Selbstoptimierung verteilter mechatronischer Systeme auf Basis paretooptimaler Systemkonfigurationen

zur Erlangung des akademischen Grades eines
DOKTORS DER INGENIEURWISSENSCHAFTEN (Dr.-Ing.)
der Fakultät für Maschinenbau
der Universität Paderborn

genehmigte
DISSERTATION

von
Dipl.-Ing. Eckehard Münch
aus Paderborn

Tag des Kolloquiums:	3. Dezember 2012
Referent:	Prof. Dr.-Ing. habil. Ansgar Trächtler
Korreferent:	Prof. Dr.-Ing. Joachim Böcker

Vorwort

Die vorliegende Dissertation entstand während meiner Zeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Regelungstechnik und Mechatronik (RtM) an der Universität Paderborn. Sie resultiert aus meinen Forschungsarbeiten im Rahmen des DFG-Sonderforschungsbereiches (SFB) 614 „Selbstoptimierende Systeme des Maschinenbaus“.

Das Verfassen einer Dissertation ist zwar eine sehr individuelle Angelegenheit, es gibt aber immer auch zahlreiche Personen, die durch Förderung, Unterstützung und Hilfe ganz unterschiedlicher Natur einen wichtigen Teil zum Gelingen einer solchen Arbeit beitragen. Bei all diesen Personen möchte ich mich an dieser Stelle herzlich bedanken.

Als erstes danke ich meinem Doktorvater, Herrn Prof. Dr.-Ing. habil. Ansgar Trächtler für die Betreuung, die fachliche Unterstützung meiner Arbeit und die fruchtbaren Diskussionen. Herrn Prof. Dr.-Ing. Joachim Böcker danke ich für die Durchsicht der Arbeit und die Übernahme des Korreferats. Mein Dank gilt auch dem verstorbenen, früheren Leiter des Lehrstuhls Herrn Prof. Dr.-Ing. Joachim Lückel. Bereits während meines Diplom-Studiums hat er meine Begeisterung für das Gebiet der mathematischen Optimierung geweckt. Diese Methoden haben letztendlich eine zentrale Rolle in meinen Arbeiten eingenommen.

Allen Mitarbeitern und Kollegen des Lehrstuhls und des SFB 614 danken ich, für die offene und kooperative Arbeitsatmosphäre und die stetige Bereitschaft zu Diskussionen. Namentlich nenne ich an dieser Stelle Herrn Dipl.-Math. Henner Vöcking und Herrn Dipl.-Math. Martin Krüger, mit denen ich gemeinsam im Sonderforschungsbereich tätig war. Bei ihnen möchte ich mich ganz besonders für die gute Zusammenarbeit, die zahlreichen Diskussionen und wertvollen Anregungen bedanken.

Und schlussendlich gilt mein ganz besonderer Dank meiner Familie, meinen Eltern, meinen Schwiegereltern und vor allem meiner Frau Karoline, die mich in den alltäglichen Dingen des Lebens entlastet und mir so den Freiraum und die notwendige Zeit zum Verfassen dieser Arbeit geschaffen haben.

Friedrichshafen, im Januar 2013 *Eckehard Münch*

Inhaltsverzeichnis

Abkürzungen

AD	Algorithmische Differentiation oder Automatische Differentiation
AKTAKON	Aktive Aufbau-Kontrolle
AMS	Autonomes Mechatronisches System
BFGS	Broyden-Fletcher-Goldfarb-Shanno
CAMeL	Computer Aided Mechatronic Laboratory
DFP	Davidon-Fletcher-Powell
DLR	Deutsches Zentrum für Luft- und Raumfahrt
DQ	Differenzenquotient
EKF	Erweiterter Kalmanfilter
FPGA	Field Programmable Gate Array
GAIO	Global Analysis of Invariant Objects
GSL	GNU Scientific Library
HG	Hilfsgruppe
HM	Hilfsmodul
IAE	Integral of the Absolute Error
ISE	Integral of the Squared Error
ITAE	Integral of time multiplied by the Absolute Error
ITSE	Integral of time multiplied by the Squared Error
KKT	Karush-Kuhn-Tucker
LZI	Linear Zeitinvariant
MEM	Memory – Speicherplatzbedarf
MFG	Mechatronische Funktionsgruppe
MFM	Mechatronisches Funktionsmodul
MKS	Mehrkörpersystem
MLaP	Mechatronik Laboratorium Paderborn
MOBU	Multi-Objective-Bottom-Up
MOP	Mehrzieloptimierungproblem
MOPO	Multi-Objective Parameter Optimization
MOPS	Multi-Objective Parameter Synthesis
MOPSO	Multi-Objective Particle Swarm Optimization
NAG	The Numerical Algorithms Group
NBI	Normal Boundary Intersection
NBP	Neue Bahntechnik Paderborn
NEOS	Network Enabled Optimization System
NLOP	Nichtlineare Operationen
OCM	Operator-Controller-Modul
PKW	Personenkraftwagen
RtM	Lehrstuhl für Regelungstechnik und Mechatronik
SFB	Sonderforschungsbereich
SQP	Sequentielle Quadratische Programmierung
VDI	Verein Deutscher Ingenieure
VMS	Vernetzte Mechatronische Systeme

Formelzeichen

Verwendung von Maßeinheiten

In der Regel sind die physikalischen Größen und Parameter mit ihren jeweiligen Einheiten angegeben. In Übertragungsfunktionen und Zustandsgleichungen bzw. -matrizen wie z. B. in den Gleichungen (8.26) und (9.5) sowie im Anhang A.3 wurde der besseren Lesbarkeit wegen aber darauf verzichtet. Dort sind die Ein- und Ausgangsgrößen, die Zustandsvariablen und die Zeit auf SI-Einheiten normiert.

Darstellung von Skalaren, Vektoren und Matrizen

x	Skalar	$\underline{x}$	Vektor	$\underline{\mathbf{X}}$	Matrix

Formelzeichen ausgewählter Größen

A	Fläche
$\underline{\mathbf{A}}, \underline{\mathbf{B}}, \underline{\mathbf{C}}, \underline{\mathbf{D}}$	Zustandsraummatrizen eines linearen zeitinvarianten Systems
$\underline{\mathbf{B}}$	Quasi-Newton-Matrix
$\underline{\mathbf{C}}$	Steifigkeitsmatrix
D_{kij}	Dominanzmaß für den Übertragungspfad kij
$\underline{\mathbf{D}}$	Dämpfungsmatrix
$\underline{\mathbf{E}}$	Störeingangsmatrix
F	Kraft
$G(s)$	Übertragungsfunktion
$\underline{\mathbf{H}}$	Hesse-Matrix
$\underline{\mathbf{I}}$	Einheitsmatrix
J	Trägheitsmoment
K	Verstärkungsfaktor
$\underline{\mathbf{K}}$	Beobachtermatrix
M	Moment
N	Anzahl
$\underline{N}$	Normalenvektor
P	Leistung
$\underline{P}$	Zielpunkt
$\underline{\mathbf{P}}$	Steuerbarkeitsmatrix
Q	Volumenstrom
$\underline{\mathbf{Q}}$	Beobachtbarkeitsmatrix
$\underline{\mathbf{R}}$	Reglermatrix
$\underline{\mathbf{S}}$	Steuermatrix

Formelzeichen ausgewählter Größen (fortgesetzt)

T	Zeitkonstante, Zeitintervall
$\underline{\mathbf{V}}$	Transformationsmatrix
$\mathcal{I}$	Indexmenge
$\mathcal{J}_{\underline{f}}$	Jacobi-Matrix der Vektorfunktion $\underline{f}$
$\mathcal{L}(\underline{x}, \underline{\lambda})$	Lagrange-Funktion
$\mathcal{M}$	Modellbeschreibung
$\mathcal{U}^i$	Indexmenge der zum Modul i direkt unterlagerten Module
$\mathcal{V}^i$	Indexmenge aller zum Modul i unterlagerten Module (einschließlich i)
c	Federsteifigkeit
d	Dämpfungskonstante
$\underline{d}_k$	Abstiegsrichtung in einem Schritt k
$\underline{e}$	Regelabweichung
f	Zielgröße, Ausgangsgröße
$\dot{f}$	Gradient der Ausgangsgröße
$\bar{f}$	totales Differential der Ausgangsgröße
$\underline{f}(\underline{x})$	vektorielle Zielfunktion
$\underline{g}(\underline{x})$	Nebenbedingungen
h	Intervalllänge, Schrittweite
l	Länge
m	Masse, Anzahl der Zielfunktionen, Anzahl der Ausgangsgrößen
n	Anzahl der Parameter, Anzahl der unabhängigen Variablen
p	Druck, Exponent
$\underline{p}$	Vektor der Reglerparameter
$\underline{q}$	verallgemeinerte Koordinaten
$\dot{\underline{q}}$	zeitliche Ableitung der verallgemeinerten Koordinaten
t	Zeitpunkt
s	komplexer Parameter der Laplace-Transformation
$\underline{u}$	Stellgrößen, Eingangsgrößen
v	Geschwindigkeit
v_i	Zwischengröße
$\dot{v}_i$	Gradient der Zwischengröße v_i
$\bar{v}_i$	totales Differential der Ausgangsgröße nach der Zwischengröße v_i
$\underline{v}^i$	Ausgangsvektor des Basismodells i
$\underline{w}$	Sollgrößen, Gewichtungsvektor
x_i	unabhängige Variable, Zustand
$\underline{x}$	Vektor der Optimierungsparameter, Zustandsvektor
$\dot{\underline{x}}$	zeitliche Ableitung des Zustandsvektors
$\underline{y}$	Ausgangsgrößen
$\underline{z}$	Störgrößen
$\underline{\Phi}$	reale Zielerfüllungen
$\underline{\mathbf{\Lambda}}$	modale Systemmatrix
$\underline{\mathbf{\Sigma}}$	Singulärwertmatrix
$\underline{\mathbf{\Theta}}$	Simplex
Ξ	Quotient zweier Zielgrößen

Formelzeichen ausgewählter Größen (fortgesetzt)

$\underline{\Psi}$	Zielgrößenrelation
α_i	Gewichtungsfaktor
$\underline{\alpha}^i$	Paretopunktreferenz im Modul i
β	Kompressionsmodul
$\underline{\beta}$	Gewichtungsvektor der Zielgrößen
ϵ_{rel}	relativer Fehler
ϵ_k	Epsilon-Umgebung in einem Schritt k
ε_i	obere Schranke
$\phi(\ldots)$	Merit-Funktion, integrales Gütekriterium
γ	künstliche Zielgröße
γ_{ist}	Skalierungsfaktor zwischen realen und optimierten Zielgrößen
λ_i	Lagrange-Multiplikator, Eigenwert
σ_i	Singulärwert
ϑ_i	Erfüllungsgrad einer Zielfunktion
ξ	Lehr'sche-Dämpfung
$\psi(\ldots)$	Systemfunktion einer nichtlinearen Differentialgleichung
$\xi(\ldots), \vartheta(\ldots), \zeta(\ldots)$	Ausgangsfunktionen einer nichtlinearen Differentialgleichung

1 Einleitung

Die Mechatronik ist heutzutage aus vielen technischen Produkten nicht mehr wegzudenken. Diese Produkte begegnen uns in fast allen Bereichen des täglichen Lebens: sie finden sich in Kraftfahrzeugen, Flugzeugen und in der Medizintechnik ebenso wie in Windkraftanlagen, in Festplatten, CD/DVD-Spielern und sogar in Spielzeugen. Mechatronische Systeme beruhen auf dem engen Zusammenwirken von mechanischen, elektrotechnischen, regelungstechnischen und informationstechnischen Komponenten.

Die VDI-Richtlinie 2206 „Entwicklungsmethodik für mechatronische Systeme“ definiert die Mechatronik folgendermaßen [VDI04]:

> Mechatronik ist das enge synergetische Zusammenwirken der Fachdisziplinen Maschinenbau, Elektrotechnik und Informationstechnik beim Entwurf und der Herstellung industrieller Erzeugnisse sowie bei der Prozessgestaltung.

Getrieben durch den rasanten Fortschritt in der Mikroelektronik kommt der Informationstechnik in technischen Produkten eine immer größere Bedeutung zu. Durch gestiegene Rechenleistungen, Speicherkapazitäten und fortgeschrittene Kommunikationstechniken wird die Entwicklung intelligenter mechatronischer Systeme ermöglicht, die in der Lage sind, autonom und flexibel auf sich ändernde Anforderungen und Betriebsbedingungen zu reagieren.

Die Anforderungen und Zielsetzungen, die an ein System gestellt werden, sind dabei recht unterschiedlicher Natur und ergeben sich im Wesentlichen aus den Aufgaben und Funktion eines Systems, dem Kontext, in dem das System zum Einsatz kommt, aber auch aus den für die Aufgabe benötigten Ressourcen.

Bei mechatronischen Systemen, die geprägt sind durch die kontrollierte Bewegung von Massen, ist die Funktionalität eng mit dem Bewegungsverhalten des Systems verbunden. Aufgrund des starken Bezuges zur Regelungstechnik werden Zielsetzungen an das Bewegungsverhalten meist über die Güte der Steuerungen/Regelungen ausgedrückt. Anforderungen an den Ressourcenverbrauch beziehen sich meist auf den Energiebedarf. Zusätzlich ergeben sich aus dem Umfeld und den konkreten Aufgaben des Systems spezielle Anforderungen, wie beispielsweise an die Lärmentwicklung einer Maschine oder an die Schadstoffemissionen von Kraftfahrzeugen.

Diese Anforderungen sind im Allgemeinen gegensätzlicher Natur und widersprechen sich in einem gewissen Maße. Bereits die Zielsetzungen an die Funktionalität sind gegensätzlicher Natur. So stehen bspw. die Forderungen, eine Bewegung sowohl schnell, als auch gut gedämpft und präzise auszuführen, im Konflikt zueinander. Besonders offensichtlich wird dies bei den Anforderungen, die auf den *Nutzen*, also auf Funktionalität eines Systems, abzielen und den Zielsetzungen, die den dafür notwendigen *Aufwand* betreffen. *Nutzen* und *Aufwand* stehen in einem fundamentalen Konflikt zueinander. So ist die Effizienz von Systemen durch das Verhältnis von *Nutzen* zu *Aufwand* definiert:

$$Effizienz = \frac{Nutzen}{Aufwand}$$

Ein technisches System stellt immer eine Kompromisslösung zwischen den unterschiedlichen Zielsetzungen dar. Bei einem optimal ausgelegten System erfordert die gleichzeitige Verbesserung aller Ziele höherwertigere Komponenten, wie bessere Sensorik und Aktorik, oft aber auch einen erhöhten Ressourcenbedarf.

Sind technische Systeme im Betrieb unterschiedlichen Betriebsbedingungen ausgesetzt, oder unterliegen sie sogar wechselnden Zielen, so müssen diese im Entwurfsprozess berücksichtigt werden. Konventionelle Systeme werden auf einen festen Kompromiss hin ausgelegt. Dieser fällt meist konservativ aus, um eine hohe Robustheit gegenüber solchen Veränderungen zu erreichen. Dies bedeutet jedoch, dass sich das System nur selten optimal verhält. Besonders Situationen, die von ihren Eigenschaften her etwas abseits der Norm liegen, können oft nur suboptimal bewältigt werden.

Dieses Potential kann durch intelligente technische Systeme erschlossen werden, die in der Lage sind die aktuellen Betriebsbedingungen und Anforderungen zu erfassen und sich autonom auf veränderte Situationen einzustellen. Solche intelligenten Systeme werden in dieser Arbeit als **selbstoptimierende Systeme** bezeichnet.

Diese Sichtweise auf selbstoptimierende technische Systeme gründet sich auf den Sonderforschungsbereich 614 (SFB 614) „Selbstoptimierende Systeme des Maschinenbaus“[1] an der Universität Paderborn. Der SFB 614 befasst sich mit der Erforschung von Methoden, Verfahren und Konzepten für den Entwurf solcher intelligenter Systeme. Der Begriff der Selbstoptimierung von technischen Systemen wird hier folgendermaßen definiert:

> Unter **Selbstoptimierung** eines technischen Systems wird die endogene Anpassung der Ziele des Systems auf veränderte Einflüsse und die daraus resultierende zielkonforme autonome Anpassung der Parameter und ggf. der Struktur und somit des Verhaltens dieses Systems verstanden. Damit geht Selbstoptimierung über die bekannten Regel- und Adaptionsstrategien wesentlich hinaus; Selbstoptimierung ermöglicht handlungsfähige Systeme mit inhärenter „Intelligenz“, die in der Lage sind, selbstständig und flexibel auf veränderte Betriebsbedingungen zu reagieren. [SFB04]

Das besondere Merkmal selbstoptimierender Systeme ist die Fähigkeit, selbstständig die eigenen Ziele anzupassen. Dies unterscheidet selbstoptimierende Systeme von klassischen Systemen, die auf bereits zum Entwurfszeitpunkt festgelegte, fixe Ziele hin optimiert sind. Die Ziele werden auf die aktuelle Betriebssituation abgestimmt und die Systemkonfiguration bezüglich dieser Ziele optimiert. Dies resultiert letztendlich in einer Anpassung des Systemverhaltens.

1.1 Neue Bahntechnik Paderborn

Zentraler Demonstrator des SFB 614 ist das RailCab der Forschungsinitiative Neue Bahntechnik Paderborn (NBP) [RC10].

Die NBP beschreibt ein neuartiges Verkehrskonzept, dessen Idee darin besteht, den Individualverkehr vom PKW auf die Schiene zu bringen. Die Trennung zwischen Nah- und Fernverkehr sowie Güter- und Personentransport wird aufgelöst. Dies wird ermöglicht

[1] Gefördert durch die Deutsche Forschungsgemeinschaft (DFG) [SFB01, SFB04, SFB08, SFB10]

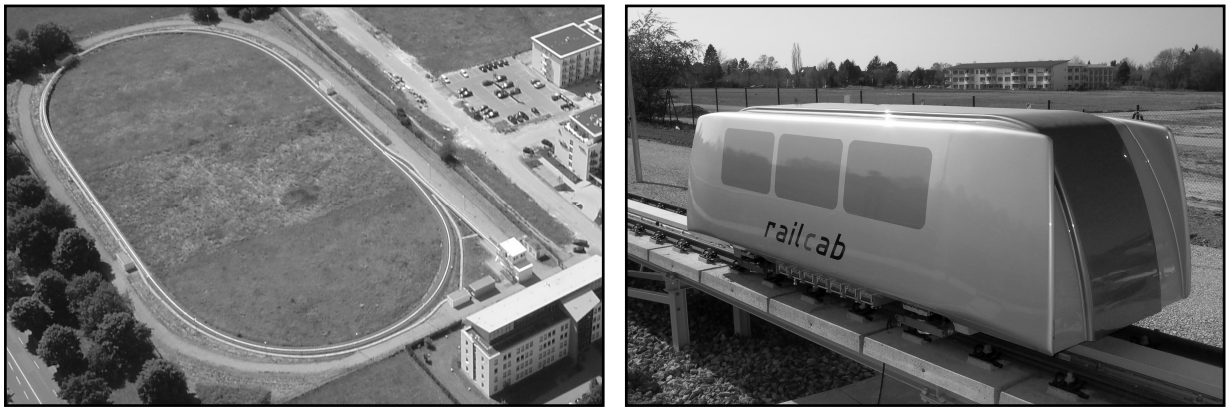


Abbildung 1.1: Teststrecke und Versuchsfahrzeug in Paderborn

durch viele kleine, autonome Fahrzeuge, den sogenannten RailCabs, die flexibel nach Bedarf Personen und Güter ohne Zwischenstopps zum Ziel transportieren.

Die Erprobung dieses komplexen mechatronischen Systems findet seit 2003 auf einer Teststrecke im Maßstab 1:2,5 an der Universität Paderborn statt (Abb. 1.1).

Das RailCab-System zeichnet sich durch eine hohe Modularität aus. Abbildung 1.2 zeigt die wichtigsten Module, deren Eigenschaften im Folgenden kurz aufgelistet werden. Eine detaillierte Beschreibung des RailCab-Systems und der intelligenten Funktionsmodule findet sich u. a. in [EHO01, Hen03, LH05, Trä06, LGH+09].

- Der Antrieb erfolgt über einen doppeltgespeisten Linearasynchronmotor. Mehrere RailCabs können sich bei diesem Konzept auf einem Statorabschnitt befinden. Die Energieübertragung geschieht berührungslos über den Antrieb, so dass Oberleitungen entfallen. Durch die hohe Positioniergenauigkeit lassen sich Konvois mit wenigen Zentimetern Abstand zwischen den Fahrzeugen aufbauen.

- Die Fahrzeuge besitzen Einzelachsfahrwerke mit Losrädern. Ein Sinuslauf, wie bei herkömmlichen Schienenfahrzeugen tritt hier nicht auf, die Zentrierung im Gleis geschieht über eine aktive Spurführung. Die Lage der Achse im Gleis wird über berührungslose Wirbelstromsensoren gemessen und die Zentrierung über eine Verdrehung der Achse erreicht. Die Weichen beim RailCab-System sind passiv ausgeführt. Die Richtungswahl wird nicht fahrwegseitig über die Stellung der Weichen bestimmt, sondern fahrzeugseitig über die aktive Spurführung gesteuert. Dies erlaubt es einzelnen Fahrzeugen bei voller Geschwindigkeit aus einem Konvoi auszuscheren.

- Die Federung des RailCabs ist als ein dämpferloses aktives Federungssystem aufgebaut. Die Dämpfung wird vollständig über geregelte hydraulische Zylinder erreicht. Ein Neigen des Aufbaus in Kurven ist über diese Zylinder ebenfalls möglich.

Das RailCab eignet sich hervorragend als Demonstrator für den SFB 614, da es eine große Bandbreite an möglichen Szenarien für den Einsatz von Selbstoptimierung aufweist. Diese reichen von den untersten Aktoren über das Fahrzeug und die Konvois bis hin zur Logistik. Ebenso stellt die NBP mit der Teststrecke, den Versuchsfahrzeugen und Prüfständen eine hervorragende Infrastruktur für die Erprobung zur Verfügung. So wird in dieser Arbeit auf den Prüfstand der NBP für ein neuartiges Unterflur-Federungssystem [Sch06] zurückgegriffen.

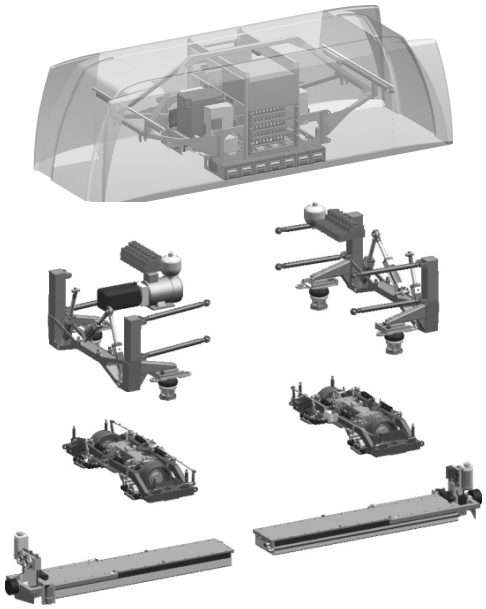


Abbildung 1.2: Modularer Aufbau des RailCabs

1.2 Zielsetzung und Motivation

Das Ziel dieser Arbeit ist die Realisierung einer selbstoptimierenden Regelung in einem hierarchisch verteilten mechatronischen System. Der besondere Schwerpunkt liegt dabei auf den unteren Strukturebenen eines mechatronischen Systems, bei denen starke gegenseitige physikalische Wechselwirkungen eine ganzheitliche Betrachtung der einzelnen Teilsysteme erfordern.

Mechatronische Systeme werden immer komplexer. Die Beherrschung dieser hohen Komplexität durch die Entwickler erfordert eine klare Strukturierung dieser Systeme. In [Lüc92, HSNL97, LHL01] und [Hes06] wird ein Strukturierungsansatz beschrieben, der sich an den Bewegungsfunktionen des mechatronischen Systems orientiert und in zahlreichen praktischen Arbeiten bewährt hat. Durch Modularisierung und Hierarchisierung wird das System anhand funktionaler Gesichtspunkte in eine modular-hierarchische Aggregatstruktur mit den folgenden Strukturelementen zerlegt:

- Mechatronische Funktionsmodule (MFM)
- Mechatronische Funktionsgruppen (MFG)
- Autonome Mechatronische Systeme (AMS)
- Vernetzte Mechatronische Systeme (VMS)

Die einzelnen Aggregate der Gruppen MFM, MFG, AMS und VMS realisieren intelligente Teilfunktionen und werden hierarchisch zu höherwertigen Funktionen des Gesamtsystems verknüpft.

Die Umsetzung dieser intelligenten Funktionen erfordert eine eigene informationsverarbeitende Einheit für jedes der Aggregate. Im Kontext selbstoptimierender mechatronischer Systeme ist das Konzept des Operator-Controller-Moduls (OCM) [OHG04a] gut geeignet, da hier neben den Funktionen zur Steuerung/Regelung, zur Ablaufsteuerung und

Überwachung des Systems auch Funktionen zum Zwecke der Selbstoptimierung in eine übersichtlich strukturierte Informationsverarbeitung mit klaren Schnittstellen eingebettet werden können. Aus der Verknüpfung der Teilfunktionen zu höherwertigen Funktionen folgt direkt eine hierarchische organisierte Kommunikation zwischen den OCM der einzelnen Aggregate.

Die Realisierung der Selbstoptimierung innerhalb dieser hierarchischen OCM-Architektur stellt besondere Anforderungen an die Methoden und Verfahren. In dieser Arbeit wird das Konzept einer über diese OCM-Architektur verteilten Wissensbasis vorgeschlagen, die auf jeder Ebene der Architektur Informationen über das System und sein Umfeld als Entscheidungsgrundlage für einen Selbstoptimierungsprozess bereitstellt. Es wird dabei besonderer Augenmerk auf die Kapselung der einzelnen Module und auf den Abstraktionsgrad der Informationen auf den verschiedenen Ebenen der Architektur gelegt.

Aufbauend auf der Wissensbasis wird in dieser Arbeit das Konzept einer selbstoptimierenden Regelung verfolgt, die auf Basis paretooptimaler Konfigurationen[2] arbeitet, die in der hierarchischen Wissensbasis hinterlegt sind. Im Fokus stehen hierbei regelungstechnische Systeme, deren Störverhalten für die Funktionalität des Systems entscheidend ist. Der Prozess der Selbstoptimierung wird über einen sog. Zielregelkreis realisiert, der analog zu einer klassischen Regelung aufgebaut ist. So können bestimmte Analyse- und Synthesemethoden für zeitdiskrete Regelungen auch auf diesen Zielregelkreis angewendet werden.

1.3 Aufbau der Arbeit

Insgesamt werden in dieser Arbeit drei thematische Stoßrichtungen verfolgt:

- Methoden und Verfahren der Mehrzieloptimierung
- Wissensbasis für selbstoptimierende hierarchische mechatronische Systeme
- Selbstoptimierende Regelungen auf Basis paretooptimaler Konfigurationen

Die Arbeit ist in die folgenden Kapitel untergliedert:

Kapitel 2 beschreibt die Grundlagen des modular-hierarchischen Strukturierungsansatzes für mechatronische Systeme. Neben der Aggregatstruktur und der Beschreibung der Strukturelemente wird auf die lokale und die hierarchisch verteilte Informationsverarbeitung selbstoptimierender mechatronischer Systeme eingegangen.

Optimierungsverfahren sind ein essenzieller Bestandteil selbstoptimierender mechatronischer Systeme. In den Kapiteln 3, 4 und 5 werden die notwendigen Grundlagen für die Anwendung mathematischer Optimierungsverfahren bei Auslegung mechatronischer Systeme vorgestellt.

Das **Kapitel 3** beschreibt die Grundlagen der mathematischen Optimierung und der Mehrzieloptimierung. Es werden einige ableitungsbehaftete Verfahren für kontinuierliche Optimierungsprobleme vorgestellt und verschiedene Wege aufgezeigt, wie diese zur Lösung von Problemen mit mehreren Zielfunktionen eingesetzt werden können.

[2] Eine paretooptimale Konfiguration, benannt nach dem italienischen Wirtschaftswissenschaftler Vilfred Pareto 1848-1923, ist das Ergebnis der Optimierung eines Systems nach mehreren Zielen gleichzeitig.

In **Kapitel 4** wird gezeigt, wie der Entwurf mechatronischer Systeme als ein Optimierungsproblem formuliert werden kann. Ausgehend von der mathematischen Präzisierung der Anforderungen und Ziele, die an das System gestellt werden, wird ein Optimierungsmodell aufgebaut, welches das Modell der Strecke und des Reglers um ein Anregungs- und Bewertungsmodell erweitert. In diesem Kapitel wird zudem eine Übersicht über häufig verwendete Bewertungsfunktionen gegeben.

In **Kapitel 5** wird auf die Sensitivitätsanalyse von Simulationsmodellen dynamischer Systeme eingegangen. Der Erfolg ableitungsbehafteter Optimierungsverfahren hängt wesentlich von der Genauigkeit der Sensitivitäten ab. Diese werden über die Gradienten der Zielfunktionen angegeben. Die Berechnung erfolgt meist mit Hilfe von *Differenzenquotienten*. Die *Algorithmische Differentiation* ist eine alternative Methode, welche die Berechnung der Gradienten von Funktionen, die in Form eines Computerprogramms vorliegen, ermöglicht. In dem Kapitel werden die grundlegenden Arbeitsweisen und die Implementierungstechniken mit ihren Vor- und Nachteilen vorgestellt. Des Weiteren wird der Einsatz bei der Simulation dynamischer Systeme diskutiert.

Kapitel 6 geht auf den Prozess der Selbstoptimierung und seine Anwendung auf Selbstoptimierende Regelungen ein. Es wird die Beziehung zu adaptiven Regelungen aufgezeigt und die Struktur einer lokalen Wissensbasis zur Unterstützung des Selbstoptimierungsprozesses vorgestellt. Diese Arbeiten bilden die Grundlage für den Aufbau der hierarchischen Wissensbasis.

Kapitel 7 beschreibt die hierarchische Wissensbasis für selbstoptimierende mechatronische Systeme. Die hierarchische Wissensbasis beruht auf zwei wichtigen Konzepten. Das „Rückgrat“ der Wissensbasis bildet ein hierarchisches über die OCM-Architektur verteiltes Modell, welches für jedes Aggregat das Verhalten in einem angepassten Detaillierungsgrad beschreibt. Auf der Basis dieses hierarchischen Modells arbeitet eine hierarchische Mehrzieloptimierung, die die paretooptimalen Systemkonfigurationen für den Selbstoptimierungsprozess zur Verfügung stellt.

Die Umsetzung des hierarchischen Modells und der hierarchischen Mehrzieloptimierung erfolgt in **Kapitel 8** am Beispiel des Unterflur-Federungsprüfstandes der NBP. Der Prüfstand wird hier nach den gegensätzlichen Zielen Komfort und Leistungsbedarf hin optimiert. Die geschieht unter Ausnutzung der modular-hierarchischen Struktur des Systems. Das Ergebnis sind die Paretomengen der Reglerparameter für jedes Modul des Prüfstandes.

In **Kapitel 9** wird das Konzept eines Ziel-Regelkreises zur Realisierung der Selbstoptimierung bei gestörten mechatronischen Systemen vorgestellt. Ein solcher Regelkreis kann unterschiedliche Regelungsabsichten verfolgen. Vorgestellt werden drei Varianten: die Zielregelung auf Zielrelationen und die Zielregelung auf absolute Zielvorgaben arbeiten analog zu einer klassischen Regelung nach einem festen Regelungsgesetz. Die Anwendung dieser Regelkreise erfolgt am Modell des Federungsprüfstands. Die gradientenbasierte Zielregelung basiert auf einem Gradientenverfahren. Die hierfür notwendigen Abstiegsrichtungen werden mit einem erweiterten Beobachter im Betrieb ermittelt.

2 Architektur

2.1 Aggregatstruktur

Mechatronische Systeme und insbesondere selbstoptimierende Systeme zeichnen sich durch ein hoch komplexes Zusammenspiel der beteiligten Systembestandteile aus. Zur Beherrschung der Komplexität ist eine übersichtliche und systematische Strukturierung dieser Systeme zwingend erforderlich. Häufig werden hierzu in Technik und Informatik die Prinzipien der *Modularisierung* und *Hierarchisierung*[1] angewendet. Durch die Zerlegung in kleinere, überschaubare Module lässt sich die Komplexität des Gesamtsystems reduzieren. Nach Koch werden an solche modulare mechatronische Systeme die folgenden Anforderungen gestellt [Koc05, LK05]:

- *Klare Definition von Bauteilschnittstellen zur Kopplung:* Bei der Strukturierung mechatronischer Systeme sind die Definition und die Spezifikation von Schnittstellen (mechanische Ankoppelstellen, informationstechnische Schnittstellen usw.) wichtig, damit die Teilkomponenten parallel entwickelt und später leicht integriert werden können.

- *Kapselung von Informationen:* Zur Steigerung der Übersichtlichkeit werden Teilsysteme als gekapselte Einheiten betrachtet, die über die Schnittstellen an die sie umgebenden Komponenten angebunden sind.

- *Hierarchischer Aufbau von komplexen Baugruppen mit Hilfe von Subsystemen:* Die Subsystemtechnik ergibt sich zwangsweise aus dem Prinzip der Hierarchisierung. So werden Teilfunktionen von Teilkomponenten übernommen, die wiederum in das Gesamtsystem eingebettet sind.

Der strukturierte Entwurf mechatronischer Systeme ist seit einigen Jahren Gegenstand von Forschungen. In [Lüc92, HSNL97, Nau00, LHL01, Koc05, Hes06] ist eine Methode entstanden, welche zu einem modular-hierarchisch strukturierten System führt und dabei sowohl die mechanische Tragstruktur, die Sensorik und Aktorik als auch die regelnde Informationsverarbeitung mit einbezieht. Diese Strukturierungs- und Entwurfsmethode hat sich in vielen Anwendungen bewährt und ist in die VDI-Richtlinie 2206 „Entwicklungsmethodik für mechatronische Systeme“ [VDI04] eingeflossen.

