/4277299/3242104.pdf

Stolt et al. 2011
Stolt, Andreas; Linderoth, Magnus; Robertsson, Anders; Johansson, Rolf, 2011a.
Force controlled assembly of emergency stop button.
In: *2011 IEEE Int. Conf. on Robotics and Automation (ICRA)*,
S. 3751–3756.
DOI: 10.1109/ICRA.2011.5979745

Stolt et al. 2012
Stolt, Andreas; Linderoth, Magnus; Robertsson, Anders; Johansson, Rolf, 2012.
Force controlled robotic assembly without a force sensor.
In: *2012 IEEE Int. Conf. on Robotics and Automation (ICRA)*,
S. 1538–1543.
DOI: 10.1109/ICRA.2012.6224837

Stolt et al. 2011
Stolt, Andreas; Linderoth, Magnus; Robertsson, Anders; Jonsson, Marie; Murray, Thomas, 2011b.
Force controlled assembly of flexible aircraft structure.
In: *2011 IEEE Int. Conf. on Robotics and Automation (ICRA)*,
S. 6027–6032.
DOI: 10.1109/ICRA.2011.5979962

Tenorth et al. 2014
Tenorth, Moritz; Bartels, Georg; Beetz, Michael, 2014.
Knowledge-based specification of robot motions.
In: *2014 Europ. Conf. on Artificial Intelligence (ECAI)*.
DOI: 10.3233/978-1-61499-419-0-873

Thomas et al. 2013
Thomas, Ulrike; Hirzinger, Gerd; Rumpe, Bernhard; Schulze, Christoph; Wortmann, Andreas, 2013.
A new skill based robot programming language using UML/P Statecharts.
In: *2013 IEEE Int. Conf. on Robotics and Automation (ICRA)*,
S. 461–466.
DOI: 10.1109/ICRA.2013.6630615

Thomas et al. 2011
Thomas, Ulrike; Wahl, Friedrich M., 2011.
Assembly Planning and Task Planning — Two Prerequisites for Automated Robot Programming.
In: Schütz, Daniel; Wahl, Friedrich M. (Hrsg.): *Robotic Systems for Handling and Assembly*,
S. 333–354.
DOI: 10.1007/978-3-642-16785-0_19

Thrun et al. 2006
Thrun, Sebastian; Burgard, Wolfram; Fox, Dieter, 2006.
Probabilistic robotics.
MIT Press.
Intelligent robotics and autonomous agents series.
ISBN 9780262201629

Universal-Robots 2019
Universal-Robots, 2019.
UR5 Technical Data,
Zugriff am: 25. 06. 2019.
Verfügbar unter: https://www.universal-robots.com/media/1801303/eng_199901_ur5_tech_spec_web_a4.pdf

Vajta 2000
Vajta, M., 2000.
Some remarks on Padé-approximations.
In: *2000 Proceedings of the 3rd TEMPUS-INTCOM Symposium*,
S. 53–58

Vanthienen et al. 2013
Vanthienen, Dominick; Klotzbücher, Markus; Schutter, Joris De; Laet, Tinne De; Bruyninckx, Herman, 2013.
Rapid application development of constrained-based task modelling and execution using domain specific languages.
In: *2013 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*,
S. 1860–1866.
DOI: 10.1109/IROS.2013.6696602

VDI 2860 1990
VDI 2860, 1990.
VDI 2860 – Montage- und Handhabungstechnik, Handhabungsfunktionen, Handhabungseinrichtungen; Begriffe, Definitionen, Symbole

Wacoh Tech 2019
Wacoh Tech, 2019.
Dynpick WEF-6A200-4-RCD,
Zugriff am: 25. 06. 2019.
Verfügbar unter: https://wacoh-tech.com/en/products/dynpick/200n_rcd.html

Wang et al. 2015
Wang, Lihui; Schmidt, Bernard; Givehchi, Mohammad; Adamson, Göran, 2015.
Robotic assembly planning and control with enhanced adaptability through function blocks.
The International Journal of Advanced Manufacturing Technology, **77** (1), S. 705–715.
DOI: 10.1007/s00170-014-6468-1

Weiss Robotics 2019
Weiss Robotics, 2019.
KMS40,
Zugriff am: 25. 06. 2019.
Verfügbar unter: https://www.weiss-robotics.com/wp-content/uploads/kms40_leaflet_en.pdf

Whitney 1982
Whitney, D. E., 1982.
Quasi-Static Assembly of Compliantly Supported Rigid Parts.
Journal of Dynamic Systems, Measurement, and Control, **104** (1), S. 65.
ISSN 00220434.
DOI: 10.1115/1.3149634

Winkler 2015
Winkler, Alexander, 2015.
Sensorgeführte Bewegungen stationärer Roboter.
ISBN 978-3-944640-78-5.
Chemnitz: Universitätsverlag der Technischen Universität, Chemnitz, TU, Habilitationsschrift, 2014.
https://nbn-resolving.org/urn:nbn:de:bsz:ch1-qucosa-197679

Winkler et al. 2012
Winkler, Alexander; Suchy, Jozef, 2012.
Position feedback in force control of industrial manipulators - An experimental comparison with basic algorithms.
In: *2012 IEEE Int. Symposium on Robotic and Sensors Environments Proceedings*,
S. 31–36.
DOI: 10.1109/ROSE.2012.6402604

Winkler et al. 2015
Winkler, Alexander; Suchy, Jozef, 2015.
Implicit Force Control Of A Position Controlled Robot – A Comparison With Explicit Algorithms.
In: *Int. Journal of Computer, Electrical, Automation, Control and Information Engineering*.
DOI: 10.5281/ZENODO.1106937

Winkler et al. 2016
Winkler, Alexander; Suchy, Jozef, 2016.
Explicit and implicit force control of an industrial manipulator — An experimental summary.
In: *2016 Int. Conf. on Methods and Models in Automation and Robotics (MMAR)*,
S. 19–24.
DOI: 10.1109/MMAR.2016.7575081

Wirnshofer et al. 2018

Wirnshofer, Florian; Schmitt, Philipp S.; Feiten, Wendelin; v. Wichert, Georg; Burgard, Wolfram, 2018.
Robust, Compliant Assembly via Optimal Belief Space Planning.
In: *2018 IEEE Int. Conf. on Robotics and Automation (ICRA)*.
DOI: 10.1109/ICRA.2018.8460995