Der wesentliche Gedanke dieser Methode ist die Gliederung des mechatronischen Systems anhand seiner Bewegungsfunktionen. Die hieraus gebildete Bewegungsfunktionsstruktur bildet die Grundlage für die Einteilung des mechatronischen Systems in einzelne mechatronische Aggregate. Einzelnen Funktionen oder Teilbäumen der Funktionsstruktur werden Aggregate zugeordnet (vgl. [Koc05, Hes06]). Diese Aggregate integrieren die mechatronischen Grundbausteine: mechanische Tragstruktur, Aktoren, Sensoren sowie die regelnde

[1] **Modularisierung:** funktionale Kapselung von Teilmodulen auf horizontaler Ebene
Hierarchisierung: vertikale Gliederung der Systemstruktur

Informationsverarbeitung. In der Hierarchie höher gelagerte Aggregate gliedern hierbei unterlagerte Aggregate als Aktoren oder auch als Aktor-Sensorgruppen ein.

Zur Klassifizierung der mechatronischen Aggregate haben sich die folgenden Strukturelemente herauskristallisiert:

- **Mechatronisches Funktionsmodul (MFM):** Das Konzept eines mechatronischen Funktionsmoduls wurde erstmals von Lückel in [Lüc92] vorgestellt. Dieses MFM bildet das Basiselement eines mechatronischen Systems und dient der Umsetzung aktorischer Teilbewegungsfunktionen. Alle vier Grundbausteine eines mechatronischen Systems sind in einem MFM enthalten. D. h., es besteht aus einer mechanischen Tragstruktur, die über Aktoren bewegt werden kann. Anhand von Sensorsignalen berechnet die Informationsverarbeitung Steuereingriffe für die Aktoren und beeinflusst so die Bewegung der Tragstruktur.

  Ein MFM kann als ein intelligenter, geregelter Aktor angesehen werden. Als solcher kann es seinerseits in ein überlagertes MFM eingebettet werden. Aus regelungstechnischer Sicht entsteht hierdurch eine Kaskadenregelung, bei der das überlagerte Modul dem unterlagerten Modul eine Sollbewegung über Referenzsignale vorgibt. Das unterlagerte MFM reicht ggf. Sensorwerte zurück. Für diese Kommunikation muss das MFM über eine informationstechnische Schnittstelle verfügen, über die regelungstechnische Signale wie Referenz- und Sensorsignale ausgetauscht werden können.

- **Mechatronische Funktionsgruppe (MFG):** Die mechatronische Funktionsgruppe ist erst nachträglich in das Strukturierungskonzept aufgenommen worden [LHL01]. Eine MFG wird bei komplexeren Systemen benötigt, um Teilbewegungsfunktionen weiter zu untergliedern und durch eigene Strukturelemente zu realisieren. Einer MFG ist kein eigenes mechanisches Tragsystem eineindeutig zugeordnet, sie beinhaltet lediglich eine Informationsverarbeitung und zugehörige Sensorik. Da verschiedene MFG eines mechanischen Grundsystems dieselben unterlagerte Aktoren nutzen können, ist auch eine eindeutige Zuordnung von Aktoren nicht möglich. Die MFG untergliedert vielmehr die Informationsverarbeitung eines MFM oder des AMS und ist somit auch der Tragstruktur des jeweiligen MFM bzw. des AMS zugeordnet.

- **Autonomes mechatronisches System (AMS):** Das AMS bildet das oberste Strukturelement das über eine eigene mechanische Tragstruktur verfügt und kommt daher innerhalb dieser Tragstruktur nur einmal vor. Es verbindet die unterlagerten MFM durch mechanische Kopplungen.

  Das AMS realisiert die Hauptbewegung des Systems und ist in der Lage, autonom in seiner Umgebung zu agieren. Im Falle eines mobilen System muss es daher mit einer eigenen Energieversorgung ausgestattet sein. Zusätzlich zur verbindenden Tragstruktur kann das AMS über Sensoren und eine eigene Informationsverarbeitung verfügen. Die Aktoren des AMS werden in der Regel durch die unterlagerten MFM realisiert. Ein mobiles System wie ein PKW oder auch das RailCab sind typische Beispiele für ein AMS. Aber auch ein in sich abgeschlossenes stationäres System wie ein Fertigungsroboter kann als AMS aufgefasst werden.

- **Vernetztes mechatronisches System (VMS):** Das VMS wurde von Honekamp und anderen in [HSNL97] eingeführt, um Fahrzeugverbände in das Strukturierungskonzept einzubinden. Werden mehrere AMS miteinander verkoppelt, entsteht ein

vernetztes mechatronisches System. Auf der Ebene der VMS findet nur noch eine rein informationstechnische Kopplung statt, welche aber eine dynamische Änderung der Struktur während des Betriebs ermöglicht. Dabei ist es nicht unbedingt notwendig, dem VMS eine eigene Rechenhardware zur Verfügung zu stellen. Die Funktionen des VMS können bspw. in der Rechenhardware des AMS implementiert werden. Das VMS entsteht durch die informationstechnische Verbindung mehrerer AMS. Eine eigene mechanische Tragstruktur existiert auf der VMS-Ebene nicht, demnach sind auch keine Aktoren auf dieser Ebene notwendig. Im Normalfall bedient sich das VMS der Sensorik der AMS, oder es werden Informationen durch Kommunikation mit anderen nicht-mechatronischen Systemen ausgetauscht. Fahrzeugkonvois und miteinander kooperierende Fertigungsroboter sind Beispiele für VMS.

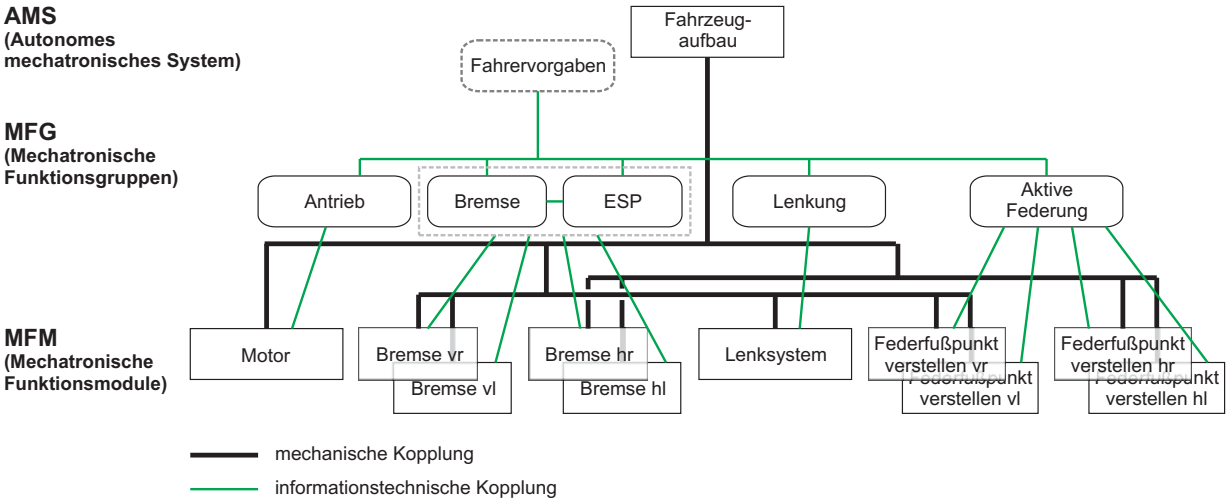


Abbildung 2.1: Strukturebenen eines mechatronischen Systems am Beispiel eines PKWs

.

2.2 Struktur der Informationsverarbeitung

Die Bewegungsfunktionen eines Aggregates werden von regelungstechnischen Komponenten auf einem digitalen Rechensystem umgesetzt und kontrolliert. Jedem Strukturelement – MFM, MFG, AMS und VMS – eines mechatronischen Systems wird dabei eine eigene digitale Informationsverarbeitung zugeordnet. Im Folgenden wird die Informationsverarbeitung auf der Ebene des einzelnen Strukturelements, die *Mikrostruktur*, sowie die informationstechnische Vernetzung aller Elemente, die *Makrostruktur*, vorgestellt.

2.2.1 Mikrostruktur

Die Informationsverarbeitung eines MFM, MFG oder auch AMS muss eine Vielzahl von Funktionen erfüllen. Hierzu gehören die:

- Quasi-kontinuierliche Regelung der Bewegung des Systems
- Überwachung der Strecke auf Fehlfunktionen und Ausführung von Notfallroutinen

- Steuerung von diskreten Abläufen (z. B. Anfahr- und Ausschaltvorgänge)
- Kommunikation von Referenz- und Messwerten an benachbarte Funktionsmodule

Die Realisierung eines selbstoptimierenden Systems erfordert weiterhin die Integration zusätzlicher Funktionalitäten:

- Anpassung der Regelung an neue Betriebsbedingungen und Zielsetzungen durch Parameter- und Strukturanpassung
- Optimierung der Regelungen unter Berücksichtigung neuer, unbekannter Situationen und vorausschauende Planung zur Bewältigung zukünftiger Situationen
- Speicherung und Bereitstellung von Optimierungs- und Planungsergebnissen
- Kommunikation mit anderen Funktionsmodulen (z. B. Systemzustände, Zielvorgaben oder Anregungsinformationen)

Die Umsetzung dieser zum Teil sehr komplexen Funktionen erfordert eine geeignete Strukturierung der Informationsverarbeitung. Naumann schlägt mit dem Operator-Controller-Modul (OCM) einen übersichtlichen Strukturierungsansatz vor, der speziell auf die Bedürfnisse adaptiver Systeme hin konzipiert wurde [Nau00]. Dieses zweiteilige OCM unterteilt die verschiedenen Funktionen in quasi-kontinuierliche Funktionen, die im Controller ausgeführt werden und ereignisdiskrete Funktionen, die im Operator angesiedelt sind. Die Ankopplung des technischen Systems erfolgt ausschließlich über den Controller. Durch den Controller erfolgt die Regelung des Systems unter Einhaltung von harten Echtzeitbedingungen[2]. Er beeinflusst somit direkt die Dynamik des Systems. Durch den Operator erfolgt die Adaption der Regelung, d. h. die Berechnung der notwendigen Anpassungen an geänderte Betriebs- und Umgebungsbedingungen. Zur Realisierung des Operators schlägt Naumann den Einsatz von Workflowmodellen vor, die ähnlich wie Zustandsautomaten, z. B. Statecharts, beschrieben werden. Adaptions- und Optimierungsalgorithmen sind demnach dem Operator zuzuordnen. Seine Arbeitsweise erfolgt asynchron zum Controller.

Auf Basis des von Naumann entwickelten OCM wurde von Oberschelp und anderen in [OHG04a] eine erweiterte Struktur für die Informationsverarbeitung von selbstoptimierenden MFM, MFG und AMS vorgeschlagen (vgl. auch [FGK+04, Hes06, Obe08]). Dieses erweiterte OCM besteht aus den drei Verarbeitungsebenen Controller, reflektorischer Operator und kognitiver Operator. Abbildung 2.2 zeigt diese dreiteilige OCM-Struktur.

[2] Ein Rechnersystem unterliegt Echtzeitanforderungen, wenn das Ergebnis einer Berechnung innerhalb einer vorgegebenen Zeitspanne, also „rechtzeitig“, vorliegen muss. Darüber hinaus wird zwischen *harter* und *weicher* Echtzeit unterschieden. In dieser Arbeit wird eine Definition verwendet, die sich an dem Nutzen und den Folgen für das System orientiert.

Von *harten Echtzeitbedingungen* wird hier gesprochen, wenn das Einhalten der definierten Zeitspanne zwingend erforderlich ist. Bei einer Verletzung ist das Ergebnis der Berechnung unbrauchbar und die Funktionalität ist nicht mehr gewährleistet. Dies kann z. B. bei der Regelung mechatronischer Systeme katastrophale Folgen haben.

Bei *weichen Echtzeitanforderungen* hat die Verletzung der Zeitspanne keine katastrophale Folge für das System. Geringe Überschreitungen der Zeitspanne liefern noch brauchbare Ergebnisse, mit fortschreitender Verletzung verringert sich jedoch der Nutzen.

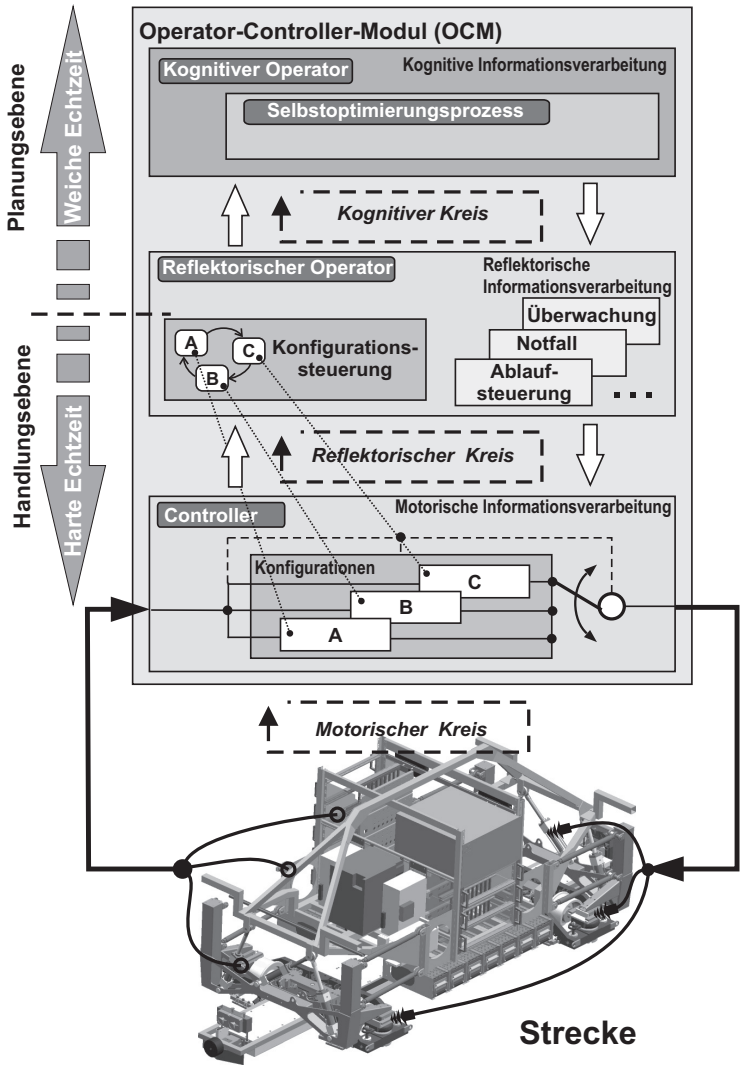


Abbildung 2.2: Aufbau des dreiteiligen Operator-Controller-Moduls [OHG04a]

Das OCM stellt eine pragmatische, aus praktischer Erfahrung mit Anwendungen erwachsene Struktur dar. Zum einen orientiert sich die Aufteilung in Controller und Operator an funktionalen Aspekten, wie der Art des Durchgriffs auf das technische System. Der Controller besitzt direkten, der Operator nur einen indirekten Zugriff auf das System. Zum anderen wird die Trennung in kognitiver und reflektorischer Operator aufgrund von Echtzeitanforderungen vorgenommen.

Ziel dieses Strukturierungsansatzes ist es, die verschiedenen Funktionen anhand ihres Einflusses auf das System und ihrer Beziehung zueinander zu ordnen. Dabei sollen aber bereits wesentliche Anforderungen an die Rechentechnik und das Betriebssystem frühzeitig im Entwurf berücksichtigt werden. Eine konkrete Zuordnung zu einer Realisierungsplattform wird durch das OCM jedoch nicht vorgegeben. So können die verschiedenen Elemente des OCM z. B. zusammen auf einem Rechensystem oder verteilt auf mehrere Rechensysteme implementiert werden.

Die drei Ebenen des OCM werden im Folgenden beschrieben:

- Die unterste Ebene des OCM bildet der **Controller**. Die Aufgabe des Controllers besteht in der Beeinflussung des dynamischen Verhaltens des technischen Systems. Er

enthält daher die regelungstechnischen Bestandteile der Informationsverarbeitung. In direkter Wirkkette werden Messsignale aufgenommen, Stellsignale berechnet und ausgegeben. Dieser Regelkreis wird auch als *motorischer* Kreis bezeichnet. Die regelungstechnischen Komponenten arbeiten üblicherweise quasi-kontinuierlich unter harten Echtzeitbedingungen. Der Controller kann mehrere Regler enthalten, zwischen denen je nach Aufgabe umgeschaltet wird. Hierzu können verschiedene Umschaltstrategien zum Einsatz kommen, deren quasi-kontinuierlichen Bestandteile ebenfalls im Controller implementiert sind. In [OMH+08] wird bspw. eine Strategie zum „glatten“ Umschalten[3] zwischen verschiedenen Reglern und Regelungsaufgaben vorgestellt.

- Der **reflektorische Operator** überwacht und steuert den Controller. Alle Hilfsfunktionen die ein „reflexartiges“ Eingreifen erfordern, sind hier angesiedelt. Hierzu gehören Funktionen wie Ablaufsteuerung, Überwachungs- und Notfallroutinen, aber auch die Routinen zur Anpassung der regelungstechnischen Komponenten des Controllers. Der reflektorische Operator besitzt keinen direkten Durchgriff auf die Aktorik des Systems. Zur Anpassung des Systemverhaltens modifiziert er die Regelung, indem er Parameter- oder Strukturänderungen im Controller initiiert.

  Die Implementierung des reflektorischen Operators arbeitet überwiegend ereignisorientiert, ist jedoch nicht hierauf beschränkt. So erfordern verschiedene Funktionen zur Überwachung des Regelungsverhaltens oder zur Aufbereitung und Analyse von Messsignalen eine quasi-kontinuierliche Verarbeitung. Ebenso werden im Rahmen der Ablaufsteuerung bei bestimmten Systemzuständen quasi-kontinuierliche Sollwerte an die Regelung im Controller weitergereicht.

  Die enge Verknüpfung mit dem Controller sowie die sicherheitskritischen Überwachungsfunktionen erfordern die Einhaltung harter Echtzeitbedingungen. Als Verbindungsebene zwischen dem Controller und dem kognitiven Operator stellt der reflektorische Operator eine Schnittstelle zwischen den nicht echtzeitfähigen bzw. mit weicher Echtzeit arbeitenden Elementen und dem Controller zur Verfügung. Er nimmt Ergebnisse des kognitiven Operators entgegen, überprüft sie und übermittelt sie zu definierten Zeitpunkten an den Controller. Im Gegenzug werden Messwerte vom Controller vorverarbeitet, zwischengespeichert und an den kognitiven Operator hochgereicht.

  Der reflektorische Operator ist weiterhin für die ereignisdiskrete Echtzeitkommunikation zwischen mehreren OCM verantwortlich.

- Der **kognitive Operator** bildet die oberste Ebene des OCM. Auf dieser Ebene kann das System durch Anwendung vielfältiger Methoden (etwa Optimierungsverfahren, Lern- oder Planungsverfahren) Wissen über sich und die Umgebung nutzen um das eigene Verhalten bezüglich des Zielsystems zu verbessern. Mit Hilfe von Optimierungsverfahren können Systemeinstellungen gefunden werden, die bezüglich der aktuellen Betriebsbedingungen und Zielsetzungen hin optimal sind. Planungsverfahren erlauben das vom realen System zeitlich entkoppelte Antizipieren zukünftiger Situationen (siehe [MAK+08]). Dies ermöglicht es, bereits im Vorfeld Maßnahmen zur Bewältigung der Situationen zu ermitteln. Während sowohl Controller als auch

[3] Mit „glatt“ sei hier ein Umschaltvorgang charakterisiert, der ohne Sprung, Knick oder Ruck etc. in den Stellsignalen der Aktoren stattfindet.

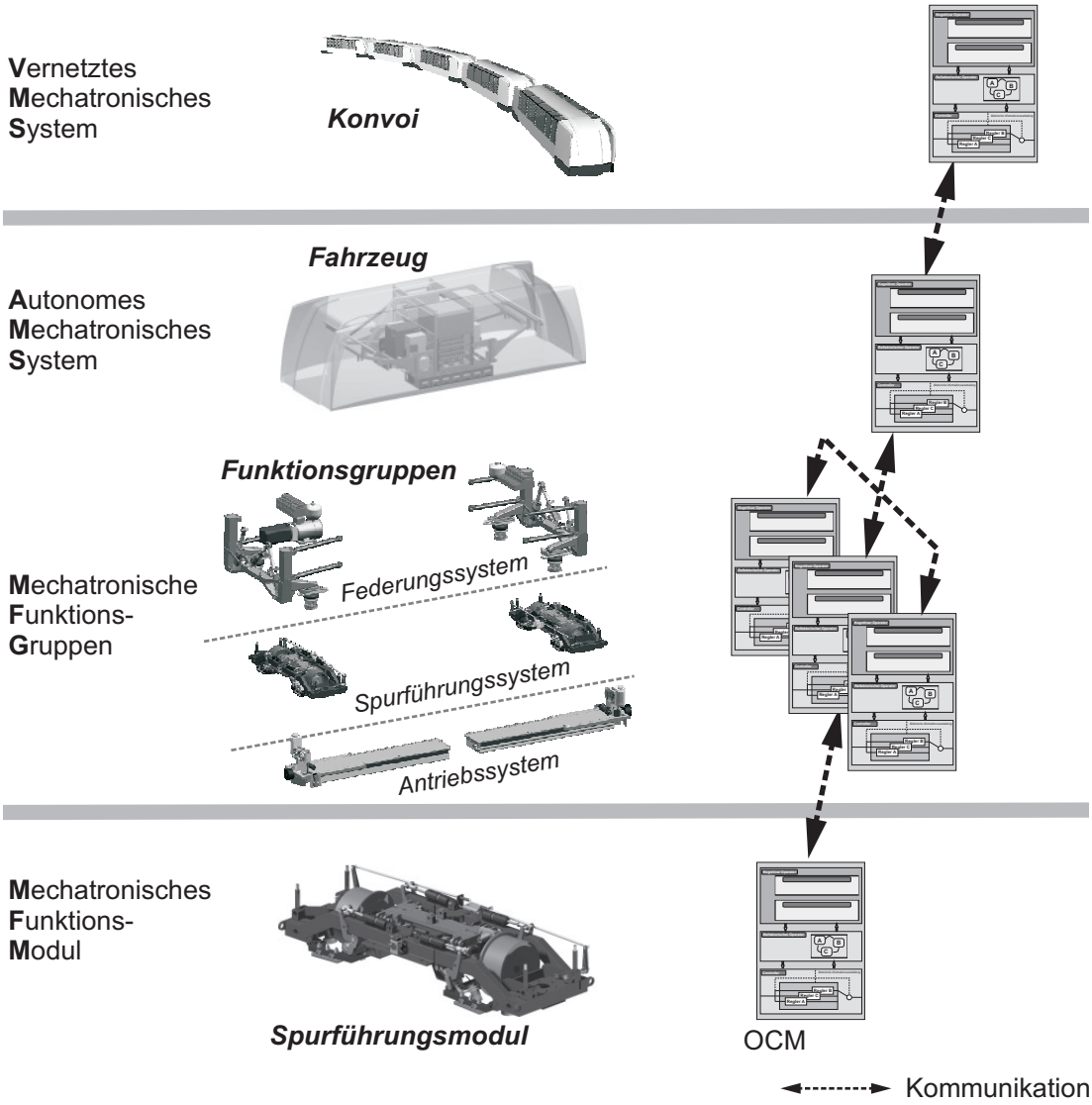


Abbildung 2.3: Das OCM innerhalb der mechatronischen Strukturebenen (am Beispiel des RailCab-Systems)

.

reflektorischer Operator harten Echtzeitanforderungen unterliegen, arbeitet der kognitive Operator asynchron zur Realzeit. Dabei ist aber selbstverständlich auch eine Antwort innerhalb eines gewissen Zeitfensters erforderlich, da die Selbstoptimierung aufgrund veränderter Umgebungsbedingungen sonst zu keinen verwertbaren Ergebnissen käme. Der kognitive Operator unterliegt folglich weicher Echtzeit.

2.2.2 Makrostruktur

Die Informationsverarbeitung des gesamten Systems ist aus den informationsverarbeitenden Einheiten der einzelnen Aggregate zusammengesetzt. Da die regelungstechnischen Komponenten der Umsetzung der verschiedenen Bewegungsaufgaben dienen, orientiert sich der Signalfluss zwischen den Reglern an der Bewegungsfunktionsstruktur und somit direkt an der Aggregatstruktur des Systems.

Im Kontext selbstoptimierender Systeme wird in [OHG04a] vorgeschlagen, die Informationsverarbeitung der einzelnen Strukturelemente durch Operator-Controller-Module zu realisieren. Bild 2.3 zeigt die Zuordnung von OCM an die Aggregate des RailCab Systems. Jedem Element der hier dargestellten Aggregatstruktur ist ein eigenes OCM zur Realisierung der jeweiligen Funktionen zugeordnet. Die Kommunikationskanäle zwischen den OCM verlaufen parallel zu den Koppelbindungen der Aggregatstruktur und orientieren sich an der funktionalen Beziehung zwischen den Aggregaten. Da die Abhängigkeiten eine Baumstruktur aufweisen, ergibt sich ebenfalls eine hierarchische Kommunikationsstruktur.

- **Kommunikation zwischen Controllern:** Die für die Systemdynamik relevanten quasi-kontinuierlichen Signale werden über die Controller-Kommunikationskanäle in harter Echtzeit ausgetauscht. Über diese Kanäle werden Soll- bzw. Referenzsignale, aber auch Signale zur Störgrößenkompensation oder Messsignale für die regelungstechnischen Komponenten übermittelt. Wie bereits erwähnt, orientiert sich die Kommunikation an der Funktionsstruktur des Systems. Dabei reicht ein Regler, der einer bestimmten Bewegungsfunktion zugeordnet ist, Sollsignale an die Regler unterlagerter Teilbewegungsfunktionen weiter. Hierdurch ergibt sich eine hierarchische, gerichtete Kommunikation, wie sie in Abbildung 2.4 dargestellt ist. Der Datenaustausch von unterlagerten hin zu überlagerten Ebenen sollte dabei nach Möglichkeit auf ein Minimum, z.B. nur zur Übertragung von Messgrößen, reduziert werden, um eine übersichtliche Struktur zu erhalten und Schleifen zwischen den von den Controllern ausgetauschten Signalen zu vermeiden.

- **Kommunikation zwischen reflektorischen Operatoren:** Über die Kommunikationskanäle zwischen den reflektorischen Operatoren können Ereignisse unter harten Echtzeitbedingungen ausgetauscht werden. Mit diesen Signalen können Umschaltungen zwischen verschiedenen Systemzuständen ausgelöst aber auch Fehlerfälle an benachbarte Module übermittelt und koordinierte Reaktionen zwischen den OCM abgestimmt werden.

- **Kommunikation zwischen kognitiven Operatoren:** Grundsätzlich kann die Kommunikation zwischen kognitiven Operatoren relativ frei gestaltet werden und sollte sich nach den Bedürfnissen und Anforderungen der jeweils implementierten Verfahren richten. Die in Abbildung 2.4 dargestellte Baumstruktur ist dabei lediglich eine Möglichkeit. Sie bietet sich jedoch an, da so eine konsistente Kommunikationsstruktur für alle drei Ebenen des OCM erreicht wird. Der Signalaustausch auf der Ebene der kognitiven Operatoren erfolgt ereignisdiskret unter weichen Echtzeitanforderungen.

  In Kapitel 7 wird ein Konzept für die Ausprägung der einzelnen kognitiven Operatoren und die Kommunikation zwischen ihnen vorgestellt.

Speziell auf den unteren Strukturebenen (MFG, MFM sowie deren unterlagerte MFM) wird eine hierarchische Kommunikationsstruktur empfohlen, bei der jedes OCM über einen Kommunikationskanal mit genau einem überlagerten OCM und mehreren unterlagerten OCM verbunden ist. Zum einen spiegelt diese Struktur die Bewegungsfunktionshierarchie wider und entspricht so dem Signalfluss zwischen den regelungstechnischen Komponenten, zum anderen erfordern die starken physikalischen Abhängigkeiten zwischen den Aggregaten

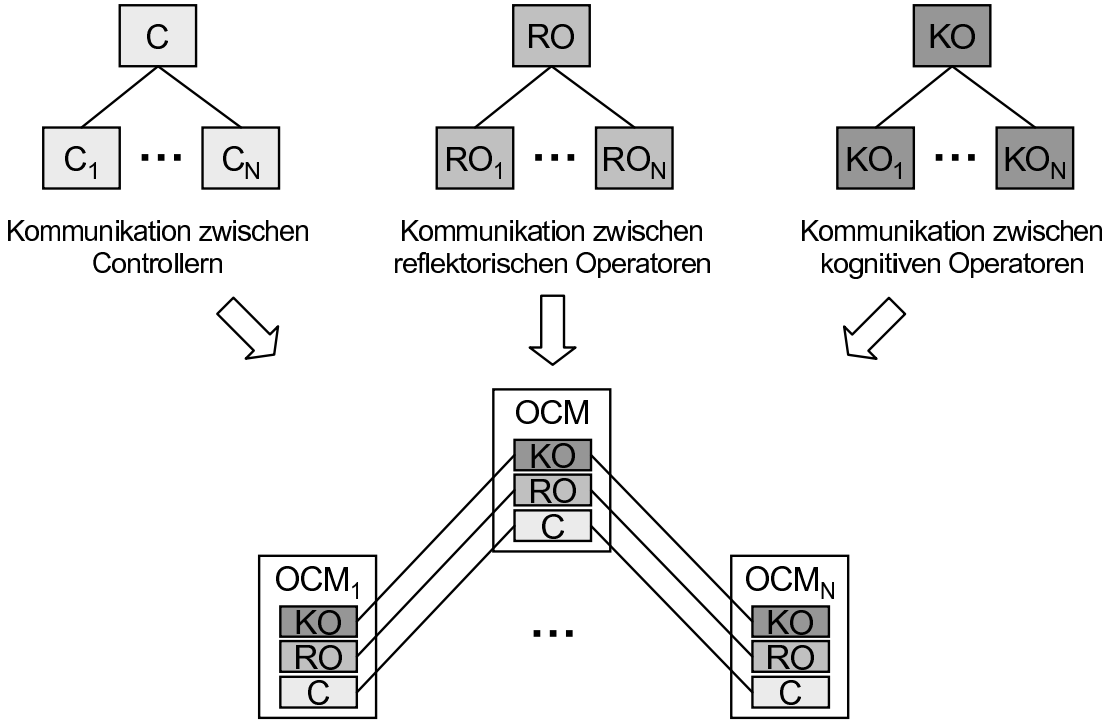


Abbildung 2.4: Kommunikationskanäle zwischen den drei Ebenen des OCM nach [OHG04b]

.

der unteren Ebenen eine direkte Kontrolle durch das jeweils überlagerte OCM. Kontrolleingriffe weiterer OCM können hier zu unerwünschten Seiteneffekten führen.

Demgegenüber kann auf den oberen Strukturebenen (VMS, AMS, aber auch MFG), bei geringer ausgeprägten physikalischen Abhängigkeiten, eine Kommunikation direkt zu benachbarten OCM gleicher Ebene erfolgen. Speziell bei den MFG greifen mitunter mehrere MFG der gleichen Strukturebene auf die selben unterlagerten MFM zu. Eine Koordination oder Entkopplung dieser Zugriffe ist zwingend erforderlich, so dass hier eine direkte Kommunikation zwischen diesen MFG erforderlich sein kann.

3 Grundlagen der Optimierung

Unter Optimierung versteht man im mathematischen Sinne das Aufsuchen des kleinsten (Minimierung) oder größten (Maximierung) Wertes einer mathematischen Funktion mehrerer Veränderlicher. [Bro87]

In Wissenschaft und Technik kommen Optimierungsverfahren immer häufiger zur Lösung verschiedener Probleme zum Einsatz. Dieser Trend besteht im Wesentlichen aus den folgenden Gründen. Aus dem rasanten Anstieg der Rechenleistung der letzten Jahrzehnte folgt der Einsatz immer detaillierterer Problembeschreibungen, deren komplexe Zusammenhänge sich oft nur noch durch den Einsatz numerischer Methoden beherrschen lassen. Optimierungsverfahren bieten sich als ein Werkzeug zur Lösung solcher Probleme an, da sie universell eingesetzt werden können und einen hohen Gestaltungsspielraum bei der Formulierung der Optimierungsprobleme aufweisen.

Der Entwurf eines technischen Systems mit Hilfe der Optimierung geht von der präzisen Formulierung der Anforderungen und Ziele aus. Aus der Sicht eines Entwicklers wird mit der Optimierung ein anderes Entwurfsparadigma verfolgt, der typische Lösungsweg wird im Grunde invertiert. Anstelle die „Entwurfsschrauben“ des Problem so lange zu verdrehen, bis das Ergebnis den Anforderungen genügt, werden die Anforderungen und Ziele mathematisch präzise formuliert und die gewünschten Entwurfsparamerter über ein Optimierungsverfahren bestimmt.

Im Gebiet der Optimierung existiert eine große Bandbreite an unterschiedlichen Problemklassen, auf die hier nicht im einzelnen eingegangen werden soll. Eine strukturierte Übersicht über die verschiedenen Klassen findet sich unter [GMW81, NEO10].

Je nach Problemklasse existieren eine Vielzahl an Lösungsverfahren. In dieser Arbeit werden nur Verfahren für kontinuierliche Optmierungsprobleme, die auf der Basis von Gradienteninformation arbeiten, betrachtet. Diese Verfahren besitzen eine hohe Konvergenzrate und eignen sich gut für Probleme, bei denen die Berechnung der Zielfunktionen den Großteil des Rechenaufwandes ausmacht. Der Nachteil dieser Verfahren ist, dass nur eine lokale Optimierung stattfindet. Es kann nicht garantiert werden, dass die gefundene Lösung das globale Optimum darstellt.

3.1 Unbeschränkte nichtlineare Optimierung

Das Problem der unbeschränkten nichtlinearen Optimierung besteht darin, einen Parametervektor $\underline{x}$ zu finden, der den Wert einer skalaren Zielfunktion $f(\underline{x})$ minimiert[1]. Das

[1] Im Folgenden wird unter Optimierung die Minimierung einer Zielfunktion verstanden. Ein Maximierungsproblem lässt sich durch Multiplikation der Zielfunktion mit -1 stets als ein Minimierungsproblem formulieren

Optimierungsproblem wird mathematisch als

$$\min_{\underline{x}\in\mathbb{R}^n} f(\underline{x}) \tag{3.1}$$

formuliert. Der Lösungsvektor des Minimierungsproblems wird mit $\underline{x}^\star$ bezeichnet und der zugehörige minimale Zielfunktionswert mit $f^\star$.

Die im Folgenden betrachteten Verfahren eignen sich zur Minimierung einer hinreichend glatten Zielfunktion f. Diese Verfahren ermitteln eine Folge von Näherungslösungen von $\underline{x}^\star$ nach der Iterationsvorschrift

$$\underline{x}_{k+1} = \underline{x}_k + \alpha_k \underline{d}_k \tag{3.2}$$

wobei der Startpunkt $\underline{x}_0$ beliebig gewählt werden kann. Im Allgemeinen ist es jedoch sinnvoll, als Startpunkt eine gute Anfangsschätzung der Lösung vorzugeben. Bei dieser Iteration gibt $\underline{d}$ eine Suchrichtung zur Minimierung des Problems und α die zugehörige Schrittweite in dieser Suchrichtung an. Die Bestimmung der Schrittweite erfolgt oft durch eine Liniensuche entlang der Suchrichtung und stellt daher ein eigenständiges, eindimensionales Minimierungsproblem dar. Häufig werden hier Verfahren eingesetzt wie *Intervallhalbierung*, *Golden-Section-Search* sowie quadratische oder kubische Interpolationsverfahren. Eine Beschreibung dieser Verfahren findet sich u. a. in [GMW81].

3.1.1 Verfahren des steilsten Abstiegs

Bei dem Verfahren des steilsten Abstiegs handelt es sich um einen sehr intuitiven Ansatz zur Minimierung von f. Die Suchrichtung ergibt sich hier direkt aus dem Gradienten der Zielfunktion:

$$\underline{d}_k = -\nabla f(\underline{x}_k) \tag{3.3}$$

Ein entscheidender Nachteil dieses Verfahrens ist das langsame Konvergenzverhalten. Bei schlecht konditionierten Problemen werden vor allem in der Nähe des Optimums sehr kleine Schrittweiten erzeugt.[2] Bereits bei quadratischen Problemen benötigt das Verfahren im Allgemeinen eine sehr große Anzahl an Optimierungsschritten, um das Minimum innerhalb einer konkreten Konvergenzschranke anzunähern. Aus diesem Grund spielt es bei praktischen Problemen eine eher untergeordnete Rolle.

Verfahren mit konjugierten Richtungen verbessern das Konvergenzverhalten, indem sie Informationen über das Krümmungsverhalten der Zielfunktion berücksichtigen. Hierzu wird die Suchrichtung des vorangegangenen Schrittes $\underline{d}_{k-1}$ in die Suchrichtungsbestimmung des aktuellen Schrittes mit einbezogen.

$$\underline{d}_k = -\nabla f(\underline{x}_k) + \beta_k \underline{d}_{k-1} \tag{3.4}$$

Die beiden bekanntesten Verfahren der konjugierten Richtungen sind die Verfahren nach Fletcher und Reeves mit:

$$\beta_k = \left(\frac{\|\nabla f(\underline{x}_k)\|_2}{\|\nabla f(\underline{x}_{k-1})\|_2} \right)^2 \tag{3.5}$$

[2] Das Verfahren des steilsten Abstiegs besitzt eine lineare Konvergenzrate. Dies bedeutet, dass eine positive Konstante η existiert, für die $\|\underline{x}_{k+1} - \underline{x}^\star\| < \eta \|\underline{x}_k - \underline{x}^\star\|$ gilt.

sowie nach Polak und Ribiere mit:

$$\beta_k = \frac{[\nabla f(\underline{x}_k) - \nabla f(\underline{x}_{k-1})]^{\mathrm{T}} \nabla f(\underline{x}_k)}{\nabla f(\underline{x}_{k-1})^{\mathrm{T}} \nabla f(\underline{x}_{k-1})} \tag{3.6}$$

Die Konvergenz des Verfahrens kann manchmal gesteigert werden, indem β_k periodisch auf Null gesetzt wird.

Bei quadratischen Zielfunktionen hat das konjugierte Gradientenverfahren die Eigenschaft, das Minimum in einer endlichen Anzahl an Schritten zu berechnen. In diesem Fall verhalten sich die Methoden von Fletcher-Reeves und das Verfahren von Polak-Ribiere äquivalent. Bei anderen nichtlinearen Zielfunktionen hat sich die Methode von Polak-Ribiere als die effizientere erwiesen [OTC03].

3.1.2 Newton-Verfahren

Das Newton-Verfahren geht von der Annahme aus, dass eine Zielfunktion f an der Stelle $\underline{x}_k$ durch eine quadratische Funktion

$$q(\underline{r}) = \frac{1}{2}\, \underline{r}^{\mathrm{T}} \underline{\mathbf{A}}\, \underline{r} + \underline{b}^{\mathrm{T}} \underline{r} + c \tag{3.7}$$

approximiert werden kann. Ist die Matrix $\underline{\mathbf{A}}$ positiv definit, kann das Minimum der quadratischen Funktion direkt angegeben werden und man erhält:

$$\underline{r}^{\star} = \underline{\mathbf{A}}^{-1} \underline{b} \tag{3.8}$$

Diese Lösung wird im Newton-Verfahren zur Bestimmung einer neuen Suchrichtung $\underline{d}_k$ verwendet.

Mit $\underline{\mathbf{A}} = \nabla^2 f(\underline{x}_k)$ und $\underline{b} = \nabla f(\underline{x}_k)$ entspricht die quadratische Approximation (3.7) einer Taylorreihenentwicklung von $f(\underline{x}_k)$ mit Abbruch nach dem zweiten Glied. Setzt man nun $\underline{d}_k = \underline{r}^{\star}$ erhält man die Vorschrift

$$\underline{d}_k = -\nabla^2 f(\underline{x}_k)^{-1}\, \nabla f(\underline{x}_k) \tag{3.9}$$

zur Berechnung der neuen Suchrichtung. Die Richtung $\underline{d}_k$ wird demnach durch eine Invertierung der Hesse-Matrix gewonnen.

Das Newton-Verfahren besitzt im Vergleich zum Verfahren des steilsten Abstiegs den Vorteil einer oftmals sehr schnellen Konvergenz und ist ihm diesbezüglich weitaus überlegen.[3]

Durch Hinzuziehen der Hesse-Matrix liefert das Verfahren nicht nur Informationen über die Richtung, sondern auch über die Entfernung zum Minimum. Daher kann in der Nähe der Lösung auf eine explizite Liniensuche zur Bestimmung der Schrittweite verzichtet werden. In diesem Fall kann die Schrittweite in (3.2) auf $\alpha_k = 1$ gesetzt werden:

$$\underline{x}_{k+1} = \underline{x}_k - \nabla^2 f(\underline{x}_k)^{-1}\, \nabla f(\underline{x}_k) \tag{3.10}$$

[3] Falls $\underline{x}_0$ hinreichend nahe an der Lösung $\underline{x}^{\star}$ ist, konvergiert das Newton-Verfahren quadratisch gegen diese Lösung. Quadratische Konvergenz bedeutet, dass eine positive Konstante η existiert, für die die Ungleichung $\|\underline{x}_{k+1} - \underline{x}^{\star}\| < \eta \|\underline{x}_k - \underline{x}^{\star}\|^2$ erfüllt ist.

Diesen Vorteilen stehen jedoch einige Nachteile gegenüber:

- Der gravierendste Nachteil ist, dass in jedem Iterationsschritt die Hesse-Matrix der Zielfunktion berechnet werden muss. In vielen praktischen Anwendungen kann die Hesse-Matrix nicht direkt bestimmt werden. Eine Berechnung geschieht daher oft numerisch mit Hilfe von Differenzenquotienten. Der Rechenaufwand hierfür ist sehr hoch und ebenso sind die Ergebnisse oft zu ungenau und führen zu Konvergenzproblemen.

- Bei großen Problemen kann das Lösen des Gleichungssystems (3.9) sehr aufwändig sein.

- Bei allgemeinen Problemen kann die Hesse-Matrix nicht positiv-definit sein. In diesem Fall stellt der Vektor $\underline{d}_k$ keine Abstiegsrichtung dar.

3.1.3 Quasi-Newton-Verfahren

Das Quasi-Newton-Verfahren arbeitet ähnlich zu dem Newton-Verfahren, vermeidet aber die Berechnung der Hesse-Matrix. Diese wird auf der Basis des vorangegangenen Optimierungsschrittes approximiert, wodurch der Rechenaufwand pro Optimierungsschritt deutlich verringert wird. Die Suchrichtung ergibt sich zu

$$\underline{d}_k = -\underline{\mathbf{H}}_k^{-1} \nabla f(\underline{x}_k) \tag{3.11}$$

wobei die sogenannte Quasi-Newton-Matrix $\underline{\mathbf{H}}_k$ eine Näherung von $\nabla^2 f(\underline{x}_k)$ darstellt. Die beiden bekanntesten Verfahren zur Berechnung dieser Matrix sind das Verfahren von Davidon-Fletcher-Powell (DFP-Verfahren) sowie das Verfahren von Broyden-Fletcher-Goldfarb-Shanno (BFGS-Verfahren). Bei verschiedenen praktischen und theoretischen Untersuchungen hat sich das BFGS-Verfahren als das praktikablere erwiesen [Dix71, OTC03].

Die Update-Formel des BFGS-Verfahrens lautet

$$\underline{\mathbf{H}}_{k+1} = \underline{\mathbf{H}}_k + \frac{\underline{y}_k \underline{y}_k^{\mathrm{T}}}{\underline{y}_k^{\mathrm{T}} \underline{s}_k} - \frac{\underline{\mathbf{H}}_k \underline{s}_k \underline{s}_k^{\mathrm{T}} \underline{\mathbf{H}}_k}{\underline{s}_k^{\mathrm{T}} \underline{\mathbf{H}}_k \underline{s}_k} \tag{3.12}$$

mit

$$\underline{y}_k = \nabla f(\underline{x}_{k+1}) - \nabla f(\underline{x}_k)$$
$$\underline{s}_k = \underline{x}_{k+1} - \underline{x}_k$$

Als initiale Quasi-Newton-Matrix $\underline{\mathbf{H}}_0$ kann prinzipiell jede positiv definite Matrix gewählt werden. In den meisten Fällen kommt hier die Einheitsmatrix $\underline{\mathbf{I}}$ oder eine Startnäherung der Hesse-Matrix zum Einsatz.

Anstatt die Hesse-Matrix schrittweise anzunähern, ist es ebenfalls möglich eine Approximation der Inversen $\nabla^2 f(\underline{x}_k)^{-1}$ durchzuführen. Definiert man $\underline{\mathbf{B}}_k = \nabla^2 f(\underline{x}_k)^{-1}$, so ergibt sich die Update-Formel für $\underline{\mathbf{B}}_k$ zu

$$\underline{\mathbf{B}}_{k+1} = \left(\underline{\mathbf{I}} - \frac{\underline{s}_k \underline{y}_k^{\mathrm{T}}}{\underline{y}_k^{\mathrm{T}} \underline{s}_k}\right) \underline{\mathbf{B}}_k \left(\underline{\mathbf{I}} - \frac{\underline{y}_k \underline{s}_k^{\mathrm{T}}}{\underline{y}_k^{\mathrm{T}} \underline{s}_k}\right) + \frac{\underline{s}_k \underline{s}_k^{\mathrm{T}}}{\underline{y}_k^{\mathrm{T}} \underline{s}_k} \tag{3.13}$$

Die Richtung $\underline{d}_k$ wird dann mit

$$\underline{d}_k = -\underline{\mathbf{B}}_k \nabla f(\underline{x}_k) \tag{3.14}$$

bestimmt. Durch die Approximation der inversen Hesse-Matrix wird das Lösen des Gleichungssystems (3.11) vermieden.

Mit der Verfügbarkeit des Quasi-Newton-Verfahrens ist das Verfahren des steilsten Abstiegs faktisch überflüssig geworden. Beide Verfahren benötigen nur Ableitungen erster Ordnung, ebenso wird bei beiden Verfahren eine Liniensuche durchgeführt. Das Quasi-Newton-Verfahren benötigt zwar mehr Rechenoperationen pro Optimierungsschritt sowie mehr Speicherplatz, diese zusätzlichen Kosten werden jedoch in den allermeisten Fällen durch den Vorteil einer schnellen Konvergenz[4] mehr als aufgewogen.

3.2 Beschränkte nichtlineare Optimierung

Die bisher beschriebenen Verfahren eigenen sich zur Behandlung von unbeschränkten nichtlinearen Optimierungsproblemen. In vielen Anwendungsfällen existieren jedoch Randbedingungen oder Nebenbedingungen, die bei der Optimierung zu berücksichtigen sind. Ein solches beschränktes nichtlineares Optimierungsproblem kann durch

$$\begin{aligned} &\min_{\underline{x}\in\mathbb{R}^n} f(\underline{x}) \\ &g_j(\underline{x}) \le 0\,, \quad j = 1, \ldots, m \end{aligned} \tag{3.15}$$

definiert werden. Die Nebenbedingungen[5] $g_j(\underline{x})$ stellen ebenfalls nichtlineare Funktionen dar und definieren den zulässigen Bereich für die Lösung $\underline{x}^\star$ des Problems.

3.2.1 SQP-Verfahren

Zur Lösung beschränkter nichtlinearer Optimierungsprobleme wurden in der Vergangenheit verschiedene Verfahren entwickelt, wie beispielsweise Penalty-, Multiplikator- oder reduzierte Gradientenverfahren. In diesem Abschnitt soll jedoch nur auf Verfahren der sequentiellen quadratischen Programmierung (SQP-Verfahren) eingegangen werden, da sie sich für allgemeine Probleme, für die keine Spezialverfahren entwickelt werden können, als besonders effizient und zuverlässig erwiesen haben.[6,7]

---

[4] Das BFGS-Verfahren weist eine superlineare Konvergenz auf.

[5] Der Einfachheit halber sollen hier nur Probleme mit Ungleichheitsnebenbedingungen betrachtet werden. Prinzipiell kann eine Gleichheitsnebenbedingung der Form $g_j(\underline{x}) = 0$ durch zwei Ungleichheitsnebenbedingungen mit gegensätzlichen Vorzeichen dargestellt werden: $g_j(\underline{x}) \le 0\,, -g_j(\underline{x}) \le 0$. Aus numerischen Gründen ist eine gesonderte Behandlung von Gleichheitsnebenbedingungen jedoch sinnvoll.

[6] Es existieren eine Vielzahl an Implementierungen, die der Klasse der SQP-Verfahren zuzuordnen sind. So finden sich beispielsweise SQP-Verfahren in mathematischen Bibliotheken wie der Matlab Optimization Toolbox (Routine: `fmincon`) [Mat07b] oder der NAG-Bibliothek (Routine: `nag_opt_nlp`) [Nag02]. Weiterhin existieren alleinstehende Optimierungswerkzeuge wie NLQPL [Sch85] oder MOPS [JBL+02, Joo03].

[7] Neben den SQP-Verfahren gewinnen zunehmend auch die sog. Innere-Punkte-Verfahren (interior point method) zur Lösung nichtlinearer beschränkter Probleme an Bedeutung (siehe z. B. [FGW02, Wäc02]). In Hinsicht auf die Anwendbarkeit stellen sie eine echte Alternative für die SQP-Verfahren dar.

Im Folgenden wird ein SQP-Verfahren mit einer Active-Set-Strategie zur Lösung des beschränkten, nichtlinearen Optimierungsproblems vorgestellt, da ein solches Verfahren in den nachfolgenden Kapiteln eingesetzt wird.

Lagrange-Funktion

Für das SQP-Verfahren sind die sog. Lagrange-Multiplikatoren zur Behandlung der Nebenbedingungen von zentraler Bedeutung. Betrachten wir das beschränkte nichtlineare Optimierungsproblem (3.15) dann lautet die zugehörige Lagrange-Funktion:

$$\mathcal{L}(\underline{x}, \underline{\lambda}) = f(\underline{x}) + \sum_{j=1}^{m} \lambda_j \, g_j(\underline{x}) \tag{3.16}$$

Dabei sind die Variablen $\lambda_j \in \mathbb{R}^m$ die Lagrange-Multiplikatoren. Mit Hilfe dieser Lagrange-Multiplikatoren können die Ungleichheitsnebenbedingungen mit der Zielfunktion so „austariert“ werden, dass die Nebenbedingungen erfüllt werden und gleichzeitig f minimiert wird. Zu einer Lösung des Optimierungsproblems $\underline{x}^\star$ werden zugehörige optimale Multiplikatoren $\underline{\lambda}^\star$ gefunden.

Bezüglich der Nebenbedingungen wird zwischen *aktiven* und *inaktiven* Nebenbedingungen unterschieden. Inaktive Nebenbedingungen sind im Punkt $\underline{x}$ erfüllt und können in diesem Fall zu Null gesetzt werden. Demgegenüber sind die Lagrange-Multiplikatoren aktiver Nebenbedingungen größer Null und sorgen für das Einhalten der jeweiligen Nebenbedingungen. Für die Nebenbedingungen gilt daher:

$$\begin{aligned} \textit{aktive} \text{ Nebenbedingung:} \quad & g_j(\underline{x}^\star) = 0\,, \quad \lambda_j^\star > 0 \\ \textit{inaktive} \text{ Nebenbedingung:} \quad & g_j(\underline{x}^\star) < 0\,, \quad \lambda_j^\star = 0 \end{aligned} \tag{3.17}$$

für $j = 1, \ldots, m$.

Optimalitätsbedingung nach Karush-Kuhn-Tucker

Zur Ermittlung der optimalen Parameter $\underline{x}^\star$ und Multiplikatoren $\underline{\lambda}^\star$ wird beim SQP-Verfahren auf die notwendige Optimalitätsbedingung erster Ordnung nach Karush-Kuhn-Tucker (KKT) zurückgegriffen [KT51]:

sind f und g_j stetig differenzierbar und ist $\underline{x}^\star$ ein lokales Minimum von (3.15), dann existiert ein $\underline{\lambda}^\star$, so dass die folgenden Bedingungen erfüllt sind:

$$\begin{aligned} \nabla_{\underline{x}} \mathcal{L}(\underline{x}^\star, \underline{\lambda}^\star) &= 0 \\ g_j(\underline{x}^\star) &\le 0\,, \quad j = 1, \ldots, m \\ \lambda_j^\star &\ge 0\,, \quad j = 1, \ldots, m \\ \lambda_j^\star \, g_j(\underline{x}^\star) &= 0\,, \quad j = 1, \ldots, m \end{aligned} \tag{3.18}$$

Zusätzlich müssen im Punkt $\underline{x}^\star$ bestimmte Regularitäts-Bedingungen[8] erfüllt sein, auf die hier nicht weiter eingegangen werden soll. Die erste Gleichung in (3.18) beschreibt eine Gewichtung der Zielfunktion und der Nebenbedingungen. Im Minimum $\underline{x}^\star$ stellt der Gradient der Zielfunktion eine positive Linearkombination der Gradienten der aktiven Nebenbedingungen dar. Die zweite Bedingung verlangt das Einhalten der Nebenbedingungen. Die letzte Gleichung drückt aus, dass für inaktive Nebenbedingungen die Lagrange-Multiplikatoren

[8] engl.: *constraint qualification*

verschwinden müssen. Sie wird auch als Komplementaritätsbedingung bezeichnet und sie berücksichtigt Gl. (3.17).

Die KKT-Bedingung stellt eine notwendige Optimalitätsbedingung dar. Sind f und g_j konvexe, zweimal stetig differenzierbare Funktionen, so bildet (3.18) die notwendige und hinreichende Bedingung für ein globales Optimum.

Das quadratische Teilproblem

Neben einer Optimalitätsbedingung liefert die KKT-Bedingung auch einen Ansatz zur Lösung des beschränkten nichtlinearen Optimierungsproblems mit. Hierzu werden die Ungleichungen der KKT-Bedingung zunächst in die Form eines Gleichungssystems gebracht. Betrachtet man nur die aktiven Nebenbedingungen, so erhält man aus (3.18) und (3.17) das Gleichungssystem

$$\begin{aligned} \nabla_{\underline{x}} \mathcal{L}(\underline{x}^\star, \underline{\lambda}^\star) &= 0 \\ \underline{\hat{g}}(\underline{x}^\star) &= \underline{0} \end{aligned} \tag{3.19}$$

wobei die vektorwertige Funktion $\underline{\hat{g}}(\underline{x})$ die aktiven Nebenbedingungen enthält. Die Berechnung von $\underline{x}^\star$ und $\underline{\lambda}^\star$ kann daher in sinnvoller Weise durch Anwendung des Newton-Verfahrens auf das Gleichungssystem

$$\begin{aligned} \nabla_{\underline{x}} \mathcal{L}(\underline{x}, \underline{\lambda}) &= 0 \\ \underline{\hat{g}}(\underline{x}) &= 0 \end{aligned} \tag{3.20}$$

erfolgen. Ausgehend von einer Anfangsnäherung $[\underline{x}_0, \underline{\lambda}_0]$ wird analog zu (3.2) eine neue Näherung der Lösung nach der Iterationsvorschrift

$$\begin{bmatrix} \underline{x}_{k+1} \\ \underline{\hat{\lambda}}_{k+1} \end{bmatrix} = \begin{bmatrix} \underline{x}_k \\ \underline{\hat{\lambda}}_k \end{bmatrix} + \alpha_k \begin{bmatrix} \underline{y}_k \\ \underline{\hat{\mu}}_k \end{bmatrix} \tag{3.21}$$

bestimmt, wobei mit $\underline{\hat{\lambda}}_k$ die Lagrange-Multiplikatoren der im aktuellen Schritt aktiven Nebenbedingungen bezeichnet werden. Die Suchrichtung $[\underline{y}_k, \underline{\hat{\mu}}_k]$ wird dann gemäß dem Newton-Verfahren (3.9) durch Lösen des Gleichungssystems

$$\begin{bmatrix} \nabla^2_{\underline{x}} \mathcal{L}(\underline{x}_k, \underline{\lambda}_k) & \nabla \underline{\hat{g}}(\underline{x}_k) \\ \nabla \underline{\hat{g}}(\underline{x}_k)^{\mathrm{T}} & 0 \end{bmatrix} \begin{bmatrix} \underline{y}_k \\ \underline{\hat{\mu}}_k \end{bmatrix} = - \begin{bmatrix} \nabla_{\underline{x}} \mathcal{L}(\underline{x}_k, \underline{\lambda}_k) \\ \underline{\hat{g}}(\underline{x}_k) \end{bmatrix} \tag{3.22}$$

ermittelt. Die Matrix auf der linken Seite von (3.22) wird durch die Jacobi-Matrix von (3.20) gebildet. Die Menge der aktiven Nebenbedingungen $\underline{\hat{g}}(\underline{x}_k)$ kann zwischen den Optimierungsschritten variieren und muss daher in jedem Schritt k neu bestimmt werden.

Die Berechnung der Suchrichtung kann beim SQP-Verfahren auch als das Lösen eines quadratischen Optimierungsproblems interpretiert werden. Stellt man das quadratische Optimierungsproblem

$$\begin{aligned} \min_{\underline{d}_k} \; & \frac{1}{2} \underline{d}_k^{\mathrm{T}} \nabla^2_{\underline{x}_k} \mathcal{L}(\underline{x}_k, \underline{\lambda}) \, \underline{d}_k + \nabla_{\underline{x}_k} \mathcal{L}(\underline{x}_k) \, \underline{d}_k \\ & \nabla g_j(\underline{x}_k)^{\mathrm{T}} \underline{d}_k + g_j(\underline{x}_k) \leq 0 \end{aligned} \tag{3.23}$$

mit $j = 1, \ldots, m$ auf, so ist die Optimalitätsbedingung von (3.23) identisch zum Gleichungssystem (3.19). Viele SQP-Implementierungen setzen aus diesem Grund spezielle Algorithmen zur Optimierung quadratischer Probleme ein, um das Teilproblem der Suchrichtungsbestimmung zu lösen.

Berechnung der Hesse-Matrix

In der Praxis verzichtet man beim SQP-Verfahren aus den gleichen Gründen wie beim Newton-Verfahren auf eine explizite numerische Berechnung der Hesse-Matrix in jedem Optimierungsschritt. Anstelle dessen wird sie analog zum Quasi-Newton-Verfahren schrittweise nach der Vorschrift (3.12) angenähert. Mit der Quasi-Newton-Matrix $\underline{\mathbf{H}}_k$ als Näherung der Hesse-Matrix $\nabla^2_{\underline{x}}\mathcal{L}(\underline{x},\underline{\lambda})$ ergibt sich dann das Gleichungssystem

$$\begin{bmatrix} \underline{\mathbf{H}}_k & \nabla\hat{\underline{g}}(\underline{x}_k) \\ \nabla\hat{\underline{g}}(\underline{x}_k)^{\mathrm{T}} & 0 \end{bmatrix} \begin{bmatrix} \underline{y}_k \\ \underline{\hat{\mu}}_k \end{bmatrix} = - \begin{bmatrix} \nabla_{\underline{x}}\mathcal{L}(\underline{x}_k,\underline{\lambda}_k) \\ \hat{\underline{g}}(\underline{x}_k) \end{bmatrix} \tag{3.24}$$

zur Berechnung der Suchrichtung.

Liniensuche

Mit Lösen von (3.22) findet eine Liniensuche in Richtung $[\underline{y}_k, \underline{\hat{\mu}}_k]$ statt. Gesucht wird ein „besserer“ Punkt $\underline{x}_{k+1}$. Die alleinige Betrachtung der Zielgröße würde ein beliebig starkes Verletzen der Nebenbedingungen zulassen. Um die Nebenbedingungen zu berücksichtigen, wird daher eine sogenannte Merit-Funktion $\phi(\underline{x},\underline{\lambda})$ eingeführt und anstelle der Zielfunktion bei der Liniensuche minimiert. Die Wahl einer Merit-Funktion ist nicht trivial, da es in vielen Fällen schwierig ist, die beiden Forderungen: Minimierung der Zielfunktion und Einhaltung der Nebenbedingungen gegeneinander zu gewichten.

Beispielsweise schlägt Schittkowski in [Sch85] vor, die Lagrange-Funktion mit Hilfe einer quadratischen Norm zu bestrafen, sobald der zulässige Bereich verlassen wird:

$$\phi(\underline{x},\underline{\lambda}) = f(\underline{x}) + \sum_{j\in\mathcal{J}} \lambda_j\, g_j(\underline{x}) + \frac{1}{2}\sum_{j\in\mathcal{J}} r_j\, g_j(\underline{x})^2 - \frac{1}{2}\sum_{j\in\mathcal{K}} \lambda_j^2/r_j \tag{3.25}$$

mit

$$\begin{aligned} \mathcal{J} &= \{j : 1 \le j \le m\,,\ g_j(\underline{x}) \ge \lambda_j/r_j\} \\ \mathcal{K} &= \{1,\ldots,m\}\backslash\mathcal{J} \end{aligned}$$

Die Bestrafungsparameter r_j werden in jedem Schritt durch die Iterationsvorschrift

$$r_{j,k+1} = \max_j \left\{\lambda_{j,k}\,,\ \frac{1}{2}(r_{j,k} + \lambda_{j,k})\right\}\,, \quad j = 1,\ldots,m \tag{3.26}$$

aktualisiert.

Mit Hilfe eindimensionaler Optimierungsmethoden wird die Schrittweite α_k derart angepasst, dass der Wert der Merit-Funktion genügend verkleinert wird.

3.3 Grundlagen der Mehrzieloptimierung

Viele reale Optimierungsaufgaben lassen sich nicht durch ein einzelnes Zielkriterium beschreiben, welches minimiert werden muss. Vielmehr existieren meist mehrere oft widersprüchliche Ziele, für die ein „optimaler“ Entwurfspunkt gefunden werden muss. Bei geregelten Systemen konkurrieren bspw. Ziele wie die Schnelligkeit, die Dämpfung aber auch der Energiebedarf eines Systems miteinander.

Diese Ziele können nur bis zu einem gewissen Grad gemeinsam verbessert werden, da die Ziele im Allgemeinen kein gemeinsames Minimum besitzen. Es existiert also kein eindeutiges Minimum für diese Probleme, sondern vielmehr eine Menge „optimaler“ Punkte, die sogenannte Paretomenge.

Die Punkte dieser Menge zeichnen sich dadurch aus, dass im gesamten Entwurfsraum keine Punkte existieren, die in allen Zielen „besser“ sind. Innerhalb der Paretomenge, können zwar Punkte gefunden werden, die in bestimmten Zielen „besser“ sind; in anderen Zielen erweisen sie sich jedoch als „schlechter“.

3.3.1 Definitionen

Ein Mehrzieloptimierungsproblem (MOP) wird mathematisch folgendermaßen formuliert:

$$\min_{\underline{x}\in\mathbb{R}^n} \underline{f}(\underline{x}), \quad \underline{f}: \mathbb{R}^n \to \mathbb{R}^m \tag{3.27}$$

Hierbei ist $\underline{f}$ als ein Vektor der einzelnen, skalaren Zielfunktionen $f_1(\underline{x}), \ldots, f_m(\underline{x})$ definiert:

$$\underline{f}(\underline{x}) = \begin{bmatrix} f_1(\underline{x}) \\ \vdots \\ f_m(\underline{x}) \end{bmatrix} \tag{3.28}$$

Paretooptimalität

Bei der Minimierung dieses Vektors $\underline{f}$ greifen die skalaren Definitionen von Optimalität nicht mehr. Um Aussagen darüber treffen zu können, ob ein Zielgrößenvektor „besser“ oder „schlechter“ ist als andere, müssen zunächst Operatoren für den Vektorvergleich definiert werden. Vergleichen wir zwei Vektoren $\underline{u} \in \mathbb{R}^m$ und $\underline{v} \in \mathbb{R}^m$ miteinander, so gilt:

$$\begin{aligned} &\underline{u} \leq \underline{v} \quad \text{wenn} \quad u_i \leq v_i \quad \text{für alle} \quad i = 1, \ldots, m \\ &\underline{u} \geq \underline{v} \quad \text{wenn} \quad u_i \geq v_i \quad \text{für alle} \quad i = 1, \ldots, m \\ &\underline{u} \not\gtrless \underline{v} \quad \text{in allen anderen Fällen} \end{aligned} \tag{3.29}$$

Eine grafische Interpretation dieses Vektorvergleichs ist in Abbildung 3.1 zu sehen. In dieser Abbildung wurde ein zweidimensionales MOP mit den beiden Zielgrößen f_1 und f_2 angenommen. Der Punkt $\underline{f}(\underline{x})$ wird hier mit Punkten innerhalb der vier eingezeichneten Quadranten verglichen. Für Punkte innerhalb dieser Quadranten erweisen sich jeweils die eingezeichneten Vergleichsoperatoren als gültig.

Mit Hilfe dieses Vektorvergleichs kann die Paretooptimalität nun folgendermaßen definiert werden:

Ein Punkt $\underline{x}^\star \in \mathbb{R}^n$ ist paretooptimal, wenn kein anderer Punkt $\underline{x} \in \mathbb{R}^n$ existiert, der die Bedingung $\underline{f}(\underline{x}) \leq \underline{f}(\underline{x}^\star)$ sowie die Bedingung $f_i(x) < f_i(x^\star)$ für wenigstens ein $i \in \{1, \ldots, m\}$ erfüllt.

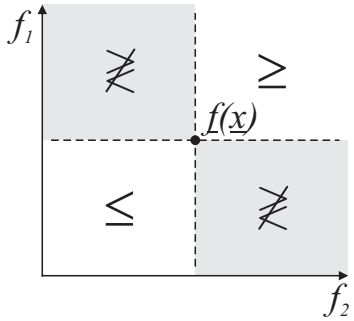


Abbildung 3.1: Vergleich von Vektoren bei MOP

Bezogen auf das Beispiel in Abbildung 3.1 bedeutet dies, dass der Punkt $\underline{f}(\underline{x})$ genau dann paretooptimal ist, wenn kein Punkt im linken, unteren Quadranten (einschließlich der gestrichelten Achsen) existiert. Weitere paretooptimale Punkte können dann nur im linken oberen oder im rechten unteren Quadranten (ausschließlich der gestrichelten Achsen) gefunden werden.

Bild- und Urbildraum

Bei einem MOP muss zwischen den optimalen Punkten im Bildraum und im Urbildraum unterschieden werden. Die Entwurfsparameter $\underline{x}$ beschreiben einen Punkt im Urbildraum, der auch als Entwurfsraum oder Parameterraum bezeichnet wird. Die Vektorfunktion $\underline{f}$ bildet den Punkt $\underline{x}$ des Urbildraums auf den Bildraum bzw. den Zielgrößenraum ab.[9] Die Darstellung 3.2 zeigt die Paretomenge eines MOP mit den Entwurfsparametern x_1 und x_2 und den Zielgrößen f_1 und f_2 im Bild- und im Urbildraum. Es sind beispielhaft zwei Punkte im Urbildraum und deren Abbildung auf den Bildraum eingezeichnet. Dabei gibt $\underline{x}^\star$ einen Punkt auf der Paretomenge und $\underline{x}$ einen Punkt abseits der Paretomenge an. Üblicherweise

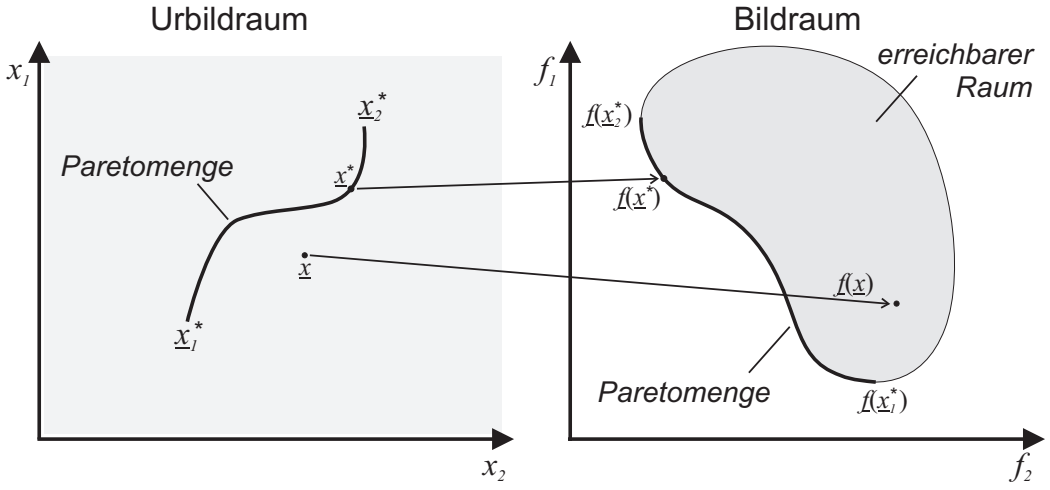


Abbildung 3.2: Paretomenge

[9] In der Literatur werden oft nur die optimalen Punkte im Urbildraum Paretopunkte genannt, wohingegen die optimalen Punkte im Bildraum als effiziente Punkte bezeichnet werden. Im Folgenden wird auf diese Unterscheidung verzichtet, wenn aus dem Zusammenhang ersichtlich wird, ob Punkte des Bild- oder Urbildraums gemeint sind.

wird die Paretomenge im Bildraum dargestellt, da hier der Zielkonflikt und die Beziehung zwischen den Zielgrößen sofort deutlich werden. Die Paretomenge wird hier durch Punkte auf dem linken unteren Rand des erreichbaren Raums gebildet und wird daher auch als Paretofront bezeichnet. Hierbei ist anzumerken, dass die Paretomenge je nach verwendeten Zielfunktionen nicht immer, wie abgebildet, zusammenhängend ist.

3.3.2 Lösungsansätze

Da für die Punkte der Paretomenge keine alternativen Punkte existieren, die in allen Zielen „besser“ sind, kann die Paretomenge als die Menge der optimalen Kompromisse angesehen werden. Der Zweck einer Mehrzieloptimierung muß daher das Auffinden eines oder mehrerer Paretopunkte sein.

Mathematisch gesehen sind alle Punkte der Paretomenge gleich „gut“. Jedoch besitzen bestimmte Kompromisse bei einem realen Entwurfsproblem ungünstige oder sogar unbrauchbare Eigenschaften. So können bei bestimmten Paretopunkten die verschiedenen Ziele ein unzweckmäßiges Verhältnis aufweisen, z. B. ein unbrauchbares Verhältnis zwischen Nutzen und Aufwand. Möglicherweise existieren auch noch weitere Randbedingungen für das reale System, die im Optimierungsproblem nicht explizit formuliert wurden bzw. die nicht auf einfache Weise formuliert werden können. Ein Entwickler ist demnach nicht an der Berechnung eines beliebigen Paretopunktes interessiert, da dieser möglicherweise seine Anforderungen trotz Paretooptimalität nicht erfüllt. Vielmehr benötigt er Methoden, mit deren Hilfe er gezielt geeignete Kompromisse bestimmen kann.

Grundsätzlich existieren zwei Herangehensweisen zur Bestimmung eines geeigneten Paretopunktes. Diese sind in Abbildung 3.3 dargestellt. Den Ausgangspunkt beider Ansätze stellt ein Mehrzieloptimierungsproblem dar und gesucht ist ein einzelner Paretopunkt, der zu einem „günstigen“ Kompromiss zwischen den verschiedenen Anforderungen an das zu optimierende System führt.

Abbildung auf eine skalare Zielfunktion: Ein häufig verwendeter Ansatz besteht darin, das MOP auf ein skalares Optimierungsproblem abzubilden und dieses anschließend z.B. mit Hilfe von klassischen Verfahren der Einzieloptimierung zu lösen. Es existieren verschiedene Ansätze, die Abbildung auf ein skalares Optimierungsproblem vorzunehmen. Zumeist

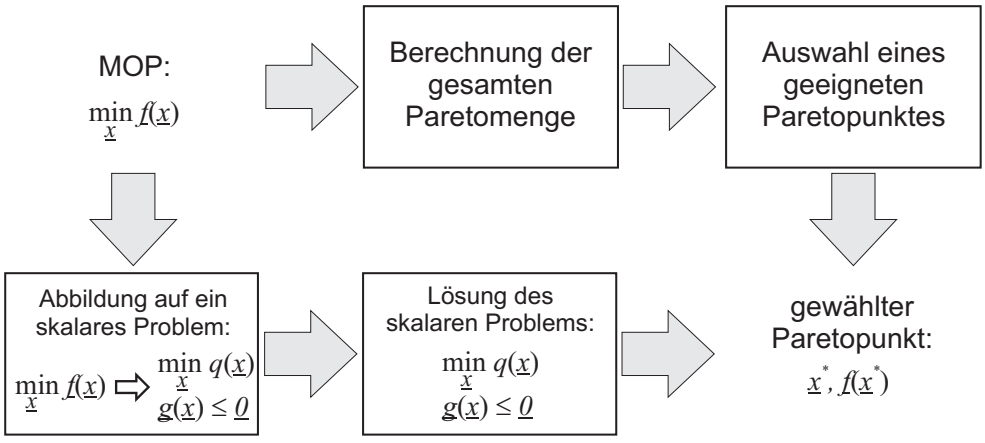


Abbildung 3.3: Herangehensweisen bei der Mehrzieloptimierung

wird bei diesen Ansätzen über bestimmte Skalierungs- oder Gewichtungsvariablen festgelegt, welcher Punkt der Paretomenge anvisiert wird. Die Entscheidung der Paretopunktauswahl findet also bereits vor der eigentlichen Optimierung statt. Der Rechenaufwand bei dieser Herangehensweise ist vergleichsweise gering, da lediglich einzelne Punkte der Paretomenge berechnet werden. Zudem können schnelle, ausgereifte und gut verfügbare Verfahren der Eingrößenoptimierung eingesetzt werden. Allerdings gelingt es oft nicht auf Anhieb einen „günstigen“ Paretopunkt zu berechnen. In diesen Fällen sind mehrere Optimierungsläufe notwendig. Eine weitere Problematik bei bestimmten nichtkonvexen Zielfunktionen ist, dass je nach verwendeter Methode nicht alle Punkte der Paretomenge erreicht werden können bzw. dass möglicherweise auch nicht-paretooptimale Punkte berechnet werden.

Berechnung der gesamten Paretomenge: Ein umfassender Ansatz zur Berechnung des gewünschten Paretopunktes besteht darin, zunächst die komplette Paretomenge des MOP zu ermitteln. Die Auswahl eines geeigneten Entwurfspunktes erfolgt anschließend auf Basis dieser Menge. Diese Entwurfspunktselektion[10] stellt ein eigenes Teilproblem dar und wird oft dem Entwickler überlassen, indem ihm die gesamte Paretomenge als Lösung zur Verfügung gestellt wird. In den meisten Fällen ist diese Vorgehensweise sinnvoll, da der Entwickler so sein Expertenwissen zur Bewertung der Paretopunkte einbringen und aus der gesamten Menge der möglichen Lösungen einen im Ingenieurverständnis „optimalen“ Punkt auswählen kann.

3.4 Verfahren zur Berechnung einzelner Paretopunkte

Im Folgenden werden einige Techniken vorgestellt, die auf der Abbildung des MOP auf eine skalare Zielfunktion beruhen. Es werden hier insbesondere Formulierungen des Mehrzieloptimierungsproblems betrachtet, die mit Hilfe der oben beschriebenen Algorithmen zur beschränkten und unbeschränkten nichtlinearen Optimierung gelöst werden können. Eine Anwendung dieser Verfahren kann daher mit Hilfe von weit verbreiteten Standardprogrammen erfolgen. Hierbei ist jedoch zu beachten, dass Einschränkungen, die für die eingesetzten skalaren Optimierungsmethoden gelten, grundsätzlich auch bei MOP zum Tragen kommen. Die nachfolgend beschriebenen Verfahren werden u. a. von Hillermeier [Hil01] und Deppe [Dep06] beschrieben und gegenübergestellt.

3.4.1 Gewichtete Summe

Das Bilden einer positiv gewichteten Summe zählt zu den Standardvorgehensweisen, um ein MOP auf eine skalare Zielfunktion zu reduzieren. Bei diesem Verfahren wird jeder einzelnen Zielfunktion f_i ein Gewichtungsfaktor $\alpha_i \geq 0$ zugewiesen. Anschließend wird das folgende Optimierungsproblem gelöst:

$$\min_{\underline{x} \in \mathbb{R}^n} \sum_{i=1}^{m} \alpha_i \, f_i(\underline{x}) \tag{3.30}$$

Es kann leicht gezeigt werden, dass die Lösung dieses Problems zwangsläufig einen paretooptimalen Punkt ergibt. Die Gewichtungsfaktoren werden dazu genutzt die Relevanz der

[10] engl. *decision making*

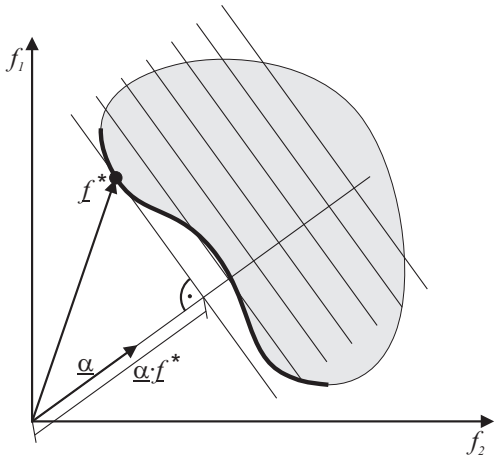


Abbildung 3.4: Gewichtete Summe

einzelnen Zielfunktionen anzugeben. Durch Variation des Gewichtungsvektors $\underline{\alpha}$ können unterschiedliche Punkte der Paretomenge berechnet werden.

In Abbildung 3.4 ist eine geometrische Interpretation des Verfahrens dargestellt. Der Gewichtungsvektor $\underline{\alpha}$ steht senkrecht auf den eingezeichneten Hyperebenen. Formuliert man die gewichtete Summe aus (3.30) als das Skalarprodukt von Gewichtungsvektor und Zielfunktionsvektor: $\sum_{i=1}^{m} \alpha_i f_i(\underline{x}) = \underline{\alpha}^{\mathrm{T}} \cdot \underline{f}(\underline{x})$, so wird deutlich, dass entlang einer Hyperebene der Wert der gewichteten Summe konstant ist. Die Hyperebenen können daher als „Höhenlinien“ der gewichteten Summe interpretiert werden. Bei der Optimierung von (3.30) wird versucht, eine Höhenlinie mit einem möglichst niedrigen Skalarproduktwert zu erreichen. Im dargestellten Fall bildet die gesuchte Hyperebene eine Tangente zu der Paretomenge. Die Lösung liegt daher im Berührungspunkt der Hyperebene und der Paretomenge.

Das Verfahren der gewichteten Summe besitzt einige gravierende Nachteile:

- Bei allgemeinen Mehrzieloptimierungsproblemen mit nichtkonvexen Zielfunktionen können Punkte der Paretomenge existieren, die mit Hilfe positiv gewichteter Summen nicht berechnet werden können (siehe [DD96a] und [Hil01]). In Abbildung 3.4 können bspw. die Punkte im mittleren Bereich der Paretofront nicht über eine gewichtete Summe bestimmt werden.

- Das scheinbar sehr intuitive Festlegen der Gewichtungsfaktoren führt in vielen Fällen nicht zu dem gewünschten bzw. erwarteten Paretopunkt. Dies liegt daran, dass mit $\underline{\alpha}$ lediglich die Steigung der Höhenlinien festgelegt wird. Über die Lage des Optimums kann jedoch ohne Kenntnis der Form der Paretomenge keinerlei Aussage gemacht werden. Speziell in Abschnitten der Paretomenge, die im Bildraum nur eine geringe Krümmung aufweisen, reagiert das Verfahren sehr empfindlich gegenüber Änderungen von $\underline{\alpha}$. Dies erschwert eine gezielte Auswahl von Paretopunkten in diesen Abschnitten.

3.4.2 Gewichtungsverfahren mit L_p-Metrik

Das Gewichtungsverfahren mit L_p-Metrik arbeitet mit einer gewichteten Vektornorm und einem unerreichbaren, utopischen Zielpunkt $\underline{P}$ im Bildraum. Gesucht wird der Paretopunkt

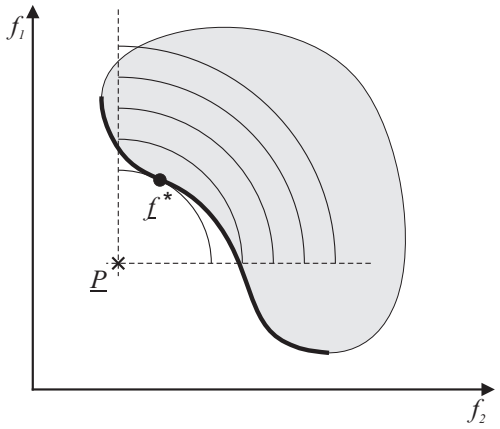


Abbildung 3.5: Gewichtungsverfahren mit Lp-Metrik

$\underline{f}^\star$, der den kleinsten Abstand zum Zielpunkt $\underline{P}$ aufweist. Der Abstand wird mit Hilfe einer gewichteten Metrik q_p der Form

$$q_p = \left(\sum_{i=1}^{m} w_i |f_i - P_i|^p \right)^{\frac{1}{p}} \tag{3.31}$$

mit $p > 1$, $w_i > 0$ und $\sum_{i=1}^{m} w_i = 1$ berechnet. Hiermit ergibt sich dann das skalare Minimierungsproblem:

$$\min_{\underline{x}} \sum_{i=1}^{m} w_i \, |f_i(\underline{x}) - P_i|^p \tag{3.32}$$

Das Minimierungsproblem kann mit Verfahren der unbeschränkten nichtlinearen Optimierung gelöst werden. Zur Berechnung verschiedener Paretopunkte können sowohl der Zielpunkt $\underline{P}$, als auch die Metrik (3.31), welche den Abstand zwischen einer Lösung $\underline{f}^\star$ und dem Zielpunkt beschreibt, variiert werden.

Die skalare Transformation dieses Gewichtungsverfahrens wird in Abbildung 3.5 veranschaulicht. Die Konfigurationsparameter der Metrik wurden hier auf $\underline{w} = (1,1)^{\mathrm{T}}$ und $p = 2$ gesetzt. Die Kreisausschnitte markieren Punkte mit konstanten Werten der skalaren Zielfunktion. Prinzipiell können mit Hilfe dieses Verfahrens alle Punkte der Paretomenge durch entsprechende Variation von $\underline{P}$, $\underline{w}$ und p berechnet werden. Leider können keine generellen Aussagen zur Variation dieser Parameter zur Berechnung verschiedener Paretopunkte getroffen werden [GN90].

3.4.3 ε-constraint Methode

Die Grundidee dieses Verfahrens besteht in der Auswahl einer einzelnen Zielfunktion f_i, $i \in \{1, \ldots, m\}$ die es zu minimieren gilt. Für alle anderen Zielfunktionen wird eine feste obere Schranke ε_j mit $j = 1, \ldots, m$ und $j \neq i$ vorgegeben, die nicht überschritten werden darf. Hierdurch ergibt sich das folgende skalare Optimierungsproblem:

$$\begin{aligned} &\min_{\underline{x} \in \mathbb{R}^n} f_i(\underline{x}), \quad \text{für ein } i \in \{1, \ldots, m\} \\ &f_j \leq \varepsilon_j, \qquad \text{mit} \quad j = 1, \ldots, m \quad \text{und} \quad j \neq i \end{aligned} \tag{3.33}$$

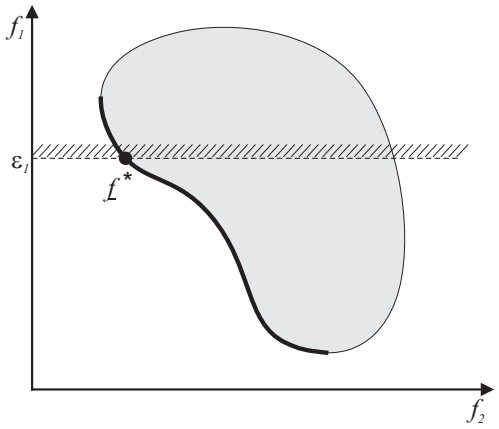


Abbildung 3.6: ϵ-Constraint Methode

Die optimale Lösung von (3.33) stellt ebenfalls eine Lösung des ursprünglichen MOP dar. Die Lösung des Problems kann mit Verfahren der beschränkten nichtlinearen Optimierung, wie zum Beispiel einem SQP-Verfahren, erfolgen.

Die Abbildung 3.6 veranschaulicht dieses Verfahren im Bildraum. Der Index i der gewählten Zielfunktion sowie die Schranken ε_j der übrigen Zielfunktionen stellen die Konfigurationsparameter dieses Verfahrens dar. Durch Variation der Schranken lassen sich im Prinzip alle paretooptimalen Punkte berechnen. Die Hauptschwierigkeit besteht allerdings darin, einen sinnvollen Wertebereich für diese Schranken zu finden. Werden die Schranken zu niedrig gewählt, so können nicht alle Schranken eingehalten werden. Das Optimierungsproblem besitzt in diesem Fall keine gültige Lösung. Wählt man eine Schranke zu hoch, so spielt die zugehörige Zielfunktion bei der Optimierung keine Rolle.

3.4.4 Goal-Attainment-Methode

Die nachfolgend beschriebene Goal-Attainment-Methode wurde von Gembicki in [Gem74] vorgestellt. Bei dieser Methode wird der Vorstellung des Entwicklers über den zu optimierenden Paretopunkt besonders Rechnung getragen. Die Parametrierung des Verfahrens geschieht auf eine intuitive Art und Weise, die in den meisten Fällen auch zu dem gewünschten Ergebnis führt. Aus diesem Grund besitzt diese Methode eine relativ große Popularität.[11]

Die Parametrierung geschieht über die Definition eines gewünschten Zielpunktes $\underline{P}$ sowie eines Gewichtungsvektors $\underline{w}$. Der Zielpunkt kann dabei beliebig vorgegeben werden, d. h. er kann erreichbar, aber auch unerreichbar sein. Er beinhaltet die Vorstellung des Entwicklers über die gewünschten oder idealen Zielfunktionswerte. Mit Hilfe des Gewichtungsvektors kann nun quantifiziert werden, „wie gut“ der Zielpunkt $\underline{P}$ während der Optimierung erreicht wurde. Für jede Zielgröße wird ein Erfüllungsgrad angegeben, der durch den linearen Zusammenhang $(f_i(\underline{x}) - P_i)/w_i$ definiert ist.

Die Idee dieser Methode besteht nun im Ausbalancieren dieser Erfüllungsgrade. Dies bedeutet, dass die Zielgrößen mit den jeweils größten Erfüllungsgraden sukzessive soweit

[11] Es existieren zahlreiche Implementierungen, die nach dem Prinzip der Goal-Attainment-Methode arbeiten. Als Beispiele seien hier das im Quellcode erhältliche FSQP [LZT97], das an der Universität Paderborn entwickelte MOPO (Multi-Objective Parameter Optimization) [Mün03, Dep06], das am DLR entwickelte Optimierungswerkzeug MOPS (Multi-Objective Parameter Synthesis) [JBL+02, Joo03] sowie die Routine `fgoalattain` aus der Matlab Optimzation Toolbox [Mat07b] genannt.

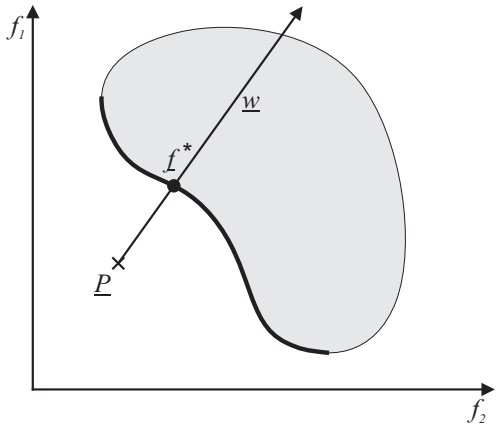


Abbildung 3.7: Goal-attainment Methode

minimiert werden, dass die Erfüllungsgrade gleiche Werte annehmen. Es gilt also ein sogenanntes Min-Max-Problem zu lösen:

$$\min_{\underline{x}} \max_{i} \frac{f_i(\underline{x}) - P_i}{w_i}, \quad i = 1, \ldots, m \tag{3.34}$$

Die Lösung von (3.34) ergibt immer einen paretooptimalen Punkt. Durch entsprechende Variation von $\underline{w}$ und $\underline{P}$ können mit der Goal-Attainment Methode alle Punkte der Paretomenge berechnet werden.

In Abbildung 3.7 ist eine grafische Interpretation der Goal-Attainment-Methode dargestellt. Das Ziel ist die Berechnung des Punktes $\underline{f}^\star$, bei dem der Vektor $\underline{w}$ die Paretomenge durchstößt.

Das Min-Max-Problem (3.34) stellt ein unbeschränktes Optimierungsproblem dar, welches wegen der Maximumsfunktion nicht überall differenziert werden kann. Es lässt sich daher nicht mit den im Abschnitt 3.1 beschriebenen Verfahren direkt behandeln. Die Lösung des Min-Max-Problems kann aber mit Hilfe einer exponentiellen Approximation oder mit Hilfe einer äquivalenten Umformung erfolgen.

Exponentielle Approximation der Maximumsfunktion

Nach Kreisselmeier und Steinhauser [KS79] kann die Maximumsfunktion aus (3.34) mit Hilfe der Exponentialfunktion:

$$q_{exp}(\underline{x}) = \frac{1}{\rho} \ln \sum_{i=1}^{m} e^{\rho \frac{f_i(\underline{x}) - P_i}{w_i}} \tag{3.35}$$

angenähert werden. Für $\rho \to \infty$ konvergiert die Funktion q_{exp} gegen die Maximumsfunktion. Mit dieser Approximation ergibt sich das Optimierungsproblem:

$$\min_{\underline{x}} q_{exp}(\underline{x}) \tag{3.36}$$

Da es sich bei (3.35) um eine differenzierbare Funktion handelt, kann das resultierende Optimierungsproblem mit einfachen Verfahren der unbeschränkten nichtlinearen Optimierung, z. B. mit einem Quasi-Newton-Verfahren, gelöst werden. Der Nachteil dieser Methode ist, dass die Genauigkeit der Lösung durch die Approximation beeinträchtigt wird.

Umformung des Min-Max-Problems

Das Min-Max-Problem kann durch das Einführen einer neuen Optimierungsvariablen in ein äquivalentes beschränktes nichtlineares Optimierungsproblem überführt werden. Der Parametervektor $\underline{x}$ wird hierzu um die skalare Variable γ erweitert:

$$\underline{\hat{x}} = \begin{pmatrix} \underline{x} \\ \gamma \end{pmatrix} \tag{3.37}$$

Es wird nun gefordert, dass diese neue Variable größer gleich dem maximalen Erfüllungsgrad der Zielfunktionen ist: $\gamma^\star \geq \max_i \{(f_i - P_i)/w_i\}$. Durch Minimierung von γ ergibt sich das beschränkte Optimierungsproblem

$$\begin{aligned} &\min_{\underline{\hat{x}}} \gamma \\ &f_i(\underline{x}) - P_i - w_i\,\gamma \leq 0\,, \quad i = 1, \ldots, m \end{aligned} \tag{3.38}$$

Durch die Nebenbedingungen wird dafür gesorgt, dass in der Lösung $\underline{\hat{x}}^\star$ die Erfüllungsgrade der Zielgrößen auch tatsächlich kleiner gleich dem maximalen Erfüllungsgrad γ sind: $(f_i(\underline{x}) - P_i)/w_i \leq \gamma$. Wählt man $w_i = 0$ so wird für die Zielgröße f_i eine obere Schranke P_i definiert, die es exakt einzuhalten gilt.

Das Optimierungsproblem (3.38) kann mittels eines SQP-Verfahrens effizient gelöst werden. Wegen der speziellen Struktur des Problems bietet es sich jedoch an, eine modifizierte Merit-Funktion $\phi(\underline{\hat{x}})$ für die Liniensuche zu verwenden [Nau02, Mat07b]:

$$\phi(\underline{\hat{x}}) = \sum_{i=1}^{m} \begin{cases} r_i \max\{0\,, f_i(\underline{x}) - P_i - w_i\,\gamma\} & \text{wenn } w_i = 0, \\ \max_j \{(f_j(\underline{x}) - P_j)/w_j\}\,,\ j = 1, \ldots, m & \text{andernfalls.} \end{cases} \tag{3.39}$$

Die Bestrafungsterme r_i werden wie in Gleichung (3.26) bestimmt.

Anmerkungen zur Methode

Bei Zielfunktionen, die beispielsweise durch Fehlerflächen formuliert wurden, liegt das „ideale“ Minimum der Zielfunktionen bei Null. Aus diesem Grund kann in vielen praktischen Anwendungen der Zielpunkt $\underline{P}$ in den Ursprung gelegt werden. Der Gewichtungsvektor gibt in diesem Fall die Relation der Zielgrößen untereinander an. Im Minimum stehen die „größten“ konkurrierenden Zielgrößen im Verhältnis

$$\frac{f_i(\underline{x}^\star)}{f_j(\underline{x}^\star)} = \frac{w_i}{w_j}$$

zueinander. Durch den Gewichtungsvektor können also die tatsächlichen Verhältnisse der miteinander konkurrierenden Zielgrößen im Lösungspunkt $\underline{f}(\underline{x}^\star)$ festgelegt werden.

Dies ist besonders nützlich, wenn in einem MOP beispielsweise Zielfunktionen für die Funktionserfüllung, also den „Nutzen“ eines technischen Systems und weitere Zielfunktionen für den hierfür notwendigen „Aufwand“ definiert werden. Mit dem Gewichtungsvektor $\underline{w}$ kann dann das für den Entwickler sehr aussagekräftige Verhältnis von: *Nutzen/Aufwand* angegeben werden.

Ein unerfahrener Anwender könnte bei Anwendung des Verfahrens der gewichteten Summe die Vorstellung besitzen, dass hier mit den Gewichtungsfaktoren die Größenverhältnisse

zwischen Zielgrößen auf anschauliche Weise justiert werden können. Tatsächlich wird diese Vorstellung aber nicht von dem Verfahren der gewichteten Summe, sondern von der Goal-Attainment-Methode umgesetzt. Der Entwickler kann bei dieser Methode mit Hilfe des Vektors $\underline{w}$ solche Zielgrößenrelationen vorgeben und kommt so sehr intuitiv zu einer Paretopunktauswahl.[12]

3.5 Verfahren zur Berechnung der gesamten Paretomenge

Zur Berechnung kompletter Paretomengen existieren spezielle Methoden, die von vornherein darauf abzielen, mehrere Punkte der Paretomenge gleichzeitig zu ermitteln. Die Entwicklung solcher Verfahren ist derzeit ein aktives Forschungsgebiet. Eine hohe Popularität besitzen hierbei genetische und evolutionäre Ansätze [FF93, FF98, Deb01, FPL05, Mat07a]. Des Weiteren werden auch biologisch inspirierte Ansätze wie Multi-Objective Particle Swarm Optimization (MOPSO) untersucht und erfolgreich zur Mehrzieloptimierung eingesetzt [CL02, Mos04]. Neben diesen heuristischen Ansätzen existieren auch Verfahren, die auf Basis von mengenorientierten Ansätzen und Unterteilungstechniken eine Approximation der kompletten Paretomenge liefern. Das an der Universität Paderborn entwickelte Optimierungswerkzeug GAIO[13] arbeitet beispielsweise nach dieser Methode [Sch04]. Auf eine genaue Beschreibung dieser verschiedenen Verfahren wird jedoch an dieser Stelle verzichtet.

Im Folgenden werden einige deterministische Methoden vorgestellt, die eine Approximation der gesamten Paretomenge durch die wiederholte Optimierung mit einer skalaren Abbildung erreichen. Diese skalare Abbildung muss über einen Vektor $\underline{\beta}$ parametrierbar sein, über den die einzelnen Punkte der Paretomenge adressiert werden können. Durch Variation von $\underline{\beta}$ vor jedem Optimierungslauf können verschiedene Punkte der Paretomenge berechnet werden. Dieses Berechnungsschema ist in Abbildung 3.8 dargestellt.

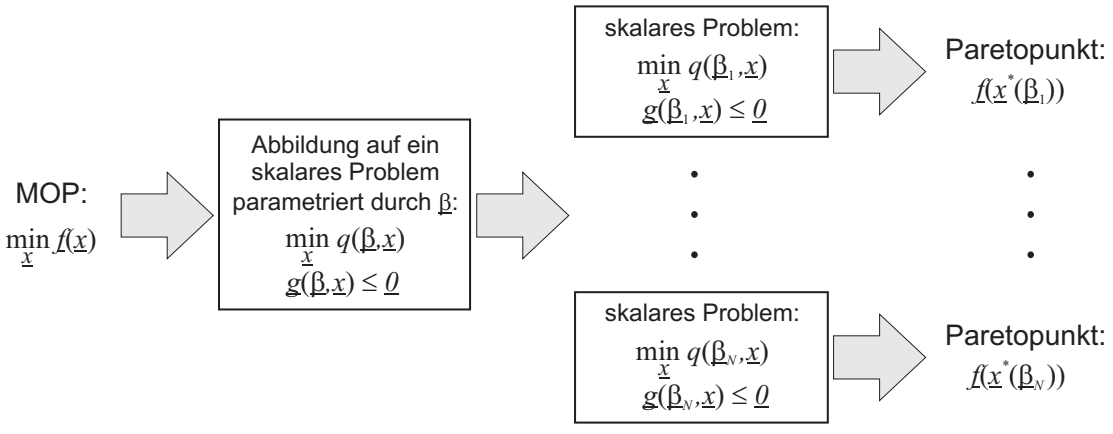


Abbildung 3.8: Approximierung der Paretomenge mit Hilfe einer Abbildung auf skalare Zielfunktionen

[12] Ähnliche Aussagen zu der Goal-Attainment Methode werden von Foellinger [Föl94] und Naumann [Nau02], aber auch in [Mat07b] getroffen.

[13] Global Analysis of Invariant Objects

Bei der Approximation der gesamten Paretomenge ist es sinnvoll, eine möglichst gleichmäßige Verteilung der Punkte im Bildraum zu erreichen. Grundsätzlich sind hierbei die im Abschnitt 3.4 vorgestellten Lösungsverfahren zur Berechnung der einzelnen Paretopunkte anwendbar. Von diesen Verfahren sind das Verfahren der Gewichteten Summe und Gewichtungsverfahren mit L_p-Metrik ohne zusätzliche Maßnahmen, z. B. Adaptionsstrategien, eher ungeeignet. Bei der Gewichteten Summe hat eine kleine Änderung der Gewichtungen oft große Auswirkungen auf das Ergebnis, so dass es schwer fällt, eine gleichmäßige Verteilung der Punkte zu erreichen. Es kann leicht gezeigt werden, dass bei einer gleichförmigen Variation der Gewichtungen in stark gekrümmten Bereichen der Paretomenge gehäuft Punkte berechnet werden, wohingegen in schwach gekrümmten Bereichen nur vereinzelt Punkte auftreten (siehe Abbildung 3.12 auf Seite 37).

3.5.1 Normal-Boundary-Intersection

Die Methode Normal-Boundary-Intersection (NBI) wurde von Das und Dennis [DD96b] entwickelt und benutzt einen geometrisch motivierten Ansatz im Bildraum, um gleichmäßig über die Paretomenge verteilte Punkte zu generieren.

In einem ersten Schritt werden beim NBI-Ansatz die Extremwerte der Paretomenge ermittelt. Durch Optimierung der individuellen Zielfunktionen $f_i(\underline{x})$ mit $i = 1, \ldots, m$ erhält man $\underline{x}_i^\star$. Die Minima der individuellen Zielfunktionen bilden im Bildraum einen Simplex[14], der durch die Matrix

$$\underline{\underline{\Theta}} = \left(\underline{f}(\underline{x}_1^\star), \ldots, \underline{f}(\underline{x}_m^\star)\right), \quad \underline{\underline{\Theta}} \in \mathbb{R}^{m \times m} \tag{3.40}$$

repräsentiert wird. Mit Hilfe eines Vektor $\underline{\beta}$ kann ein beliebiger Punkt des Simplex durch: $\underline{\underline{\Theta}}\,\underline{\beta}$ mit $\underline{\beta} \in \mathbb{R}^m$, $\beta_i \geq 0$ und $\sum_{i=1}^m \beta_i = 1$ beschrieben werden. Abbildung 3.9 zeigt einen solchen Simplex bei einem Optimierungsproblem mit zwei Zielgrößen.

Beim NBI-Ansatz wird nun an einem Punkt des Simplex $\underline{\underline{\Theta}}\,\underline{\beta}$ ein Normalenvektor $\underline{N}$ angelegt. Dieser Vektor zeigt in Abstiegsrichtung und durchstößt unter normalen Voraussetzungen den Rand des erreichbaren oder zulässigen Raums in einem Paretopunkt. In Abbildung 3.9 ist dieser Punkt mit $\underline{f}^\star$ gekennzeichnet.

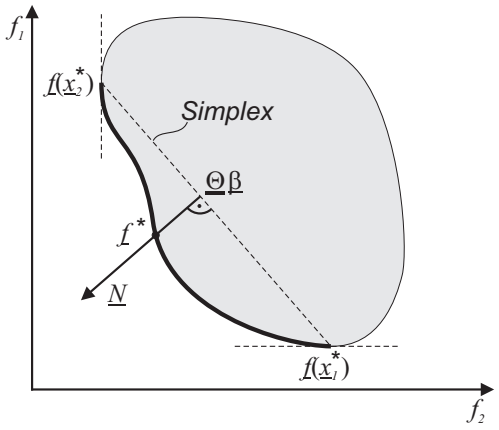


Abbildung 3.9: Normal-Boundary-Intersection-Methode zur Bestimmung der Paretofront

[14] Ein Simplex ist der einfachste Polytop in einem n-dimensionalen Raum. Ein n-dimensionales Simplex weist dabei $n + 1$ Ecken auf. Mit steigender Dimension ergibt sich hierbei die Reihe: Punkt, Strecke, Dreieck, Tetraeder usw. Ein n-Simplex ist die Fortsetzung dieser Reihe auf n Dimensionen.

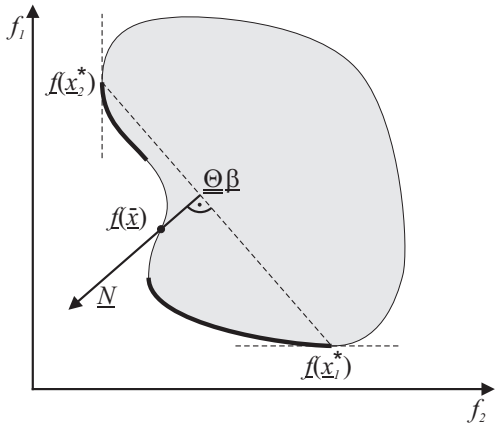


Abbildung 3.10: Normal-Boundary-Intersection: Ein Fall, bei dem die Lösung $\underline{f}(\bar{\underline{x}})$ des NBI-Problems nicht paretooptimal ist.

Dieser Punkt $\underline{f}^\star$ kann mit Hilfe des beschränkten Optimierungsproblems:

$$\begin{aligned} &\min_{\substack{\underline{x}\in\mathbb{R}^n \\ \gamma\in\mathbb{R}}} -\gamma \\ &\underline{\underline{\Theta}}\,\underline{\beta} + \gamma\,\underline{N} - \underline{f}(\underline{x}) = 0 \end{aligned} \tag{3.41}$$

berechnet werden. Die Größe γ beschreibt den Abstand eines Punktes $\underline{f}$ vom Simplex $\underline{\underline{\Theta}}\,\underline{\beta}$. Die Maximierung von γ liefert man den gesuchten Paretopunkt $\underline{f}^\star$.

Die Berechnung der Paretofront geschieht in einem übergeordneten Prozess. Durch Variation des Vektors $\underline{\beta}$ wird der Normalenvektor $\underline{N}$ auf dem Simplex verschoben und durch Lösung der resultierenden NBI-Probleme (3.41) erhält man vergleichsweise homogen über die Paretofront verteilte Punkte.

Bei MOP mit zwei Zielgrößen kann prinzipiell jeder Punkt der Paretomenge mit Hilfe des NBI-Ansatzes berechnet werden. Dies gilt jedoch nicht für MOP mit mehr als zwei Zielfunktionen. Im Falle $m \geq 3$ liegen möglicherweise die Randgebiete der Paretofront nicht mehr „senkrecht unterhalb“ des Simplex und können daher nicht erreicht werden.

Ein weiterer Nachteil besteht darin, dass bei nichtkonvexen Zielfunktionen die Lösung des NBI-Problems (3.41) nicht zwingend paretooptimal ist (siehe Abbildung 3.10). Dieser Nachteil kann durch Umformulierung von (3.41) in ein Min-Max-Problem umgangen werden. Die Lösung ist dann stets paretooptimal und es ergibt sich:

$$\min_{\underline{x}} \max_{i} \frac{f_i(\underline{x}) - \left(\underline{\underline{\Theta}}\,\underline{\beta}\right)_i}{-N_i}, \quad i = 1, \ldots, m \tag{3.42}$$

Durchstößt der Normalenvektor $\underline{N}$ allerdings nicht die Paretofront, so liefert die Lösung von (3.42) einen Punkt auf dem Rand der Paretofront (siehe [Mün03, Dep06]). Unterschiedliche Werte für $\underline{\beta}$ können in diesem Fall zu identischen Paretopunkten führen.

Zur Lösung des Min-Max-Problems (3.42) kann die Goal-Attainment-Methode mit $\underline{P}(\underline{\beta}) = \underline{\underline{\Theta}}\,\underline{\beta}$ und $\underline{w} = -\underline{N}$ herangezogen werden (vgl. mit Gl. (3.34)). Wegen der garantierten Paretooptimalität, der schnellen Konvergenz und der guten Verfügbarkeit von Implementierungen ist diese Methode bei glatten, differenzierbaren Zielfunktionen den bislang vorgestellten Verfahren vorzuziehen. Aufgrund der hohen Flexibilität in der Formulierung und Parametrierung lassen sich mit Hilfe der Goal-Attainment-Methode weitere

Lösungsverfahren zur Berechnung der gesamten Paretomenge realisieren, die ähnlich wie die NBI-Methode geometrisch motiviert sind und im Bildraum arbeiten. Zwei alternative Ausprägungen werden im Folgenden vorgestellt.

3.5.2 Goal-Attainment-Methode mit $\underline{P}$ im Ursprung

Diese Methode knüpft an die Vorstellung an, dass bei der Goal-Attainment-Methode mit dem Vektor $\underline{w}$ die Relation zwischen den Zielgrößen angegeben wird (siehe Abschnitt 3.4.4). Anstelle den Punkt $\underline{P}$ in Abhängigkeit von β zu verändern, wird hier der Vektor $\underline{w}$ variiert (vgl. mit Gl. (3.34)). Der Punkt $\underline{P}$ wird in den Ursprung gelegt. Diese Methode eignet sich daher nur für MOP, bei denen Zielfunktionswerte größer gleich 0 garantiert werden können, wie z. B. bei Fehlerflächen. Eine geometrische Interpretation ist links in Abbildung 3.11 zu sehen.

In einem ersten Schritt wird analog zur NBI-Methode eine Optimierung der individuellen Zielgrößen $f_i(\underline{x})$ durchgeführt und der zugehörige Simplex mit der Matrix $\underline{\boldsymbol{\Theta}} = \left(\underline{f}(\underline{x}_1^\star), \ldots, \underline{f}(\underline{x}_m^\star)\right)$ gebildet.

Die Berechnung der einzelnen Paretopunkte erfolgt durch Verschieben des Punktes $\underline{\boldsymbol{\Theta}}\,\underline{\beta}$ und Lösen eines Min-Max-Problems. Mit $\underline{P} = \underline{0}$ und $\underline{w}(\underline{\beta}) = \underline{\boldsymbol{\Theta}}\,\underline{\beta}$ ergibt sich das Optimierungsproblem:

$$\min_{\underline{x}} \max_{i} \frac{f_i(\underline{x})}{\left(\underline{\boldsymbol{\Theta}}\,\underline{\beta}\right)_i}, \quad i = 1, \ldots, m \tag{3.43}$$

Die Verteilung der Punkte auf der Paretofront ist bei dieser Methode vergleichsweise homogen. Jedoch wird die Paretomenge speziell im mittleren Bereich feiner approximiert, als in der Nähe der Extremwerte (Abbildung 3.12). Gegenüber der NBI-Methode ist daher kein Vorteil bezüglich der Verteilung vorhanden. Diese Methode kann jedoch dann von Vorteil sein, wenn die Relation der Zielgrößen in der Anwendung eine besondere Rolle spielt, da eine direkte Zuordnung zwischen der Relation und dem Vektor $\underline{\beta}$ möglich ist. Aufgrund dieser Eigenschaft wird auf den geometrischen Ansatz dieser Methode später in den Kapiteln 9.1.4 und 9.2.1 zurückgegriffen.

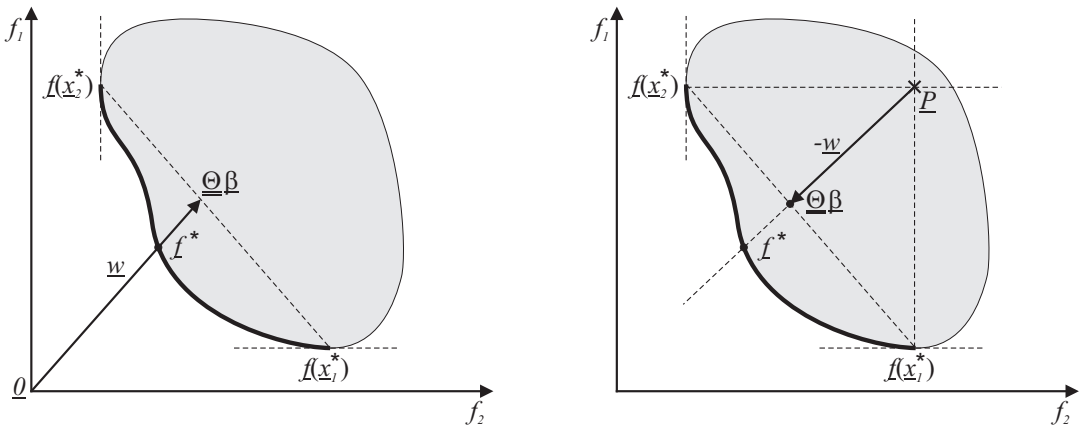


Abbildung 3.11: Zwei Ansätze zur Berechnung der Paretofront mit der Goal-Attainment-Methode

3.5.3 Goal-Attainment-Methode mit $\underline{P}$ im Maximum

Bei diesem Ansatz handelt es sich um eine Abänderung der vorangegangenen Methode, die eine höhere Auflösung in der Nähe der Extremwerte der Paretofront erreicht. Anstelle den Punkt $\underline{P}$ in den Ursprung zu legen, wird $\underline{P}$ in den Punkt gelegt, der aus den maximalen Zielfunktionswerten von $\underline{\mathbf{\Theta}}$ gebildet wird. Für jeden Punkt der Paretomenge muss dann das Min-Max-Problem (3.34) mit

$$\begin{aligned} P_i &= \max_j \underline{\mathbf{\Theta}}_{i,j}\,, \quad i = 1, \ldots, m \\ \underline{w}(\underline{\beta}) &= \underline{P} - \underline{\mathbf{\Theta}}\,\underline{\beta} \end{aligned} \tag{3.44}$$

gelöst werden. Dieser Ansatz ist in Abbildung 3.11 rechts dargestellt.

Im Diagramm 3.12 sind die Verteilungen der Punkte auf der Paretomenge bei Anwendung der verschiedenen Methoden zu sehen. Zum Vergleich wurde ebenfalls die Verteilung mit Hilfe einer gewichteten Summe dargestellt. Diese weist eine große Punktedichte im linken oberen Bereich der Paretomenge auf, wohingegen nur wenige Punkte im rechten unteren Bereich berechnet werden.

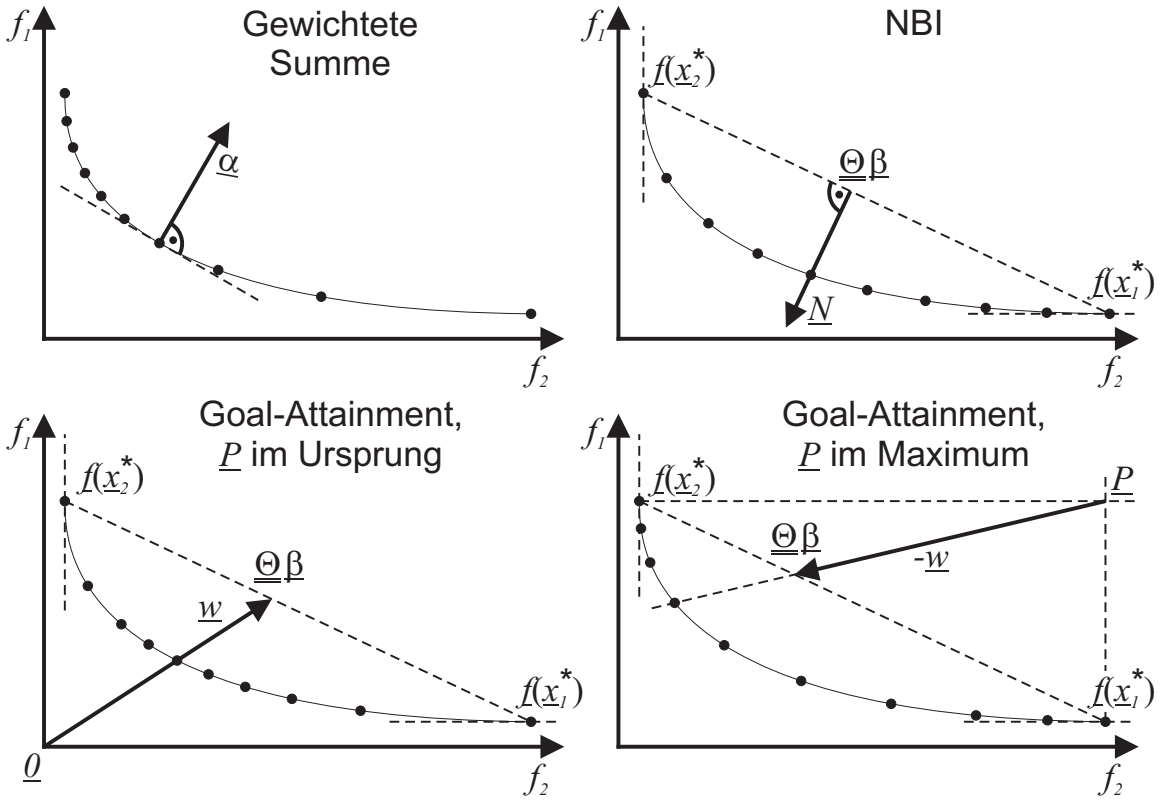


Abbildung 3.12: Verteilung der Punkte auf der Paretofront im Bildraum eines zweidimensionalen MOP bei Anwendung der verschiedenen Berechnungsverfahren. Zur Berechnung der Paretopunkte wurde $\underline{\mathbf{\Theta}}\,\underline{\beta}$ in äquidistanten Schritten von $\underline{f}(\underline{x}_1^\star)$ nach $\underline{f}(\underline{x}_2^\star)$ verschoben.

4 Optimierung mechatronischer Systeme

Selbstoptimierende mechatronische Funktionsmodule beobachten im Betrieb laufend sich selbst und ihr Umfeld, um ihr Verhalten selbstständig an sich ändernde Betriebsbedingungen anzupassen. Hierzu müssen sie in der Lage sein, eine auf die jeweils aktuelle Situation hin optimierte Systemkonfiguration bereitzustellen. Selbstoptimierende Systeme besitzen daher naturgemäß einen starken Bezug zu Optimierungsmethoden. In diesem Kapitel wird betrachtet, wie mathematische Optimierungsverfahren zur Ermittlung der optimalen Systemkonfigurationen eingesetzt werden können.

Der Einsatz von Optimierungsverfahren zur Auslegung von Reglerparametern hat eine lange Tradition. Viele klassische Methoden zum Entwurf optimaler Regelungen lassen sich im Prinzip auf Minimierungsprobleme und somit auf Optimierungsprobleme zurückführen. Prominente Vertreter solcher Entwurfsmethoden sind das H_2 und H_∞ Verfahren zur Auslegung optimaler Mehrgrößenregelungen [Rai94, SP96, Unb00, Lun06]. Diese Auslegungsverfahren setzen bewusst eine feste Struktur der Systembeschreibung und des Entwurfsproblems voraus. Die resultierenden Optimierungsprobleme werden durch den Einsatz spezieller numerischer Methoden effizient gelöst.

Um die strukturellen Einschränkungen der klassischen Methoden zu umgehen, kommen in den letzten Jahren in Theorie und Praxis verstärkt allgemeine Optimierungsverfahren zur Auslegung der Reglerparameter aber auch der Reglerstruktur zur Anwendung. Mit ihrer Hilfe können grundsätzlich auch Systeme mit mehr oder weniger stark ausgeprägten nichtlinearen Effekten hinsichtlich beliebiger Entwurfskriterien ausgelegt werden. Die Vorgehensweise zur Bestimmung von Reglern unter Einsatz numerischer Optimierungsverfahren sind bereits seit etlichen Jahren in Standardwerken der Regelungstechnik wie bspw. [Föl94, DB05] zu finden. Auf der anderen Seite werden regelungstechnische Problemstellungen gerne zur Demonstration und zum Test von Optimierungsverfahren und Algorithmen genutzt. So beschreibt bspw. Papageorgiou in [Pap96] unter anderem den Einsatz von Verfahren der nichtlinearen Programmierung[1] zur Auslegung von Reglerparametern sowie zur Lösung von einfachen als auch hierarchisch strukturierten Optimalsteuerungsproblemen.

Die Optimierung technischer und speziell auch regelungstechnischer Systeme erfordert im Allgemeinen die Berücksichtigung einer Vielzahl unterschiedlicher, oft gegenläufiger Optimierungsziele. Zur Behandlung solcher Probleme existieren spezielle Verfahren zur Mehrkriterien- bzw. Mehrzieloptimierung, die ein gezieltes Abwägen zwischen den gegenläufigen Zielen ermöglichen. Föllinger beschreibt in [Föl94] eine Methode zur Auslegung von Reglern unter Beachtung mehrerer Optimierungsziele, die auf der

[1] Verfahren der *nichtlinearen Optimierung* werden vielfach nicht zur Optimierung im eigentlichen Sinne verwendet, sondern z. B. auch zum Lösen von Gleichungssystemen oder bei Identifikationsproblemen eingesetzt. Daher werden sie oft auch allgemeiner als Verfahren zur *nichtlinearen Programmierung* bezeichnet.

Gütevektoroptimierung beruht, die von Kreisselmeier und Steinhauser in [KS79] vorgestellt wurde. Ein vergleichbarer Ansatz wurde von Kasper zur Auslegung von linearen Mehrgrößenregelungen unter Berücksichtigung verschiedener Zielkriterien verfolgt [Kas85, KLJS90]. Weitere Arbeiten, die sich mit der Mehrzieloptimierung mechanischer oder mechatronischer Systeme befassen, finden sich bspw. in [Ebe96, Pir01, Nau02, Dig04, Dep06, Du06, HD09], um einige zu nennen.

Die Methodik der Mehrzieloptimierung dem Ingenieur zugänglich zu machen, ist die Zielsetzung einiger leistungsfähiger Programmpakete zur Mehrkriterienoptimierung. Solche Werkzeuge ermöglichen zum einen den Mehrzielentwurf durch eine effiziente, robuste Numerik, zum anderen unterstützen sie aber auch die Formulierung und Konfiguration des Entwurfsproblems und erleichtern durch geeignete Visualisierungstechniken die Interpretation der Optimierungsergebnisse. Softwarepakete, die speziell auf die Bedürfnisse des Mehrzielentwurfs mechatronischer Systeme hin konzipiert wurden, sind das am DLR[2] entstandene MOPS[3] [JBL+02, Joo03] und das bereits seit Anfang der 90er Jahre am MLaP der Universität Paderborn entwickelte MOPO[4] [KLJS90, Mün01, Mün03, Dep06].

Im Folgenden wird erläutert, wie ein regelungstechnisches Entwurfsproblem unter Einsatz mathematischer Optimierungsverfahren gelöst werden kann. Diese Art der Systemauslegung und Optimierung erfolgt üblicherweise modellbasiert. Anstelle die Optimierung direkt am realen System durchzuführen, wird zunächst ein mathematisches Modell der Regelstrecke optimiert. Die hierbei gefundenen Reglereinstellungen werden anschließend auf das physikalische System übertragen. Es wird hierbei die Annahme getroffen, dass die am Modell gefundenen, optimalen Reglereinstellungen am realen System ebenfalls zu einem nahezu optimalen Verhalten führen. Für eine gute Übereinstimmung muss das Modell der Regelstrecke dabei die relevanten physikalischen Effekte hinreichend gut abbilden.

4.1 Formulierung des Entwurfsproblems

Die Formulierung der Optimierungsaufgabe stellt einen iterativen Vorgang dar. Speziell bei neuen bzw. unbekannten Systemen, über deren Optimierung bislang keine Erfahrungswerte existieren, muss der Entwickler häufig mehrere Iterationsschleifen vollführen, bevor die Optimierung die gewünschten Ergebnisse liefert. Insbesondere die Gütemaße, die der Beurteilung des Systemverhaltens dienen, unterliegen im Laufe des Optimierungsprozesses häufig Änderungen oder werden um weitere Kriterien ergänzt, um dem System schrittweise das gewünschte Verhalten aufzuzwingen. Änderungen sind oft aufgrund eines nicht zufriedenstellenden „optimierten“ Verhaltens oder aufgrund unerwünschter nicht vorhersehbarer Reaktionen des Systems notwendig. Eine Anpassung innerhalb des iterativen Prozesses kann durch den Einsatz schnell modifizierbarer Simulationsmodelle in relativ kurzer Zeit vorgenommen werden. Ist die Optimierungsaufgabe einmal formuliert, kann sie oft unverändert bzw. mit geringen Modifikationen auf ähnliche Systeme angewendet werden. Eine Optimierung verschiedener Varianten eines Systems, die ja alle dieselben Entwurfsziele verfolgen und sich in Ihrem Verhalten nicht grundlegend voneinander unterscheiden, können meist mit einer unveränderten Problemformulierung durchgeführt werden. Eine

[2] Deutsches Zentrum für Luft- und Raumfahrt
[3] Multi Objective Parameter Synthesis
[4] Multi Objective Parameter Optimization

entsprechende Automatisierung vorausgesetzt, können so in kürzester Zeit mehrere Varianten ausgelegt und bezüglich ihren Anforderungen bewerten werden.

Streckenmodell Gegenstand der mathematischen Optimierung ist das Verhalten des in Abbildung 4.1 dargestellten Modells. Dieses Modell besteht aus dem Regler und einem Modell der Regelstrecke. Während das Verhalten des Reglers exakt bekannt ist, stellt das Streckenmodell eine hinreichend gute Approximation des realen Systems dar.

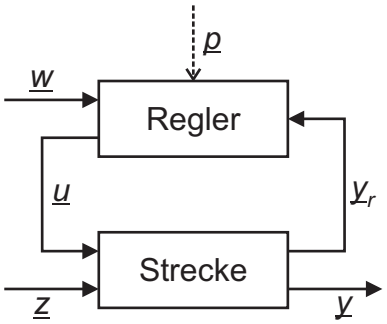


Abbildung 4.1: Blockschaltbild der Regelstrecke mit Regler

Der Vektor $\underline{p}$ fasst alle freien Entwurfsparameter, im dargestellten Fall sind dies die Reglerparameter, zusammen. Durch Variation des Vektors $\underline{p}$ können die Eigenschaften des geregelten Systems beeinflusst werden. Die Optimierungsaufgabe besteht darin, einen Parametervektor $\underline{p}^\star$ zu finden, bei dem das geregelte System ein „optimales“ Verhalten aufweist.

Optimierungsmodell Um das Regelungsverhalten zu optimieren, muss zunächst das „optimale“ Verhalten des Systems spezifiziert werden. Zu diesem Zweck wird das in Abbildung 4.1 dargestellte Modell definierten Einflüssen ausgesetzt und die hieraus resultierenden Systemantworten werden anhand von Gütekriterien bewertet.

Technische Systeme sind im Betrieb einer Vielzahl von äußeren Einflüssen ausgesetzt. Diese können abhängig vom System und dessen Einsatzgebiet bei der Optimierung berücksichtigt werden. Unter äußeren Einflüssen seien hier sowohl Anregungen, die von der Umgebung auf das System einwirken, als auch von anderen Teilsystemen vorgegebene Sollwerte oder Referenzsignale verstanden. Mit diesen äußeren Einflüssen und konkreten Anfangszuständen des Systems können unterschiedliche Betriebssituationen im Modell nachgebildet, simuliert und letztendlich optimiert werden.

Die Vorstellung des Entwicklers über das Systemverhalten wird in Bewertungskriterien abgebildet. Diese werden in Form mathematischer Funktionen ausgedrückt, um so die Güte des Systemverhaltens zu quantifizieren und einem mathematischen Optimierungsverfahren zugänglich zu machen. In einigen eher einfachen Fällen kann die Beschreibung der Systemgüte durch ein einzelnes Bewertungskriterium erfolgen. Im Allgemeinen existieren jedoch mehrere meist gegenläufige Zielsetzungen, die es gemeinsam zu berücksichtigen gilt. Aus diesem Grund wird im Folgenden davon ausgegangen, dass das System anhand mehrerer Kriterien bewertet wird. Die Entwurfsaufgabe stellt demnach ein Mehrzieloptimierungsproblem dar.

Die Formulierung des Optimierungsproblems besteht also im Wesentlichen aus der Definition der Einflüsse auf das System und entsprechender Kriterien zur Bewertung des resul-

tierenden Systemverhaltens. Dies führt nach KASPER zu einem erweiterten Streckenmodell, welches das geregelte Streckenmodell um ein *Anregungsmodell* sowie ein *Bewertungsmodell* ergänzt [Kas85].

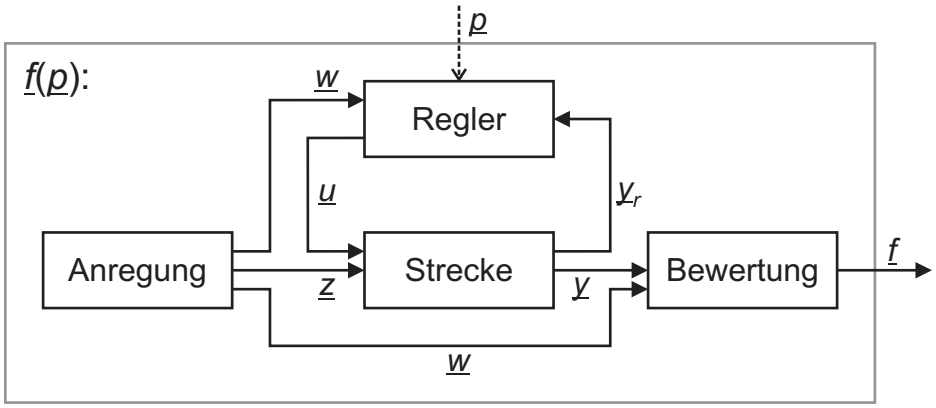


Abbildung 4.2: Blockschaltbild des Optimierungsmodells

Das **Anregungsmodell** bildet die oben genannten äußeren Einflüsse ab. Diese lassen sich im Wesentlichen in Führungs- und Störgrößen unterscheiden. Über den Referenzsignaleingang $\underline{w}$ werden Führungs- bzw. Referenzwerte für die Regelung vorgegeben. Den Referenzwerten soll das System so gut wie möglich folgen. Die Signale, die am Störeingang $\underline{z}$ anliegen, beschreiben die Anregungen bzw. Störungen, die auf physikalische Systeme wirken. Die Auswirkungen dieser Störungen sollen im Normalfall durch regelungstechnische Maßnahmen möglichst gering gehalten werden. Sowohl Referenz- als auch Störeingang müssen nicht zwingend im Modell vorhanden sein. Ist bei einer Optimierung bspw. nur das Störverhalten des Systems von Interesse, so wird lediglich am Störeingang des Streckenmodells ein entsprechendes Signal aufgeschaltet. Bei einer Optimierung des Führungsverhaltens ist dementsprechend nur der Referenzsignaleingang des Reglermodells aktiv. Die Simulation unterschiedlicher Anregungssituationen ist durch eine Variation dieser Signale möglich. Ebenso kann der Fokus der Optimierung durch unterschiedliche Ausprägung des Referenz- und des Störsignals beliebig zwischen Verbesserung des Führungs- oder des Störverhaltens verschoben werden.

Das **Bewertungsmodell** dient der Überführung der aus der Anregung resultierenden Systemantwort in konkrete Zahlenwerte, den Zielgrößenvektor $\underline{f}$, welcher die Güte des Systemverhaltens widerspiegelt. Die Bewertungskriterien sind dabei so zu wählen, dass dem System im Laufe der Optimierung das vom Entwickler gewünschte Verhalten aufgezwungen wird. Die Ankopplung des Bewertungsmodells geschieht über den Ausgangsvektor $\underline{y}$, in dem beliebige Größen des Systems zusammengefasst werden können. Die Optimierung am Modell besitzt den Vorteil, dass zur Auswertung der Bewertungskriterien auch auf Systemzustände und Ausgänge zurückgegriffen werden kann, die am realen System nicht oder nur unter hohem Aufwand messbar sind.

Der Fokus der Optimierung kann durch Variation des Bewertungs- und des Anregungsmodells gezielt beeinflusst werden. Aus diesem Grund sind Anregungs- und Bewertungsmodell als Einheit aufzufassen und müssen gemeinsam entworfen werden. Dieser Sachverhalt wird dadurch unterstrichen, dass manche Bewertungsfunktionen bestimmte Anregungsfor-

men benötigen, um sinnvolle Ergebnisse zu liefern[5].

Das in Abbildung 4.2 dargestellte erweiterte Streckenmodell bildet aus Sicht eines Optimierungsverfahrens die zu optimierende, vektorielle Zielfunktion $\underline{f}(\underline{p})$. Jegliche Modifikation der verschiedenen Modellbestandteile – des Reglers, der Regelstrecke, der Bewertung und auch der Anregung – stellt direkt eine Veränderung der Zielfunktion dar. Ohne Einschränkung der Allgemeingültigkeit kann die Optimierungsaufgabe als die Minimierung des Zielfunktionsvektors unter Einhaltung der Nebenbedingungen $\underline{g}(\underline{p})$ angesehen werden:

$$\begin{aligned} &\min_{\underline{p}} \underline{f}(\underline{p}) \\ &\underline{g}(\underline{p}) \leq 0 \end{aligned} \tag{4.1}$$

4.2 Anforderungen an mechatronische Systeme

Es existiert eine Reihe von grundsätzlichen Forderungen, die bei der Optimierung regelungstechnischer Systeme beachtet werden müssen. Föllinger formuliert in [Föl94] vier Forderungen, die an jedes regelungstechnische System zu stellen sind:

1. Der Regelkreis muss stabil sein. Dies bedeutet bei linearen zeitinvarianten Systemen, dass seine Führungs- und Störsprungantwort für $t \to \infty$ gegen feste Werte streben müssen.

2. Der Regelkreis muss genügende stationäre Genauigkeit aufweisen. D. h.: Bei Aufschaltung eines Sprungs am Referenzeingang $\underline{w}$, muss die Regelabweichung $\underline{y} - \underline{w}$ für $t \to \infty$ gegen hinreichend kleine Werte streben (für eine Aufschaltung am Störeingang $\underline{z}$ entsprechend).

3. Die Antwort auf einen Führungs- bzw. Störsprung soll hinreichend gedämpft sein, d. h. keine zu starken Oszillationen aufweisen.

4. Der Regelkreis soll genügend schnell sein, d. h. die Antwort auf einen Führungs- bzw. Störsprung soll in hinreichend kurzer Zeit dem stationären Wert genügend nahekommen.

Föllinger teilt diese vier Kriterien ein in die zwei Grundforderungen nach *Stabilität* und *stationärer Genauigkeit* und die beiden qualitativen Forderungen nach hinreichender *Dämpfung* und *Schnelligkeit* des Regelkreises. Die Grundforderungen *Stabilität* und *stationäre Genauigkeit* reichen zur Beurteilung eines geregelten Systems alleine nicht aus. Sie sind aber harte Kriterien, die unter allen Umständen zu erfüllen sind und stellen demnach essentielle Randbedingungen dar, die bei einer Auslegung unbedingt beachtet werden müssen. Die Regelgüte des Systems wird vielmehr durch die beiden qualitativen Forderungen nach hinreichender *Dämpfung* und *Schnelligkeit* des Regelkreises beschrieben.

Kasper kommt in [Kas85] zu einer weiteren Klassifikation der Forderungen, die an geregelte Systeme zu stellen sind. Neben der grundlegenden Forderung, die *Stabilität* des Systems zu gewährleisten, wird die qualitative Bewertung des Störungs- und

[5] Als Beispiel sei hier die Bewertung mit einer zeitgewichteten Fehlerfläche genannt, die sinnvoll nur mit Anregungsfunktionen eingesetzt werden kann, die ein markantes Anfangsereignis aufweisen, wie z. B. die Sprung- oder Rampenfunktion (siehe Abschnitt 4.3.1)

Führungsverhalten anhand von Kriterien im *Zeitbereich* und im *Frequenzbereich* vorgenommen. Die Kriterien des Zeitverhaltens untergliedern sich in das Einschwingverhalten und das stationäre Verhalten. Das Frequenzverhalten beurteilt die Bandbreite des Systems im Gesamtverlauf der interessierenden Frequenzgänge. Diesen Forderungen überlagert ist die Forderung nach *Unempfindlichkeit* und *Robustheit*. D. h. trotz ungenau bekannter oder innerhalb gewisser Grenzen variierender Systemparameter muss die Grundforderung nach Stabilität erfüllt sein, und das Zeit- und Frequenzverhalten sollte sich dabei möglichst wenig ändern.

Eine weitere wichtige Forderung, welche insbesondere bei der Realisierung eines Systems zum Tragen kommt, ist die *Realisierbarkeit der Stellgrößen*. Mit der konstruktiven Ausgestaltung der Aktorik, wird gleichzeitig auch deren Leistungsfähigkeit festgelegt. Stellwerte können nur innerhalb bestimmter Grenzen durch die Aktorik realisiert werden. Ein regelungstechnisches System muss diese Grenzen berücksichtigen. Diese kann beispielsweise durch eine Abbildung der Aktorbeschränkungen im Streckenmodell geschehen. Ein anderer Weg, der oft beschritten wird, ist die Einführung zusätzlicher Kriterien, die eine Bestrafung hoher Stellwerte vornehmen.[6].

Neben diesen klassischen regelungstechnischen Kriterien werden zahlreiche weitere Anforderungen an das mechatronische System gestellt, die sich nicht direkt auf das Regelungsverhalten des Systems beziehen. Beispiele für Kriterien sind der Energieverbrauch oder die Emissionen des Systems, die Belastung und der Verschleiß einzelner Bauteile, die Lärmentwicklung des Systems im Betrieb usw. Diese Auflistung lässt sich beliebig erweitern. Welche Kriterien konkret zum Einsatz kommen, richtet sich im Wesentlichen nach der Beschaffenheit und dem Zweck des Systems, den Einsatzfällen sowie dem Umfeld in dem das System zum Einsatz kommen soll.

Eine notwendige Voraussetzung für die Berücksichtigung solcher Kriterien bei der Optimierung ist eine mathematische Abbildung der zugrundeliegenden physikalischen Effekte im Optimierungsmodell. Diese Effekte werden aus dem Verlauf der Zustände des Systems berechnet oder zumindest abgeschätzt. Gegebenenfalls wird das Modell um zusätzliche Modellbestandteile zur Berechnung der benötigten Größen erweitert. Diese Berechnungen können auch innerhalb des Bewertungsmodells stattfinden. Bei Verwendung von physikalisch motivierten Modellen werden die benötigten Größen aufgrund bekannter physikalischer Gesetzmäßigkeiten direkt berechnet. In vielen Fällen kann jedoch eine solche Formulierung nicht mit einem vertretbaren Modellierungs- oder auch Berechnungsaufwand erfolgen. In diesem Fall werden häufig empirische oder phänomenologische Modelle, z. B. in Form von Kennfeldern, angesetzt, welche die jeweiligen Effekte anhand ihres äußeren Erscheinungsbildes abbilden.

Bei einer Optimierung kann es sinnvoll sein, bestimmte Frequenzbereiche im Bewertungsmodell stärker zu bewerten als andere. Bspw. wird der Komfort für Insassen oft über die frequenzgewichtete Beschleunigung des Fahrzeugaufbaus bemessen (siehe [VDI02]). Die Frequenzgewichtung trägt der Tatsache Rechnung, dass Schwingungen in einigen Frequenzbereichen für Menschen als besonders störend empfunden werden. Dieser Ansatz, die Verwendung eines Bewertungsfilters, ist bereits vom Entwurf von Mehrgrößenreglern bekannt [Kas85, Hen97] und wurde bspw. von Lückel in [LK85] zur Bewertung des Komforts bei aktiven Fahrwerkssystemen eingesetzt. Ebenso können frequenzgewichtete Bewertungskriterien zur Bestrafung unerwünschter Schwingungsanteile eingesetzt werden [Mün03].

[6] Diese Vorgehensweise wird z. B. bei der H_2 Entwurfsmethode verwendet.

4.3 Bewertung im Zeitbereich

Bewertungsfunktionen im Zeitbereich beurteilen das Verhalten auf Basis der zeitlichen Verläufe der Systemzustände, -eingänge und -ausgänge. Die hierfür notwendigen Zeitverläufe werden aus Simulationen des Optimierungsmodells gewonnen. Nichtlineare Modelle erlauben es dabei auch in der Praxis häufig auftretende nichtlineare Effekte, wie Begrenzungen der Aktorik oder nichtlineare Reibung, abzubilden. Der Einsatz nichtlinearer Modelle stellt oft die einzige Möglichkeit dar, solche Effekte in die qualitative Bewertung des Systemverhaltens mit einzubeziehen. Bei vielen Anwendungen kommen daher Gütefunktionen im Zeitbereich zum Einsatz.

4.3.1 Fehlerflächen und zeitgewichtete Fehlerflächen

Die klassischen Bewertungsfunktionen für optimale Regelungen bewerten die Dynamik des Systems über die Regelabweichung, also der Abweichung des Systemausgangs $y(t)$ und der Referenzgröße $w(t)$:

$$e(t) = y(t) - w(t) \tag{4.2}$$

Diese Abweichungen werden in Form eines integralen Gütekriteriums über einen Zeitraum $[0, T]$ zusammengefasst:

$$f = \frac{1}{T} \int_0^T \phi(e(t), t)\, dt \tag{4.3}$$

In Tabelle 4.1 sind die am häufigsten zur Bewertung verwendeten Fehlerflächen zusammengefasst. Minimierung von Betragsflächen und quadratischen Fehlerflächen liefern eine

Tabelle 4.1: Bewertungsfunktionen im Zeitbereich für Regelabweichungen [DB05]

	betragslineare Fehlerflächen	quadratische Fehlerflächen
konstante Gewichtung	$f_{IAE} = \frac{1}{T} \int_0^T \lvert e \rvert\, dt$ IAE (Integral of the Absolute Error)	$f_{ISE} = \frac{1}{T} \int_0^T e^2\, dt$ ISE (Integral of the Squared Error)
mit zeitlicher Gewichtung	$f_{ITAE} = \frac{1}{T} \int_0^T \lvert e \rvert\, t\, dt$ ITAE (Integral of Time multiplied by the Absolute Error)	$f_{ITSE} = \frac{1}{T} \int_0^T e^2\, t\, dt$ ITSE (Integral of Time multiplied by the Squared Error)

Kompromisslösung zwischen einem ungedämpften und einem extrem überdämpften System. Dauerschwingungen werden durch beide Kriterien ebenso bestraft, wie ein langsames überdämpftes Annähern des Referenzwertes. Die quadratische Fehlerfläche bestraft hohe Regelabweichungen stärker und führt daher tendenziell zu einem schnelleren aber weniger stark gedämpften System. Große Anfangsabweichungen, wie sie z. B. bei Sprungantworten auftreten, fallen bei der Betragsfehlerfläche aber vor allem bei der quadratischen Fehlerfläche stark ins Gewicht. Um den großen Einfluss dieser anfänglichen Fehler zu reduzieren, aber auch um später auftretende Auslenkungen und stationäre Abweichungen stärker zu berücksichtigen, bietet sich eine zeitliche Gewichtung der Fehlerflächen an. Neben den

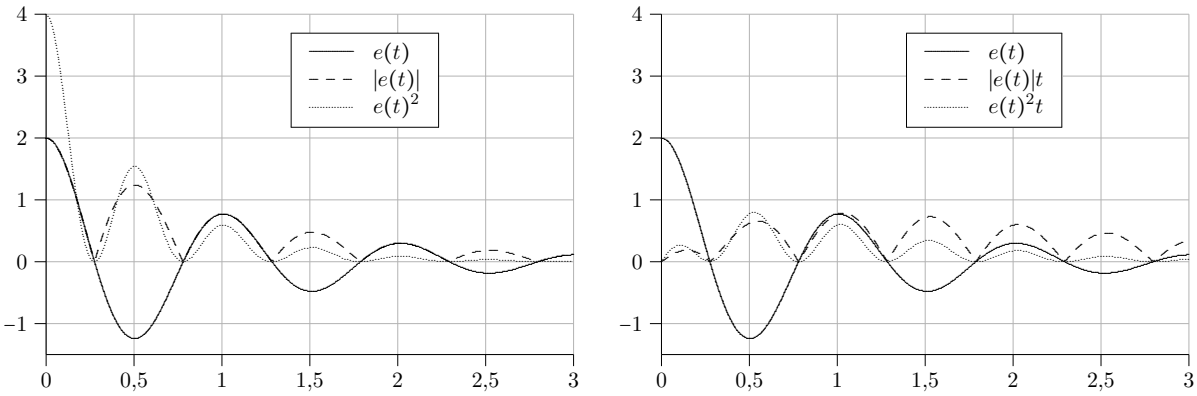


Abbildung 4.3: Veranschaulichung der verschiedenen Fehlerflächen

vorgestellten Fehlerintegralen sind weitere Varianten mit z. B. höheren Exponenten oder quadratischen zeitlichen Gewichtungen denkbar. Ganz allgemein besitzen solche Fehlerintegrale die Form:

$$f = \int_0^T |e|^n \, t^m \, dt \quad \text{mit} \quad n > 0\,,\; m \geq 0 \tag{4.4}$$

In Abbildung 4.3 ist der Unterschied zwischen den vorgestellten Bewertungskriterien veranschaulicht. Dargestellt ist hier der Einschwingvorgang eines PT_2-Gliedes nach einer Sprunganregung, sowie der Verlauf der Integranden der einzelnen Fehlerintegrale.

Betragsfehlerflächen und quadratische Fehlerflächen besitzen die höchste praktische Bedeutung. Zum Beispiel kann die Reduzierung des Kraftstoffverbrauchs eines Fahrzeugs direkt mit der Minimierung einer Betragsfehlerfläche in Beziehung gesetzt werden. Obwohl eigentlich keine generellen Aussagen getroffen werden können, liefern zur Bewertung eines Einschwingvorgangs erfahrungsgemäß die zeitgewichteten Fehlerflächen sehr gute Ergebnisse. Sie sind jedoch nur dann sinnvoll einsetzbar, wenn die Anregungs- bzw. Referenzsignale ein eindeutiges Startereignis und einen stationären Verlauf gegen Ende des Bewertungszeitraums besitzen. Durch das Ereignis lässt sich der Start der zeitlichen Gewichtung festlegen. Meist werden Sprungfunktionen, seltener auch Rampenfunktionen verwendet. Bei Fehlerflächen ohne Zeitgewichtung können im Prinzip beliebige Anregungsfunktionen, z. B. Rauschsignale oder periodische Signale wie Sinusschwingungen oder Rechtecksignale, eingesetzt werden.

4.3.2 Punkt- und abschnittsweise bewertende Kriterien

Eine weitere häufig verwendete Klasse von Bewertungskriterien sind punkt- und abschnittsweise bewertende Kriterien. Diese Kriterien beurteilen das System, anders als die oben beschriebenen Fehlerflächen, nicht am gesamten Zeitverlauf, sondern anhand einzelner Zeitpunkte bzw. Zeitabschnitte.

Mit punktweise bewertenden Funktionen werden einzelne Ereignisse im Zeitverlauf der Simulation detektiert. Die Zielfunktionswerte ergeben sich aus den Simulationsgrößen zum Zeitpunkt des Ereignisses t_e. Wichtige punktweise definierte Funktionen sind das *Maximalwertkriterium* sowie verschiedene *Triggerkriterien*.

Das *Maximalwertkriterium* ermittelt den maximalen während der Simulation auftreten-

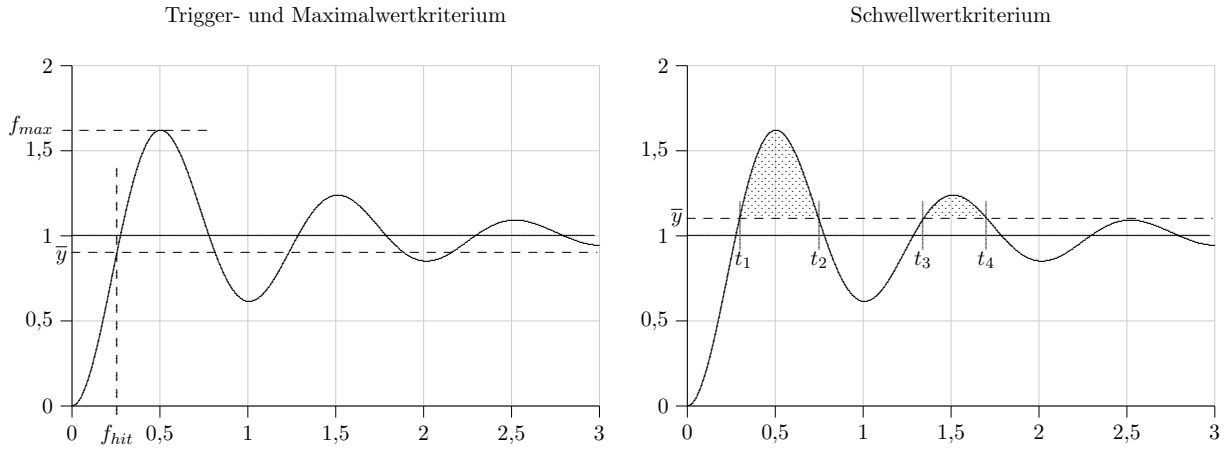


Abbildung 4.4: Bewertung einer Sprungantwort mit punkt- und abschnittsweise bewertenden Kriterien

den Funktionswert:

$$f_{max} = \max_t y(t) \tag{4.5}$$

Die folgende *Triggerfunktion* ermittelt den Zeitpunkt t, bei dem ein bestimmter Schwellwert $\bar{y}$ das erste Mal erreicht wurde:

$$f_{hit} = \min_t \{t \mid y(t) = \bar{y}\} \tag{4.6}$$

Diese beiden Kriterien sind in Abbildung 4.4 (links) am Beispiel der Sprungantwort eines PT_2-Gliedes veranschaulicht. Sie stehen hier in direktem Zusammenhang zu wesentlichen Eigenschaften des Systems. Das Maximalwertkriterium beurteilt den Überschwinger der Sprungantwort und liefert somit ein Maß für Dämpfung. Das Triggerkriterium gibt die Schnelligkeit des Systems an.

Das *Schwellwertkriterium* wird eingesetzt, um das Überschreiten[7] vorgegebener Grenzen graduell zu bestrafen. Diese abschnittsweise bewertende Funktion ergibt aus der positiven Fläche zwischen dem Systemausgang $y(t)$ und dem Schwellwert $\bar{y}(t)$:

$$f_{ueber} = \int_0^T \max(y(t) - \bar{y}(t), 0)\, dt \tag{4.7}$$

Anders als beim *Maximalwertkriterium* fließt hier mit der Fläche sowohl die Höhe als auch der Zeitraum der Über- bzw. Unterschreitung in die Zielgröße ein. Das *Schwellwertkriterium* wird in Abbildung 4.4 (rechts) zur Beurteilung des Überschwingers einer Sprungantwort genutzt.

[7] Die Bewertung einer Schwellwertunterschreitung kann durch Vorzeichenänderung von $y(t)$ und $\bar{y}(t)$ erreicht werden.

5 Sensitivitätsanalyse

In vielen naturwissenschaftlichen und technischen Problemstellungen werden für differenzierbare Funktionen neben den Funktionswerten auch die Ableitungen erster oder höherer Ordnung benötigt. Beispiele für solche Problemstellungen sind bei der Untersuchung von Parameterempfindlichkeiten, bei der Lösung von nichtlinearen Gleichungssystemen und Differenzialgleichungen, aber vor allem auch bei der Optimierung zu finden. Insbesondere arbeiten die in Kapitel 3 beschriebenen Optimierungsverfahren mit den Ableitungen der Zielfunktionen nach den Entwurfsparametern. Eine erfolgreiche Anwendung dieser Verfahren hängt wesentlich von der Genauigkeit der Ableitungsbrechnung der Zielfunktionen und Nebenbedingungen ab.

Die Ableitung einer skalaren Funktion $f : \mathbb{R}^n \to \mathbb{R}$ nach den Parametern $\underline{x} \in \mathbb{R}^n$ wird durch den Gradienten:

$$\operatorname{grad} f(\underline{x}) = \nabla f(\underline{x}) = \left[\begin{array}{ccc} \frac{\partial f}{\partial x_1}\Big|_{\underline{x}} & \cdots & \frac{\partial f}{\partial x_n}\Big|_{\underline{x}} \end{array} \right] \tag{5.1}$$

an einer beliebigen Stelle $\underline{x}$ angegeben. Im Falle einer Vektorfunktion $\underline{f} : \mathbb{R}^n \to \mathbb{R}^m$ erhält man die Jacobi-Matrix:

$$\mathcal{J}_{\underline{f}}(\underline{x}) = \nabla \underline{f}(\underline{x}) = \left[\begin{array}{ccc} \frac{\partial f_1}{\partial x_1}\Big|_{\underline{x}} & \cdots & \frac{\partial f_1}{\partial x_n}\Big|_{\underline{x}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}\Big|_{\underline{x}} & \cdots & \frac{\partial f_m}{\partial x_n}\Big|_{\underline{x}} \end{array} \right] \tag{5.2}$$

Eine symbolische Differentiation der Funktion f ist trotz der Verfügbarkeit von Computeralgebrasystemen bei vielen Anwendungen nicht möglich, da die Funktionswerte oft durch aufwändige numerische Algorithmen oder Simulationen berechnet werden und daher nur in Form eines Computerprogramms vorliegen. Eine gängige Praxis ist es, diese Gradienten durch mehrfache Auswertung von f mit Hilfe von Differenzenquotienten numerisch anzunähern. Im folgenden Abschnitt wird zunächst auf diese Technik näher eingegangen. Anschließend wird eine alternative Methode zur Berechnung der Ableitungen von Funktionen, die als Computeralgorithmus vorliegen, die *Algorithmische Differentiation*, vorgestellt. Dabei besitzen beide Methoden spezifische Vor- und Nachteile und haben daher ihre Daseinsberechtigung.

5.1 Differenzenquotienten

Vorwärtsdifferenzenquotient

Die Berechnung der Differenzenquotienten wird motiviert durch die Definition der Ableitung. Die Ableitung einer skalaren Funktion $f : \mathbb{R} \to \mathbb{R}$ im Punkt $x \in \mathbb{R}$ ist definiert durch den Grenzwert:

$$f' = \frac{\mathrm{d}f}{\mathrm{d}x} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{5.3}$$

Wird h klein aber ungleich Null gewählt, dann ergibt der rechte Ausdruck eine Näherung für die Ableitung f', den sogenannten Vorwärtsdifferenzenquotienten:

$$f' \approx \frac{f(x+h) - f(x)}{h} \tag{5.4}$$

Die Berechnung von f' erfolgt bei (5.4) durch zweifache Auswertung der Funktion f. Der Diskretisierungsfehler ist von der Ordnung $\mathcal{O}(h)$, d. h. er wächst proportional mit der Diskretisierungsschrittweite h an.[1] Man wird daher erwarten, dass die Näherung der Ableitung durch (5.4) besser wird, wenn die Schrittweite h verkleinert wird. Dies ist jedoch nur bedingt der Fall, da mit kleiner werdendem h auch die Auslöschung im Zähler des Differenzenquotienten zunimmt. Im Prinzip muss h so gewählt werden, dass die Summe aus Diskretisierungsfehler und dem Fehler auf Grund der Auslöschung minimal wird. Das ist ungefähr dann der Fall, wenn beide Fehler die gleiche Größenordnung besitzen. Dies führt auf die folgende Abschätzung für eine optimale Schrittweite $h^\star$ [CDB80, Nag02]:

$$h^\star = 2\sqrt{\frac{(1 + |f(x)|)\,\epsilon_{rel}}{|f''(x)|}} \tag{5.5}$$

wobei $f''(x)$ für die zweite Ableitung von $f(x)$ steht und ϵ_{rel} eine Abschätzung des relativen Fehlers bei der Berechnung von $f(x)$ angibt.

Zentraldifferenzenquotient

Eine bessere Näherung kann mit Hilfe des Zentraldifferenzenquotienten:

$$f' \approx \frac{f(x+h) - f(x-h)}{2h} \tag{5.6}$$

erreicht werden. Der Diskretisierungsfehler von (5.6) ist von der Ordnung $\mathcal{O}(h^2)$. Er verringert sich demnach bei kleiner werdendem h quadratisch, so dass vergleichsweise große Diskretisierungsschrittweiten möglich werden. Dies wiederum führt zu geringeren Auslöschungseffekten.

Differenzenquotienten höheren Grades

Alternativ können auch Differenzenquotienten höheren Grades gebildet werden. Im Folgenden sind einige symmetrische Formeln angeführt:

[1] Das Landau-Symbol $\mathcal{O}$ steht für die Größenordnung von Funktionen. Landau-Symbole sind benannt nach dem deutschen Zahlentheoretiker Edmund Landau (1877-1938). Sie werden in der Mathematik und in der Informatik verwendet, um das asymptotische Verhalten von Funktionen und Folgen bei Annäherung an einen endlichen oder unendlichen Grenzwert zu beschreiben. [BSMM95]

Tabelle 5.1: Symmetrische Differenzenquotienten höheren Grades [Hol08]

N	N-Punkte-Differenzenquotienten
5	$\dfrac{f_{-2} - 8f_{-1} + 8f_{+1} - f_{+2}}{12h}$
7	$\dfrac{-f_{-3} + 9f_{-2} - 45f_{-1} + 45f_{+1} - 9f_{+2} + f_{+3}}{60h}$
9	$\dfrac{3f_{-4} - 32f_{-3} + 168f_{-2} - 672f_{-1} + 672f_{+1} - 168f_{+2} + 32f_{+3} - 3f_{+4}}{840h}$
	mit $f_k = f(x + k \cdot h)$

Die Differenzenquotienten in Tabelle 5.1 basieren auf Taylorreihenansätzen und ermitteln daher die erste Ableitung von Polynomen bis zum Grad $N - 1$ exakt. Durch die größere Anzahl an Stützstellen werden bei der Herleitung Fehlerterme niedriger Ordnung eliminiert und es ergeben sich Diskretisierungsfehler mit $\mathcal{O}(h^{N-1})$. Der resultierenden gute Approximationsgenauigkeit stehen jedoch vielfache Funktionsauswertungen und somit ein gesteigerter Rechenaufwand entgegen.

Differenzenquotienten vektorwertiger Funktionen

Zur Approximation der Jacobimatrix (5.2) einer vektorwertigen Funktion $\underline{f} : \mathbb{R}^n \to \mathbb{R}^m$ wird für jedes Element des Parametervektors $\underline{x} \in \mathbb{R}^n$ ein eigener vektorwertiger Differenzenquotient ermittelt. Zur Berechnung eines Elements $\partial f_j / \partial x_i$ der Jacobimatrix wird der Parametervektor in Richtung des i-ten Einheitsvektors $\underline{e}_i \in \mathbb{R}^n$ ausgelenkt. Im Falle von Vorwärtsdifferenzenquotienten berechnet sich das Element zu:

$$\frac{\partial f_j}{\partial x_i} \approx \frac{f_j(\underline{x} + h\,\underline{e}_i) - f_j(\underline{x})}{h} \tag{5.7}$$

Die Differentiation mit Differenzenquotienten höheren Grades verläuft analog zu (5.7) mit $f_{j,i,k} = f_j(\underline{x} + k\,h\,\underline{e}_i)$.

Bei Verwendung von Vorwärtsdifferenzenquotienten erfordert die Berechnung der kompletten Jacobimatrix insgesamt $n + 1$ Auswertungen der Vektorfunktion $\underline{f}$; bei Verwendung der Differenzenquotienten aus Tabelle 5.1 sind insgesamt $n \cdot (N - 1)$ Auswertungen notwendig.

Im Allgemeinen ist die optimale Diskretisierungsschrittweite einer Parameteränderung $h\,\underline{e}_i$ für die einzelnen Funktionen in $\underline{f}$ unterschiedlich. Idealer Weise muss die Wahl der Diskretisierungsschrittweite für jede Funktion f_j und jeden Parameter x_i unabhängig getroffen werden, d. h. für jede partielle Ableitung $\partial f_j / \partial x_i$ ergibt sich eine individuelle optimale Schrittweite $h_{i,j}$.

Sind die einzelnen Funktionen f_j unabhängig voneinander zu berechnen, so kann für jede Funktion ohne weiteren Rechenaufwand eine individuelle Schrittweite gewählt werden. Speziell bei mechatronischen Systemen werden aber mehrere, wenn nicht sogar alle Funktionswerte aus einer einzelnen Simulation ermittelt. Bei Berücksichtigung der individuellen Schrittweite sind daher zusätzliche Simulationen notwendig. Die Anzahl der benötigten Funktionsauswertungen steigt im ungünstigsten Fall auf $m \cdot n \cdot (N - 1)$. Dem kann jedoch

ein Stück weit begegnet werden, indem man die Differenzenquotienten von Funktionen mit einer ähnlichen Diskretisierungsschrittweite in einem Parameter x_i gemeinsam ermittelt.

Zusammenfassend ergeben sich bei den Differenzenquotienten die folgenden Vor- und Nachteile:

+ Die Funktion f kann als *Black-Box* behandelt werden, d. h. es werden nur die Funktionswerte benötigt. Sie kann dabei das Ergebnis einer beliebig komplexen Berechnung sein. So ist z. B. die Verwendung kommerzieller Simulationsmodelle, die nicht im Quellcode vorliegen, möglich. Aus diesem Grund stellen die Differenzenquotienten oft die einzige Chance für den Entwickler dar, die benötigten Gradienten zu ermitteln. Ebenso ist der Entwicklungsaufwand zur Berechnung der Differenzenquotienten sehr gering.

– Die Wahl des Diskretisierungsparameters h beeinflusst stark die Genauigkeit der Approximation. Um den Diskretisierungsparameters optimal auszulegen, ist zusätzlicher Aufwand nötig. Die NAG-Bibliothek [Nag02] und die GNU Scientific Library (GSL) [GDT+06] liefern beispielsweise Routinen zur Bestimmung eines optimalen h.

– Die Näherung der Ableitung kann auch bei optimal ausgelegtem Diskretisierungsparameter h ungenau sein. Als Erfahrungswert gilt, dass bei Vorwärtsdifferenzenquotienten ca. die Hälfte der Stellen ausgelöscht werden (bei Zentraldifferentiation ist es etwa ein Drittel) [Gri00]. Dieses Problem verstärkt sich bei „verrauschten“ Zielfunktionen, z. B. wenn die Berechnung der Zielfunktionen selbst aufgrund von signifikanten Rundungs- bzw. Auslöschungseffekten nicht exakt durchgeführt werden kann.

5.2 Arbeitsweise der Algorithmischen Differentiation

Die *Algorithmische Differentiation (AD)* ist eine teilsymbolische Technik, die die Berechnung der Gradienten von Funktionen ermöglicht, die in Form eines Computerprogramms, bspw. erstellt in einer Hochsprache wie Fortran oder C/C++, vorliegen. Dabei wird, ausgehend von der zu differenzierenden Funktion, keine explizite Ableitungsformel, sondern ein Algorithmus für die Berechnung des Gradienten gebildet. Die resultierenden Ergebnisse sind im Rahmen der Maschinengenauigkeit exakt. Da keine Näherungsformel herangezogen wird, müssen ausschließlich Rundungsfehler beachtet werden; diese treten jedoch bereits bei der Berechnung des Funktionswertes auf. Die berechneten Ableitungen haben in der Regel eine vergleichbare Genauigkeit wie die Funktionswerte selbst [Gri00]. Die Anwendung der AD erfolgt weitgehend automatisiert[2], d. h. mit einem Minimum an Aufwand für den Anwender. Sowohl einfache als auch umfangreiche Computerprogramme lassen sich effizient differenzieren, wobei der Berechnungsaufwand und Speicherplatzbedarf moderat mit der Programmgröße und der Anzahl der zu berechnenden Ableitungen ansteigen.

Die AD nutzt die Tatsache, dass jedes Computerprogramm aus einer Folge von elementaren, arithmetischen Operationen und Funktionen besteht. Eine solche Folge von Operationen kann als ein azyklischer, gerichteter Auswertegraph eindeutig beschrieben werden.

[2] Aus diesem Grund wird diese Technik oft auch als *Automatische Differentiation* bezeichnet.

Dies soll anhand der folgenden Beispielfunktion erläutert werden:

$$f(\underline{x}) = \sin(2 \cdot x_1) - e^{x_1/x_2} \tag{5.8}$$

Zur Auswertung wird diese Funktion in eine Folge von elementaren Operationen[3] abgebildet. In Abbildung 5.1 ist diese Folge als Auswertegraph dargestellt.

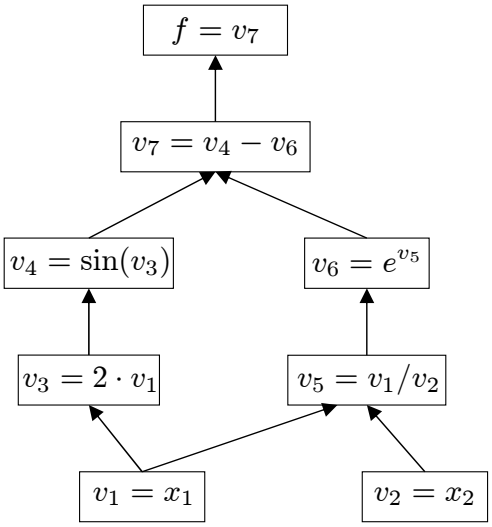


Abbildung 5.1: Auswertegraph der Funktion (5.8)

In diesem Auswertegraphen werden die Knoten den Eingangsgrößen (unabhängige Variablen) x_i, den Zwischengrößen v_i und der Ausgangsgröße f (abhängige Variable) zugeordnet. Während die Eingangsgrößen von vornherein bekannt sind, werden die Zwischengrößen und die Ausgangsvariablen sequentiell durch Auswertung der jeweiligen elementaren Rechenoperationen ermittelt.

Zur Berechnung der möglicherweise sehr komplexen Ableitung $\frac{df}{dx_i}$ der Gesamtfunktion werden zunächst die elementaren Operationen separat differenziert und die Ableitungen entlang der Kanten des Graphen 5.1 angetragen. Diese Kantenableitungen geben an, wie sich Veränderungen der Vorgängerknoten auf den betrachteten Knoten auswirken. Die Differentiationsregeln können auf die einzelnen Operationen sehr einfach z. B. durch einen Tabellenvergleich angewendet werden. Abbildung 5.2 zeigt den Graphen aus 5.1 mit seinen Kantenableitungen.

Die Bestimmung der Ableitungen der Ausgangsvariablen nach den Eingangsvariablen erfolgt durch wiederholtes Anwenden der Kettenregel auf die Kantenableitungen des Auswertegraphen.

Für das Beispiel (5.8) berechnen sich die Ableitungen df/dx_1 und df/dx_2 wie folgt:

$$\begin{aligned} \frac{\mathrm{d}f}{\mathrm{d}x_1} &= \frac{\partial v_7}{\partial v_1} = \frac{\partial v_7}{\partial v_4}\frac{\partial v_4}{\partial v_3}\frac{\partial v_3}{\partial v_1} + \frac{\partial v_7}{\partial v_6}\frac{\partial v_6}{\partial v_5}\frac{\partial v_5}{\partial v_1} = 1 \cdot \cos(v_3) \cdot 2 + (-1) \cdot e^{v_5} \cdot \frac{1}{v_2} \\ \frac{\mathrm{d}f}{\mathrm{d}x_2} &= \frac{\partial v_7}{\partial v_2} = \frac{\partial v_7}{\partial v_6}\frac{\partial v_6}{\partial v_5}\frac{\partial v_5}{\partial v_2} = (-1) \cdot e^{v_5} \cdot \frac{-v_1}{{v_2}^2} \end{aligned} \tag{5.9}$$

[3] Zu den wichtigsten elementaren Operatoren zählen u. a.: $+, -, \times, /, x^y, \sin, e^{(\,)}, \ln, \mathrm{sgn}$

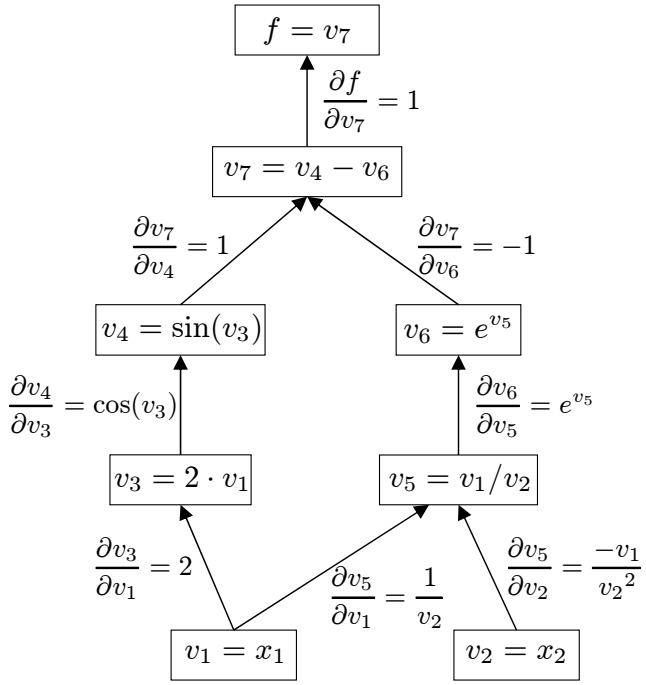


Abbildung 5.2: Auswertegraph der Funktion (5.8) mit den Kantenableitungen

Durch diesen Formalismus erweitert man das ursprüngliche Programm um die Ableitungen der einzelnen Operationen und verknüpft diese Ableitung durch Anwendung der Kettenregel. Diese zusätzlichen Gleichungen schließlich propagieren die gewünschten Ableitungen entlang des Auswertegraphen. Anders als beim symbolischen Differenzieren wird bei der AD somit keine explizite Ableitungsformel generiert, sondern der bestehende Algorithmus um zusätzliche Gleichungen zur Berechnung der Ableitungen erweitert. Die Berechnung erfolgt also nicht symbolisch, sondern auf Basis konkreter Zahlenwerte. Interessant hierbei ist, dass dabei auch Konstrukte wie Verzweigungen und Schleifen problemlos behandelt werden können. Die AD eignet sich daher grundsätzlich auch zur Anwendung auf sehr umfangreiche Berechnungsprogramme.

Die Auswertung der Kettenregel in (5.9) kann sowohl in Vorwärts- als auch in Rückwärtsrichtung erfolgen. Diese beiden Varianten werden im Folgenden vorgestellt.

5.2.1 Vorwärtsmodus

Die Differentiation im Vorwärtsmodus entspricht einer Auswertung der Kettenregel ausgehend von den Eingangsvariablen hin zu den Ausgangsvariablen. Angewendet auf einen Auswertegraphen bedeutet dies, dass die Berechnung der Ableitungen in derselben Richtung erfolgt wie die Funktionsauswertung.

Im Vorwärtsmodus wird jeder Zwischenvariablen v_i des Auswertegraphen ein Gradientenvektor $\underline{\dot{v}}_i \in \mathbb{R}^n$ zugewiesen. Dieser Gradientenvektor beschreibt die Ableitung der Zwischenvariablen v_i nach dem Vektor der unabhängigen Variablen $\underline{x}$:

$$\underline{\dot{v}}_i = \frac{\partial v_i}{\partial \underline{x}^{\mathrm{T}}} \tag{5.10}$$

Die Berechnung der Gradienten erfolgt in drei Schritten:

- *Initialisierung:* Vor dem eigentlichen Start der Funktions- und Gradientenberechnung werden die direkt den unabhängigen Variablen zugeordneten Gradienten jeweils mit dem i-ten Einheitsvektor $\underline{e}_i \in \mathbb{R}^n$ initialisiert: $\underline{\dot{v}}_i = \underline{e}_i$ für $i = 1, \ldots, n$. Dieser Anfangswert stellt die Quelle[4] für das Propagieren der Gradienten durch den Auswertegraphen dar. Durch diese Zuordnung wird festgelegt, dass das i-te Element eines Gradientenvektors $\underline{\dot{v}}_j$ die Ableitung nach der unabhängigen Variable x_i beschreibt: $\dot{v}_{j_i} = \partial v_j / \partial x_i$.

- *Berechnung:* Die anschließende Berechnung der Ableitungen erfolgt beim Vorwärtsmodus parallel zur Funktionsauswertung, d. h. zusammen mit der Berechnung eines Zwischenwertes v_i wird auch der zugehörige Gradient $\underline{\dot{v}}_i$ ermittelt. Hierdurch sind alle Eingangsabhängigkeiten für die nachfolgende Funktion im Auswertegraphen v_{i+1} und deren Ableitungen $\underline{\dot{v}}_{i+1}$ erfüllt, so dass mit der Berechnung dieser Funktion fortgefahren werden kann. Nach Abarbeitung des kompletten Auswertegraphen liegen dann sowohl die Werte aller Zwischenvariablen als auch deren Gradienten als numerische Werte vor.

- *Ergebniszuweisung:* Am Ende erfolgt die Zuweisung des Funktionswertes an f und des Gradienten an $\underline{\dot{f}}$.

Angewendet auf das Beispiel (5.8) erhält man die in Tabelle 5.2 angegebene Berechnungsfolge. Die Zahlenwerte wurden für die Eingangswerte $x_1 = 2{,}0$ und $x_2 = 3{,}0$ ermittelt.

Tabelle 5.2: Berechnung des Funktionswertes und des Gradienten von (5.8) im Vorwärtsmodus

Funktionswert		Ableitungen		$\frac{\partial}{\partial x_1}$	$\frac{\partial}{\partial x_2}$
$v_1 = x_1$	2,0	$\underline{\dot{v}}_1 = (1\ 0)^{\mathrm{T}}$		**1,0**	0,0
$v_2 = x_2$	3,0	$\underline{\dot{v}}_2 = (0\ 1)^{\mathrm{T}}$		0,0	**1,0**
$v_3 = 2 \cdot v_1$	4,0	$\underline{\dot{v}}_3 = 2 \cdot \underline{\dot{v}}_1$		2,0	0,0
$v_4 = \sin(v_3)$	0,06976	$\underline{\dot{v}}_4 = \underline{\dot{v}}_3 \cdot \cos(v_3)$		1,995	0,0
$v_5 = v_1 / v_2$	0,6667	$\underline{\dot{v}}_5 = (\underline{\dot{v}}_1 \cdot v_2 - v_1 \cdot \underline{\dot{v}}_2) / {v_2}^2$		0,3333	−0,2222
$v_6 = e^{v_5}$	1,948	$\underline{\dot{v}}_6 = \underline{\dot{v}}_5 \cdot e^{v_5}$		0,6492	−0,4328
$v_7 = v_4 - v_6$	−1,878	$\underline{\dot{v}}_7 = \underline{\dot{v}}_4 - \underline{\dot{v}}_6$		1,346	0,4328
$f = v_7$	−1,878	$\underline{\dot{f}} = \underline{\dot{v}}_7$		**1,346**	**0,4328**

Rechenaufwand

Für jede Zwischenvariable muss beim Vorwärtsmodus ein Gradientenvektor der Länge n bestimmt werden. Der Berechnungsaufwand steigt daher linear mit der Anzahl n der Eingangsvariablen an. Anhand der Quotientenregel in Tabelle 5.2 wird jedoch ersichtlich, dass

[4] Im Englischen wird dieser Anfangswert anschaulich mit *seed* – Saat – bezeichnet.

Tabelle 5.3: Berechnungsaufwand für die Gradienten einiger elementarer Funktionen im Vorwärtsmodus. *MEM* gibt die Anzahl der benötigten Lese- und Schreibzugriffe[5] an, *NLOP* bezeichnet die Anzahl der nichtlinearen Funktionsauswertungen.

Funktion	Gradient	*MEM*	$+,-$	$\times,/$	*NLOP*
$v = u$	$\underline{\dot{v}} = \underline{\dot{u}}$	$2+2n$			
$v = u_1 \pm u_2$	$\underline{\dot{v}} = \underline{\dot{u}}_1 \pm \underline{\dot{u}}_2$	$3+3n$	$1+n$		
$v = u_1 \cdot u_2$	$\underline{\dot{v}} = \underline{\dot{u}}_1 \cdot u_2 + u_1 \cdot \underline{\dot{u}}_2$	$3+3n$	n	$1+2n$	
$v = u_1/u_2$	$\underline{\dot{v}} = (\underline{\dot{u}}_1 - v \cdot \underline{\dot{u}}_2)/u_2$	$3+3n$	n	$1+2n$	
$v = \sqrt{u}$	$a = 0{,}5/v$ $\underline{\dot{v}} = a \cdot \underline{\dot{u}}$	$2+2n$		$1+n$	1
$v = u^w$	$a = w \cdot v/u\,;\ b = v \cdot \ln(u)$ $\underline{\dot{v}} = a \cdot \underline{\dot{u}} + b \cdot \underline{\dot{w}}$	$3+3n$	n	$3+2n$	2
$v = e^u$	$\underline{\dot{v}} = v \cdot \underline{\dot{u}}$	$2+2n$		n	1
$v = \ln(u)$	$\underline{\dot{v}} = \underline{\dot{u}}/u$	$2+2n$		n	1
$v = \sin(u)$	$a = \cos(u)$ $\underline{\dot{v}} = a \cdot \underline{\dot{u}}$	$2+2n$		n	2

zur Berechnung einer einzelnen Ableitung (mit $n = 1$) mehr Rechenoperationen erforderlich sind, als für die Auswertung der ursprünglichen Funktion. Griewank hat unter realistischen Annahmen gezeigt, dass das Verhältnis des Aufwandes zur Berechnung einer skalaren Richtungsableitung $T_{\dot{f}_i}$ zum Aufwand der Funktionsauswertung T_f zwischen 2 und 2,5 liegt (siehe [Gri00]):

$$T_{\dot{f}_i} \leq 2{,}5 \cdot T_f \tag{5.11}$$

Die Berechnung der gesamten Jacobimatrix geschieht mit einem Aufwand von:

$$T_{\underline{\dot{f}}} \leq (1 + 1{,}5\,n) \cdot T_f \tag{5.12}$$

Der Aufwand ist im Vorwärtsmodus also unabhängig von der Anzahl m der Ausgangsgrößen.

Tabelle 5.3 zeigt die Berechnungsvorschriften für einige elementare Funktionen. In den vier rechten Spalten ist der jeweilige Berechnungsaufwand aufgeführt, der für die Ermittlung des Funktionswertes und des Gradientenvektors der Länge n erforderlich ist. Der Aufwand ist hierbei in der Anzahl der benötigten Operationen angegeben. Die einzelnen Berechnungsvorschriften wurden dahingehend optimiert, dass mehrfach verwendete Teilausdrücke nach Möglichkeit zusammengefasst und so nur einmal ausgewertet werden. Dies verringert die Anzahl der Rechenoperationen bei den vektoriellen Gleichungen der Gradienten.

Speziell bei den „teuren“ nichtlinearen Rechenoperationen wie Wurzeln, reellen Potenzen oder trigonometrischen Funktionen wirkt sich positiv aus, dass die Funktionsauswertung und die Berechnung der äußeren Ableitung wesentlich aufwändiger sind, als die aus der

[5] Die temporären Variablen a und b wurden in der Spalte *MEM* nicht berücksichtigt. Diese Variablen besitzen eine sehr kurze Lebensdauer, werden aber häufig benutzt. Solche Variablen werden oft innerhalb von Prozessorregistern gehalten. In diesem Fall entfällt der Aufwand für die Speicherzugriffe.

Kettenregel resultierende Multiplikation mit der inneren Ableitung. Im vektoriellen Fall müssen Funktionswert und äußere Ableitung nur einmal berechnet werden, so dass unabhängig von Vektorlänge n maximal zwei nichtlineare Rechenoperationen für beliebige nichtlineare, unäre Funktionen notwendig sind.

Demnach steigt der Rechenaufwand im Vektormodus moderat mit der Größe der Gradientenvektoren an, da speziell bei nichtlinearen Funktionen der zusätzliche Aufwand für weitere Richtungsableitungen relativ gering gehalten werden kann.

5.2.2 Rückwärtsmodus

Bei einer großen Anzahl unabhängiger Variablen erweist sich die Differentiation im Vorwärtsmodus als ungünstig, da der Rechenaufwand linear mit der Gradientenlänge n ansteigt. In solchen Fällen bietet sich die Differentiation im sogenannten Rückwärtsmodus an. Im Folgenden soll zunächst der Fall einer skalaren Ausgangsvariablen f betrachtet werden.

Beim Rückwärtsmodus wird die Ableitung der Ausgangsvariablen f nach den jeweiligen Zwischenvariablen v_i rückwärts entlang des Auswertegraphen propagiert. Einer Zwischenvariablen v_i wird eine skalare Größe $\bar{v}_i$ zugeordnet, welche die totale Ableitung der Ausgangsvariablen f nach der jeweiligen Zwischenvariablen repräsentiert:

$$\bar{v}_i = \frac{\mathrm{d}f}{\mathrm{d}v_i} \tag{5.13}$$

Die Berechnung der einzelnen Kantenableitungen muss dabei auf Basis der Zahlenwerte der Zwischenvariablen v_i erfolgen (siehe Abbildung 5.2). Da die Funktionsauswertung nur vorwärts entlang des Auswertegraphen erfolgen kann, muss die Berechnung des Gradienten in zwei Phasen unterteilt werden.

- *Funktionswertberechnung:* In der ersten Phase wird die Auswertung der Zwischengrößen v_i sowie der Funktion f vorgenommen. Die Zwischenwerte werden dabei gespeichert. Auf ihrer Basis erfolgt dann in der zweiten Phase die Gradientenberechnung.

- *Gradientenberechnung:* In der zweiten Phase wird der Auswertegraph rückwärts durchlaufen und die totalen Ableitungen $\bar{v}_i$ werden ausgehend von der Ausgangsvariablen hin zu den Eingangsvariablen durch den Graphen propagiert. Die Berechnung von $\bar{v}_i$ geschieht dabei durch Verknüpfung der Kantenableitungen mit Hilfe der Kettenregel. Vor der Gradientenberechnung muss ähnlich wie beim Vorwärtsmodus eine Initialisierung der hierbei beteiligten Variablen stattfinden. Die Ableitungen der Zwischengrößen werden mit $\bar{v}_i = 0$ und der Startwert der totalen Ableitung mit $\bar{f} = 1$ initialisiert.

In Tabelle 5.4 sind die Operationen zur Berechnung des Gradienten im Rückwärtsmodus für die Beispielfunktion (5.8) angegeben. Die Kurzschreibweise $a \mathrel{+}= b$ entstammt der Programmiersprache C und steht für die Zuweisung $a = a + b$.

Tabelle 5.4: Rechenweg des Funktionswertes und der Ableitungen von (5.8) im Rückwärtsmodus

Funktionswertberechnung:

v_1	=	x_1	2,0
v_2	=	x_2	3,0
v_3	=	$2 \cdot v_1$	4,0
v_4	=	$\sin(v_3)$	0,06976
v_5	=	v_1 / v_2	0,6667
v_6	=	e^{v_5}	1,948
v_7	=	$v_4 - v_6$	−1,878
f	=	v_7	−1,878

Gradientenberechnung:

$\bar{f}$	=	1	1,0
$\bar{v}_i$	=	$0\,, \quad i = 1 \ldots 7$	0,0
$\bar{v}_7$	+=	$\bar{f}$	1,0
$\bar{v}_6$	+=	$-\bar{v}_7$	−1,0
$\bar{v}_5$	+=	$\bar{v}_6 \cdot e^{v_5}$	−1,948
$\bar{v}_2$	+=	$-\bar{v}_5 \cdot v_1 / {v_2}^2$	0,4328
$\bar{v}_1$	+=	$\bar{v}_5 / v_2$	−0,6492
$\bar{v}_4$	+=	$\bar{v}_7$	1,0
$\bar{v}_3$	+=	$\bar{v}_4 \cdot \cos(v_3)$	0,9976
$\bar{v}_1$	+=	$\bar{v}_3 \cdot 2$	1,346
$\bar{\mathbf{x}}_\mathbf{1}$	=	$\bar{v}_1$	**1,346**
$\bar{\mathbf{x}}_\mathbf{2}$	=	$\bar{v}_2$	**0,4328**

Rechenaufwand

Der Aufwand zur Berechnung einer skalaren Richtungsableitung ist im Rückwärtsmodus ca. zwei- bis dreimal so groß wie der Rechenaufwand der ursprünglichen Funktion und somit nur unwesentlich größer als im Vorwärtsmodus (vgl. (5.11)). Betrachtet man das Beispiel in Tabelle 5.4, so fällt auf, dass die Vorschrift zur Berechnung einer Zwischenvariablen v_i lediglich um eine skalare Vorschrift erweitert wird, die der Bestimmung des totalen Differentials $\bar{v}_i = \frac{\mathrm{d}f}{\mathrm{d}v_i}$ dient. Der Pfad zur Berechnung dieses totalen Differentials verläuft von der Ausgangsvariablen f hin zu der Zwischenvariablen v_i und die Knoten der Eingangsvariablen x_j treten in diesem Pfad nicht auf. Sind die Werte der Zwischenvariablen und der Funktionswert bekannt, erfolgt die Berechnung der Ableitungen $\bar{v}_i$ unabhängig vom Eingangsvektor $\underline{x}$. Der für die Berechnung des Gradienten $\nabla_{\underline{x}} f$ erforderliche Aufwand steigt nicht mit der Anzahl n der unabhängigen Variablen an. Für den Rechenaufwand des Rückwärtsmodus gilt daher unabhängig von der Anzahl der Eingangsgrößen:

$$T_{\underline{\bar{x}}} \leq 4 \cdot T_f \tag{5.14}$$

Diese Abschätzung ist jedoch als obere Grenze anzusehen. Meist werden in praktischen Anwendungen Werte zwischen 2 und 3 beobachtet [Gri00].

Im Gegenzug steigt der Rechenaufwand im Rückwärtsmodus linear mit der Anzahl m der Ausgangsgrößen. Im Falle eines Ausgangsvektors $\underline{f}$ muss der Gradient dieses Vektors nach den Zwischengrößen $\bar{\underline{v}}_i = \mathrm{d}\underline{f}/\mathrm{d}v_i$ berechnet und entlang des Auswertegraphen propagiert werden. Der Rechenaufwand beträgt dann:

$$T_{\bar{\underline{x}}} \leq (1{,}5 + 2{,}5\,m) \cdot T_f \tag{5.15}$$

In Tabelle 5.5 ist der Aufwand einiger elementarer Rechenfunktionen für einen Ausgangsvektor $\underline{f}$ der Größe m angegeben.

Tabelle 5.5: Berechnungsaufwand für die Ableitungen einiger elementarer Funktionen im Rückwärtsmodus. *MEM* gibt die minimale Anzahl der benötigten Lese- und Schreibzugriffe und *NLOP* die Anzahl der nichtlinearen Funktionsauswertungen an.

Funktion	Ableitung	*MEM*	+, −	×, /	*NLOP*
$v = u$	$\bar{\underline{u}}$ += $\bar{\underline{v}}$	2+2 m	m		
$v = u_1 \pm u_2$	$\bar{\underline{u}}_1$ += $\bar{\underline{v}}$ $\bar{\underline{u}}_2$ ±= $\bar{\underline{v}}$	3+3 m	1+2 m		
$v = u_1 \cdot u_2$	$\bar{\underline{u}}_1$ += $u_2 \cdot \bar{\underline{v}}$ $\bar{\underline{u}}_2$ += $u_1 \cdot \bar{\underline{v}}$	3+3 m	2 m	1+2 m	
$v = u_1/u_2$	$a = v/u_2$ $\bar{\underline{u}}_1$ += $\bar{\underline{v}}/u_2$ $\bar{\underline{u}}_2$ −= $a \cdot \bar{\underline{v}}$	3+3 m	2 m	2+2 m	
$v = \sqrt{u}$	$a = 0{,}5/v$ $\bar{\underline{u}}$ += $a \cdot \bar{\underline{v}}$	2+2 m	m	1+m	1
$v = u^w$	$a = w \cdot v/u\,;\ b = v \cdot \ln(u)$ $\bar{\underline{u}}$ += $a \cdot \bar{\underline{v}}$ $\bar{\underline{w}}$ += $b \cdot \bar{\underline{v}}$	3+3 m	2 m	3+2 m	2
$v = e^u$	$\bar{\underline{u}}$ += $v \cdot \bar{\underline{v}}$	2+2 m	m	m	1
$v = \ln(u)$	$\bar{\underline{u}}$ += $\bar{\underline{v}}/u$	2+2 m	m	m	1
$v = \sin(u)$	$a = \cos(u)$ $\bar{\underline{u}}$ += $a \cdot \bar{\underline{v}}$	2+2 m	m	m	2

Die Algorithmische Differentiation im Rückwärtsmodus ist vor allem interessant für Anwendungen mit einer hohen Anzahl n an Entwurfsparametern und einer vergleichsweise geringen Anzahl m an Ausgangsvariablen. Sie kann daher auf sehr große Probleme mit z. B. mehreren hundert oder tausend Eingangsvariablen effizient angewendet werden. Solche Probleme mit vielen Eingangsgrößen treten bei vielen praktischen Anwendungen bspw. bei Finite-Elemente-Rechnungen oder bei Optimalsteuerungsproblemen auf.

Speicherplatzbedarf

Wie oben beschrieben, ist für die Berechnung der Ableitungen $\bar{x}_i$ die Kenntnis der Zwischenergebnisse v_j notwendig. Während der ersten Phase, der Funktionsauswertung,

müssen daher die Werte aller Zwischenvariablen gespeichert werden, damit sie bei der anschließenden Gradientenberechnung zur Verfügung stehen. Abhängig von dem Aufbau und der Art des Algorithmus zur Berechnung der Funktion f, kann hierfür sehr viel Speicherplatz erforderlich sein.

In vielen Algorithmen werden Variablen im Laufe einer komplexen Berechnung wiederverwendet und somit überschrieben. Für den Rückwärtsmodus werden jedoch auch die „alten“ Werte benötigt, da während der Gradientenberechnung der komplette Berechnungspfad rekonstruiert und rückwärts durchlaufen wird. Treten bspw. Schleifen innerhalb eines Algorithmus auf, so müssen für sämtliche Iterationen die Werte aller innerhalb der Schleife veränderten Variablen gespeichert werden. Der hierfür erforderliche Speicherplatzbedarf kann je nach Anzahl der Schleifendurchläufe sehr hoch ausfallen.

Einige existierende Bibliotheken zur Algorithmischen Differentiation, wie z. B. ADOL-C [GJU96], setzten zur Speicherung des gesamten Berechnungsverlaufs im Rückwärtsmodus ein sogenanntes *Tape* ein. Mit *Tape* wird, in Analogie zu den früher gebräuchlichen Magnetbändern, ein Speicher bezeichnet, der einen rein sequentiellen Zugriff auf die Daten erlaubt. Bei der Funktionsauswertung wird das *Tape* vorwärts in sequentieller Weise mit den Werten der Zwischenvariablen sowie der zur Berechnung verwendeten Rechenoperationen beschrieben. Anschließend können diese Daten rückwärts ausgelesen und die Gradienten rückwärts entlang der Berechnungsgraphen propagiert werden.

Bei kleinen Problemen wird das Tape meist komplett im Arbeitsspeicher des Rechners gehalten, wohingegen es bei großen Problemen in eine Datei auf der Festplatte des Rechners ausgelagert werden muss. Die langsame Zugriffsgeschwindigkeit der Festplatte kann sich dabei schnell als Flaschenhals bei der Gradientenberechnung erweisen. Der Speicherplatzbedarf beim Vorwärtsmodus ist demgegenüber vergleichsweise gering, da direkt nach der Berechnung eines Zwischenergebnisses sofort der entsprechende Gradientenvektor ermittelt werden kann. Anschließend wird der Wert der Zwischenvariablen nicht mehr zu Zwecken der Ableitungsberechnung benötigt. Die Gradienten der Zwischenvariablen besitzen hier die gleiche Lebensdauer wie die Zwischenvariablen selbst.

Als Faustformel gilt, dass der Rückwärtsmodus aus oben genannten Gründen erst ab $n > 10 \ldots 100$ Parametern effizienter ist als der Vorwärtsmodus. Der Rückwärtsmodus hat sich bei den in dieser Arbeit verwendeten Anwendungsbeispielen als ineffizient herausgestellt. Daher wird im Folgenden nur der Vorwärtsmodus betrachtet.

5.3 Implementierungstechniken

Die praktische Anwendung der AD beinhaltet im Wesentlichen zwei Aspekte. Dies ist zum einen das AD-Werkzeug selbst, welches implementiert oder ausgewählt werden muss. Zum anderen muss der Quelltext eines bestehenden, zu differenzierenden Programms angepasst werden, um die Funktionalität des AD-Werkzeuges nutzen zu können.

Existierende Werkzeugimplementierungen arbeiten nach zwei Prinzipien, dem Prinzip des *Operatorüberladens* und der *Codetransformation*.

5.3.1 Operatorüberladung

Werkzeuge, die nach dem Prinzip des Operator-Überladens arbeiten, benutzen die Möglichkeit objektorientierter Programmiersprachen wie C++ oder Fortran 90, die elementaren Rechenoperationen neu zu definieren. Hierzu wird ein neuer Datentyp für die abhängigen, unabhängigen und Zwischenvariablen (aktive Variablen) eingeführt. Die Operatoren dieses Datentyps werden dahingehend modifiziert, dass neben der ursprünglichen Rechenoperation ebenfalls die für die Berechnung der Ableitungen notwendigen Rechenschritte ausgeführt werden. Die Methode des Operator-Überladens erweist sich dahingehend als vorteilhaft, dass im Quelltext lediglich eine Neudefinition der aktiven Variablen[6] vorgenommen werden muss.

Die Wirkungsweise dieser Methode wird im Folgenden anhand der einfachen Implementierung des Vorwärtsmodus beschrieben[7]. Die nachfolgenden vier Punkte sind dabei umzusetzen:

- Definition eines neuen Datentyps (im Folgenden AD-Typ genannt), welcher die Zahlenwerte einer Variablen v und deren Ableitungen $\underline{\dot{v}}$ enthält.
- Definition der arithmetischen Operationen und Funktionen für den Datentyp AD-Typ. Neben der eigentlichen Operation müssen diese ebenfalls die Ableitungen gemäß der Kettenregel berechnen.
- Definition von Funktionen zur Initialisierung des Datentyps zu Beginn sowie zum Auslesen der Gradienten am Ende eines Programmdurchlaufs.
- Weiterhin werden Routinen zur Konvertierung zwischen Fließkommazahlen und Variablen des AD-Typs benötigt. Eine Konvertierung von Fließkommazahlen zu AD-Variablen kann implizit, z. B. über entsprechende Konstruktoren, erfolgen. Sofern keine anderen Werte bekannt sind, wird $\underline{\dot{v}}$ standardmäßig auf den Nullvektor gesetzt.

[6] Aktive Variablen werden von den Eingangsvariablen beeinflusst. Sie ergeben sich aus der Menge der unabhängigen, abhängigen und Zwischenvariablen. Demgegenüber sind numerische Konstanten ein Beispiel für inaktive Variablen.

[7] Die Implementierung des Rückwärtsmodus ist durch die Speicherung des Programmflusses in dem *Tape* aufwändiger. In dieser Arbeit wird daher nicht näher darauf eingegangen, für nähere Informationen sei auf [GJU96] und [WKG05] verwiesen.

Der Quelltext 5.1 zeigt beispielhaft eine Klassendefinition in C++ zur Berechnung der ersten Ableitung im Vorwärtsmodus. Neben dem Variablenwert v enthält die Klasse den Vektor $\underline{\dot{v}}$, ein Array zur Speicherung des Gradienten.

Quelltext 5.1: Definition der AD-Klasse

```
class ad_double {
   double v;          // Wert der Variablen
   double v_p[n];     // Vektor der Ableitungen
};
```

Die Berechnung von $\underline{\dot{v}}$ geschieht direkt bei Aufruf der Operatoren, die hierzu entsprechend neu definiert werden. Ebenso müssen auch die elementaren mathematischen Funktionen aus den Standardbibliotheken überladen werden. Im Quelltext 5.2 ist die Definition des Multiplikationsoperators und der Sinusfunktion für den neuen AD-Typ dargestellt.

Quelltext 5.2: Überladen von Operatoren und Funktionen

```
ad_double ad_double::operator * (ad_double& a) {
   ad_double y;
   y.v = v*a.v;
   for(int i=0; i<n; i++)
      y.v_p[i] = v*a.v_p[i] + v_p[i]*a.v;
   return y;
}

ad_double sin(ad_double& a) {
   ad_double y;
   double tmp;
   y.v = sin(a.v);
   tmp = cos(a.v);
   for(int i=0; i<n; i++)
      y.v_p[i] = tmp*a.v_p[i];
   return y;
}
```

Zur Anwendung des AD-Werkzeuges müssen am Quelltext eines bestehenden Programms einige Änderungen vorgenommen werden:

- Ersetzung der aktiven Variablen durch den neu definierten AD-Typ.
- Initialisierung der Ableitungen der aktiven Variablen. Die Gradienten der unabhängigen Variablen sind bereits zu Beginn bekannt und werden vor dem Start der Berechnung mit Einheitsvektoren initialisiert. Sie stellen den „Keim“ für die Weiterleitung der Gradienten durch das Berechnungsprogramm dar. Die Ableitungen der übrigen aktiven Variablen werden mit Null initialisiert.
- Einfügen von Routinen zum Auslesen der gewünschten Gradienten. Nach Abschluss der Berechnung enthalten alle Variablen neben dem Funktionswert auch die Ableitungswerte. Die Werte der abhängigen Variablen müssen lediglich abgerufen werden.

In den meisten Fällen ist der Algorithmus selbst ohne weitere Anpassungen lauffähig.

Das nachfolgende Programm zeigt die Berechnung des Beispiels (5.8). Der linke Quelltext enthält das ursprüngliche Programm, im rechten Code sind die für die AD notwendigen Änderungen enthalten.

Quelltext 5.3: Ursprüngliches Programm

```
double x1, x2;
double f;

// Initialisierung
x1 = 2.0;
x2 = 3.0;




// Funktionsauswertung
f = sin(2.0*x1)-exp(x1/x2);

// Ergebnis
printf("f = %f\n", f);
```

Quelltext 5.4: AD-Programm

```
ad_double x1, x2;
ad_double f;

// Initialisierung
x1.v = 2.0;
x1.v_p[0] = 1.0; x1.v_p[1] = 0.0;

x2.v = 3.0;
x2.v_p[0] = 0.0; x2.v_p[1] = 1.0;

// Funktionsauswertung
f = sin(2.0*x1)-exp(x1/x2);

// Ergebnis
printf("f = %f\n", f.v);
printf("df/dx1 = %f\n", f.v_p[0]);
printf("df/dx2 = %f\n", f.v_p[1]);
```

Die Methode des Operator-Überladens besitzt nach Bischof und Bücker [BB00] verschiedene Vor- und Nachteile:

+ **Einfachheit:** Die Umsetzung erfordert lediglich die Definition einer AD-Klasse. Die gesamte Komplexität der Gradientenberechnung wird durch diese Klasse vor dem Anwender verborgen.

+ **Flexibilität:** Die Anwendung der AD auf ein bestehendes Programm erfolgt durch den Austausch der Datentypen der aktiven Variablen durch die AD-Klasse. Der eigentliche Quelltext bleibt unverändert. Änderungen an den Ableitungsmechanismen, z. B. die zusätzliche Berechnung der zweiten Ableitung, betreffen nur die AD-Klasse und nicht das zu differenzierende Programm.

– **Transparenz:** Bei Anwendung der AD ändert sich zwar nicht der Quelltext des Programms, jedoch seine Bedeutung wird geändert. Die Bedeutung der einzelnen Operationen des Codes muss mit der zugehörigen Klassendefinition nachvollzogen werden, wodurch das Debuggen des Programms erschwert wird.

– **Kompatibilität:** Viele Computerprogramme sind in prozeduralen Programmiersprachen wie Fortran 77 oder ANSI-C geschrieben, die das Überladen von Operatoren nicht unterstützen. Dies trifft insbesondere auch auf automatisch generierten Code zu, wie er von Regelungstechnik-Werkzeugen zur Modellbeschreibung[8] häufig erzeugt wird. Abwärtskompatible objektorientierte Sprachen wie Fortran 90 und C++ bieten hier einen Ausweg. Erfahrungen zeigen aber, dass trotz der Abwärtskompatibilität meist umfangreiche Anpassungen am Quelltext vorgenommen werden müssen bevor er fehlerfrei übersetzt werden kann.

[8] Z. B. Dymola, CAMeL-View oder der Realtime-Workshop von Mathworks

– **Performanz:** Bei der Operator-Überladung wird anstelle des Aufrufs einer simplen Operation im Grunde eine aufwändigere Unterroutine aufgerufen. Der Overhead für den Sprung in die Unterroutine ist nicht unerheblich.

  Die meisten Compiler wenden bei der Übersetzung eines Programms Optimierungstechniken auf die vorliegenden Anweisungen an. Durch den Sprung wird die eigentliche Funktionalität vor dem Compiler verborgen und manche Compileroptimierungen können nicht mehr angewendet werden.

5.3.2 Codetransformation

Bei der Codetransformation wird das Computerprogramm untersucht und in ein neues Programm umgeschrieben, welches um Anweisungen zur Berechnung der Gradienten erweitert ist. Die Anwendung dieser Methode geschieht durch Einsatz eines Preprozessors, der die notwendigen Erweiterungen und Anpassungen am Quellcode vornimmt.

Die Anwendung der Quellcodetransformation erfordert die folgenden Arbeitsschritte:

- Festlegung von Eingangs- und Ausgangsvariablen.
- Isolierung des aktiven Programmteils:
  Um zum einen ein möglichst kleines, schnelles Programm zu erhalten und zum anderen dem Preprozessor die Arbeit zu erleichtern, sollte der relevante (aktive) Teil des zugrundeliegenden Programms isoliert und möglichst in einer separaten Funktion abgelegt werden. Die Eingangs- und Ausgangsvariablen bilden die Schnittstelle dieser Funktion.
- Aufruf des Preprozessors und Generierung des abgeleiteten Codes:
  Der aktive Teile des Programms wird hier analysiert und ein neues um die Ableitungsvorschriften erweitertes Programm generiert.
- Einbindung des abgeleiteten Codes in einem neuen Programm:
  Da das ursprüngliche Programm nicht für die Verarbeitung der Gradienteninformation vorbereitet ist, muss der Code in einem neuen bzw. angepassten Hauptprogramm eingebunden werden. Hier können die zusätzlich berechneten Gradienten z. B. einem Optimierungsverfahren zur Verfügung gestellt werden. Bezüglich des generierten Codes müssen hier drei Schritte umgesetzt werden. Zuerst wird eine Initialisierung der Eingangsvariablen und deren Ableitungen vorgenommen. Anschließend kann die abgeleitete Funktion aufgerufen werden. Zuletzt erfolgt die Ausgabe bzw. Weiterverarbeitung der Ergebnisse.

Die Codetransformation besitzt gegenüber der im vorangegangenen Abschnitt vorgestellte Operator-Überladung einige Vorteile. So besteht hier die Möglichkeit, die Abhängigkeiten der Variablen untereinander zu analysieren um kleineren und performanteren Code zu erzeugen. Im folgenden werden die wesentlichen Vor- und Nachteile angeführt:

+ **Code-Optimierung:** Die Methode des Operator-Überladens betrachtet nur eine elementare Operation zu einem Zeitpunkt. Demgegenüber besitzt die Quellcodetransformation Zugriff auf komplette Berechnungsvorschriften bzw. Code-Blöcke. Sie besitzt daher eine höhere Flexibilität bei der Anwendung der Ableitungsregeln. So kann bspw. auf der Ebene einzelner Anweisungen der Rückwärtsmodus angewendet

werden, wohingegen die Ableitung im Vorwärtsmodus durch das gesamte Programm propagiert wird (vgl. [BRM97]). Ebenso sind durch die Bearbeitung ganzer Anweisungen oder Anweisungsblöcke weitere Code-Optimierungen möglich.

+ **Kompatibilität:** Die Transformation von Quellcode kann im Prinzip auf eine beliebige Programmiersprache angewendet werden. Das Resultat wird in der Quellsprache dargestellt, wodurch sich keine Kompatibilitätsprobleme ergeben. Optimierungsfunktionen des Compilers haben auf den transformierten Code einen vergleichbaren Effekt wie auf das ursprüngliche Programm.

+ **Transparenz:** Bei der Codetransformation ist gegenüber dem Operator-Überladen die Gradientenberechnung im Quelltext sichtbar. Dem Anwender wird so der Zugriff auf den Code erleichtert, wodurch z. B. die Fehlersuche stark vereinfacht wird.

– **Implementierungsaufwand:** Der Implementierungsaufwand für die Codetransformation ist relativ hoch. Für Hochsprachen wie ANSI-C oder Fortran 77 sind zwar verschiedene erforderliche Compiler-Werkzeugen wie Parser, Code-Manipulatoren und Generatoren erhältlich. Dennoch ist der Aufwand, insbesondere wenn der komplette Sprachumfang abgedeckt werden soll, nicht unerheblich.

– **Code Komplexität:** Ab einem gewissen Komplexitätsgrad des zu transformierenden Programms ist die Quellcodetransformation nicht mehr praktikabel. Bedingt durch die zusätzlichen Anweisungen kann der transformierte Code zu groß werden, um vom Compiler oder von manchen Optimierungsstufen des Compilers verarbeitet zu werden.

5.4 Behandlung von Simulationsprogrammen

In diesem Abschnitt sollen einige Aspekte betrachtet werden, die bei der Anwendung der AD auf Simulationen von mechatronischen Systemen beachtet werden müssen.

Viele Berechnungsprogramme für technische und wissenschaftliche Anwendungen setzen auf numerischen Algorithmen zur Berechnung der gewünschten Ergebnisse auf. Das Berechnungsproblem selbst wird in Form linearer oder nichtlinearer Gleichungs- und Differentialgleichungssysteme beschrieben. Solche Programme bestehen also vereinfacht gesagt aus einer mathematischen Problembeschreibung sowie einem Lösungsalgorithmus, der auf diese Problembeschreibung angewendet wird (siehe Abb. 5.3).

Grundsätzlich existieren zwei Möglichkeiten, die AD auf solche Programme anzuwenden:

- ***Blackbox*-Ansatz:** Aus Sicht des verwendeten AD-Werkzeuges wird bei dieser Vorgehensweise das Programm als *Blackbox* betrachtet, welches aus gegebenen Eingangsgrößen $\underline{u}$ über irgendeinen Rechenweg die Ausgangsgrößen $\underline{y}$ berechnet. Neben der Differentiation der Problembeschreibung wird hier jedoch auch der Lösungsalgorithmus selbst differenziert. In vielen Fällen führt dieses Vorgehen zu den korrekten Ableitungen $\frac{\partial \underline{y}}{\partial \underline{u}^{\mathrm{T}}}$. Eine genaue Betrachtung des verwendeten Lösungsalgorithmus ist jedoch unabdingbar, da die Differentiation des Algorithmus zu Seiteneffekten und somit zu ungenauen oder auch falschen Ergebnissen führen kann.

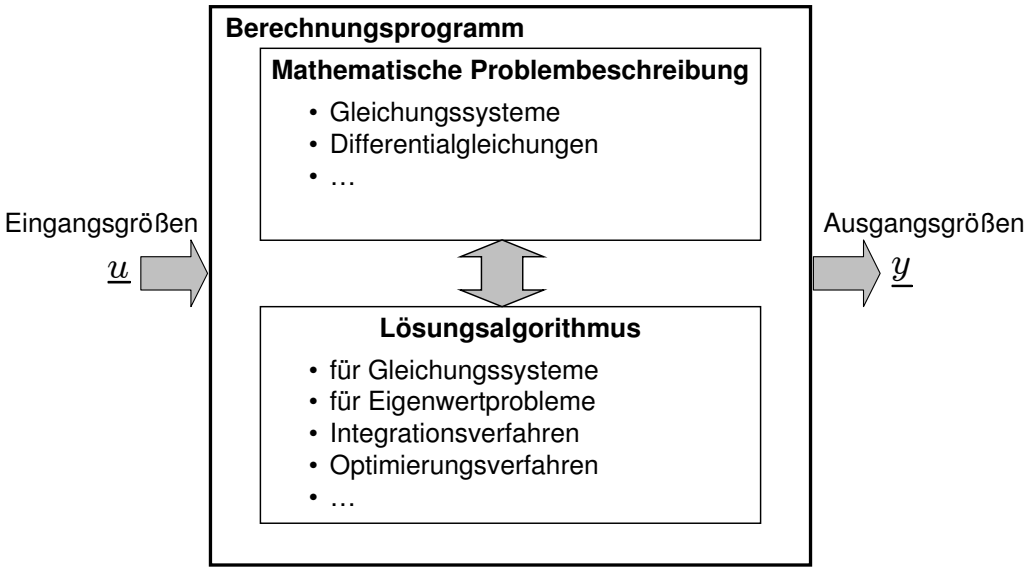


Abbildung 5.3: Vereinfachte Struktur numerischer Berechnungsprogramme

- **Erstellen der Sensitivitätsgleichungen:** Bei dieser Vorgehensweise werden nur die Sensitivitätsgleichungen der Problembeschreibung mit Hilfe der AD erzeugt. Diese werden anschließend gemeinsam mit den ursprünglichen Gleichungen behandelt und gelöst. Jedoch sind hier mehr oder weniger umfangreiche Anpassungen am Lösungsalgorithmus notwendig.

Im Folgenden wird auf die Anwendung der AD auf Programme zur Lösung von Differentialgleichungen, wie sie bei der Simulation mechatronischer Systeme eingesetzt werden, eingegangen. Wie Eberhard und Bischof in [EB99] gezeigt haben, kann der *Black-box*-Ansatz bei Lösungsverfahren, die mit Schrittweitensteuerung oder variabler Ordnung arbeiten, zu überraschenden Ergebnissen führen.[9]

Die hier betrachteten Simulationsmodelle liegen in Form gewöhnlicher Differentialgleichungen vor:

$$\dot{\underline{x}} = \underline{\psi}(\underline{x}, \underline{p}, t) \quad \text{mit} \quad \underline{x}(t = 0) = \underline{x}_0 \tag{5.16}$$

Die Anwendung numerischer Integrationsverfahren auf (5.16) führt zu einer zeitdiskreten, rekursiven Berechnungsvorschrift. Vereinfachend sei im Folgenden angenommen, dass diese die Form eines Einschrittverfahrens (z. B. Euler oder Runge-Kutta) aufweist:

$$\underline{x}_{k+1} = \underline{x}_k + h_k \dot{\underline{\tilde{x}}}_k \,, \quad t_{k+1} = t_k + h_k \tag{5.17}$$

Der Index k bezeichnet dabei den k-ten Integrationsschritt, h_k gibt die aktuell gewählte Schrittweite an und $\dot{\underline{\tilde{x}}}_k$ ist die vom Verfahren geschätzte zeitliche Änderung der Zustandsgrößen. Üblicherweise wird $\dot{\underline{\tilde{x}}}_k$ über mehrfache Auswertungen der Funktion ψ zu verschiedenen Zeitpunkten und Zustandsschätzungen ermittelt. Die einfachste Variante stellt das explizite Euler-Verfahren mit $h_k = h = konstant$ und $\dot{\underline{\tilde{x}}}_k = \dot{\underline{x}}_k = \underline{\psi}(\underline{x}_k, \underline{p}, t)$ dar.

Bei vielen hoch entwickelten Verfahren wird h_k zur Steigerung der Rechengeschwindigkeit über eine Schrittweitensteuerung angepasst. Basierend auf einer Fehlerschätzung wird eine

[9] Bei iterativen Lösungsverfahren für nichtlineare Gleichungssysteme, wie bspw. beim Newton-Verfahren, können im Grunde ähnliche Sachverhalte beobachtet werden (siehe Anhang A.2 und [GBC+93])

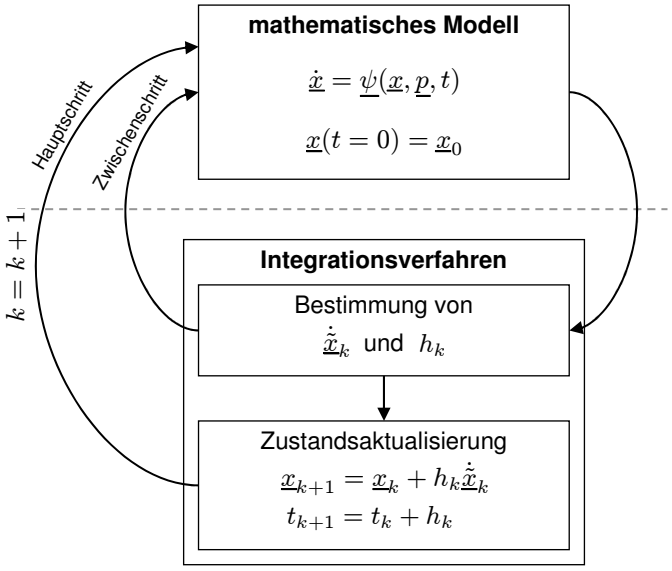


Abbildung 5.4: Numerische Integration gewöhnlicher Differentialgleichungen

möglichst große Schrittweite gewählt ohne eine gegebene Fehlerschranke zu verletzen. Der schematische Ablauf einer solchen Integration ist in Abbildung 5.4 zu sehen.

Für gegebene Parameter $\underline{p}$ können die Ableitungen $\frac{\partial \underline{x}}{\partial \underline{p}^T}\big|_{t=t_{end}}$ zum Simulationsende t_{end} mit Hilfe eines AD-Werkzeuges bestimmt werden. Die Anwendung der AD mit den oben genannten Ansätzen führt dabei entweder auf eine *Differentiation des Modells mitsamt Integrationsalgorithmus* oder auf eine *Integration der Sensitivitätsgleichungen.*

5.4.1 Differentiation des Modells mitsamt Integrationsalgorithmus

Das Simulationsprogramm wird hier als *Blackbox* betrachtet. Die Anwendung der AD erfolgt also auf Modellgleichungen und Integrationsverfahren gleichermaßen. Bei dieser Vorgehensweise können zunächst die folgenden Sachverhalte beobachtet werden:

1. Da die AD nicht den Fluss des Programms selbst verändert, hängt die Zeitdiskretisierung des Integrationsverfahrens ausschließlich von der zugrundeliegenden Differentialgleichung (5.16) ab. Die Genauigkeit der Ableitungen $\frac{\partial \underline{x}}{\partial \underline{p}^T}$ geht nicht in die Schrittweitensteuerung ein und kann somit auch nicht berücksichtigt werden.

2. Bei Verfahren mit variabler Schrittweite hängt im Allgemeinen auch die Integrationsschrittweite h_k von den Parametern $\underline{p}$ ab. Die Schrittweite h wird vom Integrationsverfahren aus den Zuständen $\underline{x}_k$ errechnet, die wiederum von den Parametern $\underline{p}$ beeinflusst werden. Ein AD-Werkzeug wird daher einer Schrittweite h_k auch einen Gradienten ∇h_k zuordnen. Hierdurch entsteht diese Abhängigkeit der Simulationszeit von den Parametern, d. h.: $\nabla t_{k+1} = \nabla t_k + \nabla h_k \neq \underline{0}$.

   Von einem physikalischen Standpunkt aus betrachtet ist diese Abhängigkeit unsinnig, aus Sicht des Berechnungsalgorithmus äußert sich hier jedoch der Zusammenhang zwischen der adaptiven Zeitdiskretisierung und den Parametern: je nach Wahl der Parameter, ändert sich die Zeitdiskretisierung des Integrationsverfahrens. Anstelle

des gewünschten Ergebnisses $\frac{\partial \underline{x}}{\partial \underline{p}^{\mathrm{T}}}\Big|_{t=t_{end}}$ ergibt sich tatsächlich der Gradient:

$$\nabla \underline{x}(t_{end}(\underline{p}), \underline{p}) = \left.\frac{\partial \underline{x}}{\partial t}\right|_{t=t_{end}} \nabla t_{end} + \left.\frac{\partial \underline{x}}{\partial \underline{p}^{\mathrm{T}}}\right|_{t=t_{end}} \tag{5.18}$$

Die Berechnung der gewünschten Lösung $\frac{\partial \underline{x}}{\partial \underline{p}^{\mathrm{T}}}\Big|_{t=t_{end}}$ kann mit einer der folgenden Strategien erreicht werden:

- *Verwendung eines Verfahrens mit fester Schrittweite:* Bei einer festen Schrittweite ist $h_k = h = konstant$ und es gilt $\nabla h_k = 0,\ \forall k$. Es ergibt sich hieraus $\nabla t_{end} = 0$ und somit Gleichung (5.18) zu:

$$\nabla \underline{x}(t_{end}, \underline{p}) = \left.\frac{\partial \underline{x}}{\partial \underline{p}^{\mathrm{T}}}\right|_{t=t_{end}} \tag{5.19}$$

- *Erzwingen von $\nabla t = 0$:* Besteht die Möglichkeit Änderungen am Integrationsalgorithmus selbst vorzunehmen, so kann in jedem Schritt die Unabhängigkeit der Simulationszeit von den Parametern erzwungen werden. Hierbei wird im Grunde so getan, als wäre der Verlauf der Diskretisierungsschrittweiten von vornherein bekannt und somit unabhängig von den Parametern $\underline{p}$, d. h. $\nabla h_k = 0$. Dies ist im Prinzip auch bei Verfahren mit fester Schrittweite der Fall.

- *Nachträgliche Korrektur des Ergebnisses:* Mit Gleichung (5.18) lässt sich die gewünschte Lösung durch Einsetzen von (5.16) im Nachhinein rekonstruieren:

$$\left.\frac{\partial \underline{x}}{\partial \underline{p}^{\mathrm{T}}}\right|_{t=t_{end}} = \nabla \underline{x}(t_{end}) - \underline{\psi}(\underline{x}(t_{end}), \underline{p}, t_{end}) \nabla t_{end} \tag{5.20}$$

  Durch diese Korrektur wird der korrekte Wert des Gradienten hergestellt. Es muss hierbei jedoch beachtet werden, dass der Simulationsverlauf weiterhin physikalisch unplausible Werte beinhaltet.

- *Verwendung fester Ausgabezeitpunkte:* Viele Simulationsprogramme verwenden fest vorgegebene Zeitpunkte t_i zur Ausgabe der Ergebnisse. Ist in einem Schritt $t_k + h_k > t_i$ so wird die Schrittweite auf

$$h_k = t_i - t_k \quad \text{mit} \quad \nabla h_k = -\nabla t_k \tag{5.21}$$

  gesetzt. Die neue Simulationszeit berechnet sich dann zu:

$$t_{k+1} = t_k + h_k \quad \text{mit} \quad \nabla t_{k+1} = \nabla t_k + \nabla h_k = 0 \tag{5.22}$$

  Die Simulation liefert demnach zu den Ausgabezeitpunkten t_i korrekte Ergebnisse. Der selbe Sachverhalt gilt auch, wenn das Simulationsende t_{end} fest vorgegeben wurde, so dass hier ebenfalls eine nachträgliche Korrektur entfallen kann.

  Diese Strategie führt jedoch nur dann zum gewünschten Ergebnis, wenn die Ausgabezeitpunkte vom Integrationsverfahren gezielt angefahren werden. Einige Lösungsalgorithmen schreiten jedoch zunächst über den gewünschten Zeitpunkt hinaus und berechnen die Ausgabewerte $\underline{x}(t_i)$ nachträglich über eine Interpolation. In diesem Fall gilt $\nabla t \neq 0$.

5.4.2 Aufstellen der Sensitivitätsgleichungen

Bei diesem Weg wird zunächst das Modell um Sensitivitätsgleichungen erweitert. Diese ergeben sich durch Ableitung der Differentialgleichung nach den unabhängigen Variablen $\underline{p}$.

$$\frac{\mathrm{d}\dot{\underline{x}}}{\mathrm{d}\underline{p}^{\mathrm{T}}} = \frac{\mathrm{d}}{\mathrm{d}\underline{p}^{\mathrm{T}}}\left(\frac{\mathrm{d}\underline{x}}{\mathrm{d}t}\right) = \frac{\partial\underline{\psi}}{\partial\underline{x}^{\mathrm{T}}}\frac{\mathrm{d}\underline{x}}{\mathrm{d}\underline{p}^{\mathrm{T}}} + \frac{\partial\underline{\psi}}{\partial\underline{p}^{\mathrm{T}}} \tag{5.23}$$

Durch Vertauschen der Differenzierungsreihenfolge erhalten wir eine neue Differentialgleichung für die Sensitivitäten $\nabla\underline{x}$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\mathrm{d}\underline{x}}{\mathrm{d}\underline{p}^{\mathrm{T}}}\right) = [\dot{\nabla\underline{x}}] = \frac{\partial\underline{\psi}}{\partial\underline{x}^{\mathrm{T}}}\nabla\underline{x} + \frac{\partial\underline{\psi}}{\partial\underline{p}^{\mathrm{T}}} \tag{5.24}$$

Mit einem geeigneten Integrationsalgorithmus lässt sich diese Sensitivitätsgleichung gemeinsam mit (5.16) numerisch lösen und man erhält die gewünschten Ableitungen.

Differenziert wird nur der Teil des Programms, der die Modellgleichungen beinhaltet. Das Integrationsverfahren selbst wird bei der AD nicht berücksichtigt. Zur Behandlung der Sensitivitätsgleichungen sind jedoch Anpassungen am Integrationsalgorithmus notwendig. Die Sensitivitäten $\nabla\underline{x}$ müssen erfasst, d. h. aus Variablen des AD-Typs ausgelesen, und gleichberechtigt neben den Zuständen $\underline{x}$ für die Integration und Schrittweitenbestimmung herangezogen werden. Die Genauigkeit der Ableitungen liegt innerhalb des Toleranzbereichs des eingesetzten Lösungsverfahrens.

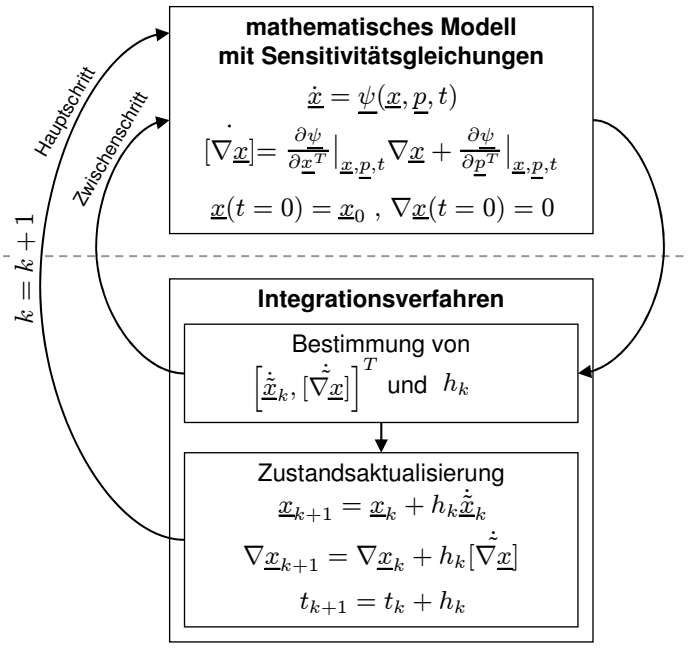


Abbildung 5.5: Numerische Integration der Sensitivitätsgleichungen

5.4.3 AD von Blockdiagrammen

Viele Entwurfswerkzeuge zur Modellierung, Analyse und Synthese regelungstechnischer Systeme nutzen eine Modellbeschreibung in Form von Blockdiagrammen. Häufig wird für eine Simulation aus dem Blockdiagramm ein lauffähiges Computerprogramm erzeugt, welches die Differentialgleichungen an ein entsprechendes Lösungsverfahren anbindet. Eine Codegenerierung überführt die Blockdiagramme in mathematische Gleichungen, die anschließend als Quelltext, z. B. in einer Hochsprache wie C, C++ oder Fortran, abgelegt werden. Der Quelltext kann mit handelsüblichen Compilern in ein ausführbares Programm übersetzt werden.

Die Ableitungen können bspw. durch Anwendung von AD-Werkzeugen auf den Quellcode nach dem Prinzip des *Operatorüberladens* oder der *Codetransformation* erzeugt werden. Bei dieser Vorgehensweise gilt es jedoch die in den beiden vorangegangenen Abschnitten genannten Fallstricke zu beachten. Ist das Lösungsverfahren Teil des Quellcodes müssen die in 5.4.1 angeführten numerischen Probleme berücksichtigt werden. Wird die AD zur Generierung der Sensitivitätsgleichungen genutzt, so sind Änderungen am Lösungsalgorithmus notwendig, um neben den Differentialgleichungen des Modells auch die Sensitivitätsgleichungen zu integrieren (vgl. Abschnitt 5.4.2).

In diesem Abschnitt wird mit der *Modelltransformation* eine weitere Implementierungstechnik für eine direkte Differentiation regelungstechnischer Blockdiagramme vorgestellt. Die oben genannten Probleme werden durch die Differentiation auf Modellebene vermieden. Diese Form der AD wurde in [MÇT07] für das CAE-Werkzeug CAMeL-View [Hah99, iXt01a] vorgestellt. Des weiteren existiert mit Diffedge ein kommerzielles Tool für die AD von Simulink-Modellen [MC03].

Betrachtet man sich den Berechnungsgraphen eines Computerprogramms, z. B. den Graphen der Beispielfunktion in Abbildung 5.1, so fällt eine große Ähnlichkeit zu regelungstechnischen Blockdiagrammen auf. Blockdiagramme können im Prinzip als ein Berechnungsgraph aufgefasst werden: die Blöcke bilden die elementaren mathematischen Funktionen, die Verbindungen zwischen Blöcken stellen die Kanten des Berechnungsgraphen dar. Die Differentiation eines Blockdiagramms besteht aus zwei Transformationsschritten:

1. Differentiation der elementaren Blöcke

2. Verknüpfung der Sensitivitätsgleichungen

Das Blockdiagramm bildet den Berechnungsgraphen der Systemfunktion $\underline{\psi}(\underline{x}, \underline{p}, t)$ zu einem Zeitpunkt t ab. Über die Transformation wird die Systemfunktion $\underline{\psi}(\underline{x}, \underline{p}, t)$ im Vorwärtsmodus nach $\underline{p}$ differenziert und man gelangt zu der Sensitivitätsgleichung (5.24).

Differentiation der elementaren Blöcke

Im ersten Transformationsschritt werden die Blöcke des Diagramms unabhängig voneinander betrachtet. Durch Anwendung der Differentiationsregeln können die Ableitungen elementarer Blöcke leicht bestimmt werden. Die Idee besteht nun darin, die ursprünglichen Blöcke um diese Sensitivitätsgleichungen zu erweitern und wieder in das Blockdiagramm einzusetzen. Diese Erweiterung ist in Abbildung 5.6 am Beispiel einer nichtlinearen Funktion $y = f(u, \underline{p}, t)$ für einen Parametervektor $\underline{p}$ der Größe n dargestellt.

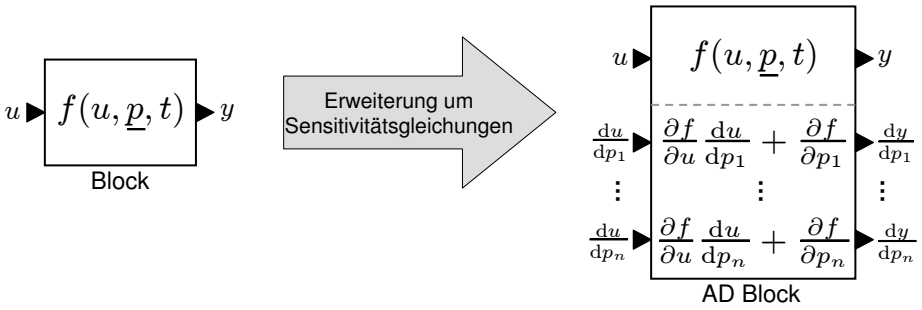
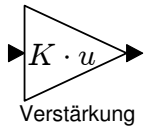
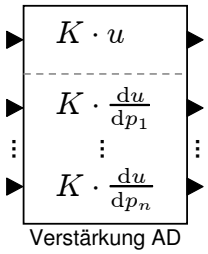
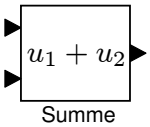
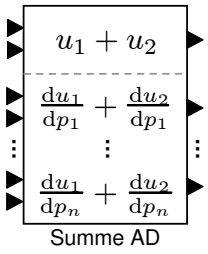
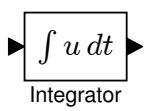
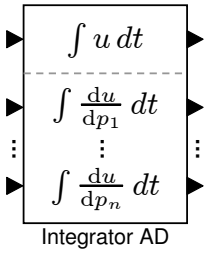


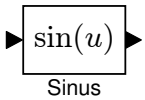
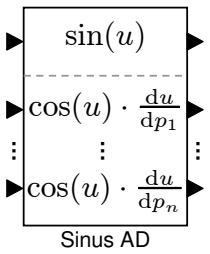
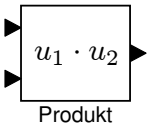
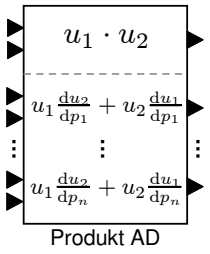
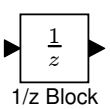
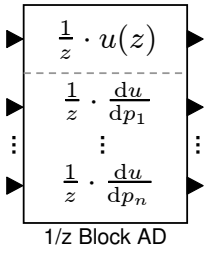
Abbildung 5.6: Erweiterung eines Blocks um Sensitivitätsgleichungen

Für die Anbindung der Sensitivitätsgleichungen an das Modell muss eine Erweiterung der Schnittstelle vorgenommen werden. Neben den ursprünglichen Ein- und Ausgängen $\underline{u}$ und $\underline{y}$ werden neue Ein- und Ausgänge für die Gradienten $\mathrm{d}\underline{u}/\mathrm{d}\underline{p}^{\mathrm{T}}$ und $\mathrm{d}\underline{y}/\mathrm{d}\underline{p}^{\mathrm{T}}$ hinzugefügt. In Tabelle 5.6 sind die AD-Transformationen für einige elementare Blöcke zu sehen.

Blöcken mit Zuständen kommt eine besondere Rolle zu. Das Blockdiagramm bildet den Berechnungsgraphen nur für einen konkreten Zeitpunkt t ab. Die Informationsübertragung von einem Zeitpunkt t zum nächsten Zeitpunkt $t + h$ geschieht über die Zustände eines Modells. Im Wesentlichen finden sich Zustände in den Integratoren und, im Falle eines zeitdiskreten Modells, in den $1/z$-Blöcken. Die Integration der Sensitivitäten $\nabla\underline{x}$ geschieht

Tabelle 5.6: AD-Transformation für einige elementare Blöcke

Block	Block erweitert um n Sensitivitätsgleichungen
	
	
	

Block	Block erweitert um n Sensitivitätsgleichungen
	
	
	

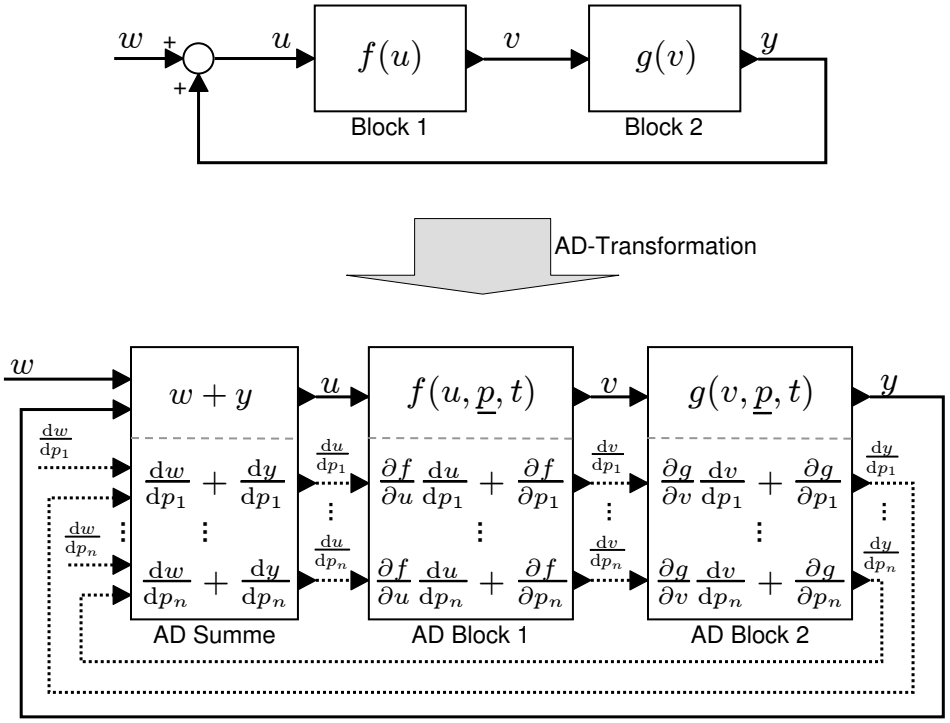


Abbildung 5.7: Algorithmische Differentiation eines Blockdiagramms

unabhängig voneinander. Die Zustände von Integratorblöcken werden daher dupliziert, so dass parallel zu den zeitlichen Ableitungen $\dot{\underline{x}}$ ebenfalls die zeitlichen Ableitungen der Sensitivitätsgleichungen $[\dot{\nabla}\underline{x}]$ integriert werden. Analog hierzu verfährt man mit den $1/z$-Blöcken (siehe Integrator und $1/z$-Block in Tabelle 5.6).

Verknüpfung der Sensitivitätsgleichungen

Im zweiten Transformationsschritt werden die Ein- und Ausgänge der Gradienten $\mathrm{d}\underline{u}/\mathrm{d}\underline{p}^{\mathrm{T}}$ und $\mathrm{d}\underline{y}/\mathrm{d}\underline{p}^{\mathrm{T}}$ miteinander verkoppelt. Die Verkopplung verläuft parallel zu den ursprünglichen Verbindungen der Ein- und Ausgänge. Ist z. B. der Ausgang y mit dem Eingang u verbunden, so werden auch die Gradienten $\mathrm{d}y/\mathrm{d}\underline{p}^{\mathrm{T}}$ und $\mathrm{d}u/\mathrm{d}\underline{p}^{\mathrm{T}}$ elementweise miteinander verknüpft.

In Bild 5.7 ist diese Verkopplung dargestellt. Die Abbildung zeigt oben ein Beispielmodell bestehend aus zwei Blöcken und einer Summationsstelle. Unten ist das Ergebnis der Transformation dargestellt. Die gepunkteten Linien kennzeichnen diese neu erstellten Gradientenverbindungen.

Diese AD-Technik ist vom Prinzip her eine Codetransformation und weist entsprechende Vor- und Nachteile auf (vgl. Abschnitt 5.3.2). Anstelle von Quellcode in einer Programmiersprache liefert sie jedoch ein Blockdiagramm, welches im Entwicklungswerkzeug eingeladen und weiterverarbeitet werden kann. Hierdurch ergeben sich gegenüber Techniken, die auf einem aus dem Modell generierten Quellcode aufsetzen, die folgenden Vor- und Nachteile:

+ **Kompatibilität:** Die transformierten Modelle stellen wieder kompatible Modelle dar. Die Sensitivitätsgleichungen stehen gleichberechtigt neben den Gleichungen des

ursprünglichen Modells. Die Modelle erfordern daher keine Änderungen an der Entwicklungsumgebung, wie Modifikationen an der Codegenerierung, dem Solver oder den Analyse- und Visualisierungswerkzeugen.

+ **Wiederverwendbarkeit:** Die transformierten Modelle können im Entwurfswerkzeug weiterverarbeitet werden. Der Anwender erhält so Zugriff auf das Modell und ist in der Lage, das transformierte Modell auf Fehler zu überprüfen, weitere Anpassungen vorzunehmen oder das Modell im Zusammenhang mit anderen Modellen weiter zu verwenden.

+ **Handhabung:** Der Anwender arbeitet wie gewohnt in seiner grafischen Entwicklungsumgebung. Für die Anwendung der AD-Transformation sind keine Programmierkenntnisse nötigt.

+ **Echtzeit:** Die evtl. vorhandene Echtzeitfähigkeit eines Modell bleibt erhalten.

+ **Performanz:** Einige Werkzeuge analysieren die Modellgleichungen und optimieren den Quellcode bei der Codegenerierung, wie z. B. Dymola [OEC96]. Bei der Modelltransformation werden auch die Sensitivitätsgleichungen in eine solche Codeoptimierung einbezogen.

  Einige komplexere Blöcke oder Funktionen des Modells, wie z. B. Blöcke für das Lösen linearer Gleichungssysteme, können bei der Modelltransformation gesondert behandelt werden. Hierdurch lässt sich zum einen die Konvergenz der Lösungsverfahren garantieren, zum anderen kann der Berechnungsaufwand erheblich reduziert werden (vgl. Anhang A.2 sowie [GBC+93, Gri00]).

– **Flexibilität**: Die Modelltransformation liefert ein Modell fester Dimension. Bei den Optimierungsverfahren aus Kapitel 3 wechseln sich die Suchrichtungsbestimmung, bei der die Gradienten benötigt werden, und Liniensuche, bei der nur die Funktionswerte gebraucht werden, ab. Die Methode des *Operatorüberladens* ermöglicht es, die Anzahl der unabhängigen Variablen zur Laufzeit zu variieren; ebenso lassen sich mit ihr prinzipiell auch Ableitungen höherer Ordnung mit dem selben Quellcode berechnen.

5.4.4 Dynamik der Sensitivitätsgleichungen

Die Erweiterung des Modells um die Sensitivitätsgleichungen führt zu zusätzlichen Differentialgleichungen, die eine eigene, zunächst einmal unbekannte Dynamik aufweisen. Die Anwendung eines Integrationsverfahrens, welches bei den ursprünglichen Modellgleichungen ein stabiles Verhalten zeigt und zu korrekten Ergebnissen führt, kann bei der Lösung der Sensitivitätsgleichungen prinzipiell zu unbrauchbaren Ergebnissen führen. Für Integrationsverfahren mit fester Schrittweite kann ein Stabilitätsbereich angegeben werden, mit dem die Anwendbarkeit des Verfahrens auf ein bestimmtes Problem überprüft werden kann. Das Verfahren weist bei einem linearen Modell dann ein stabiles Verhalten auf, wenn die Eigenwerte der Systemmatrix innerhalb dieses Stabilitätsbereiches liegen. Es ist daher wünschenswert eine generelle Aussage darüber treffen zu können, wie sich die Eigenwerte der ursprünglichen Modellgleichungen auf die Eigenwerte der Sensitivitätsgleichungen abbilden. In [MT06] wurde gezeigt, dass die Eigenwerte eines ursprünglichen linearen Modells und die Eigenwerte der zugehörigen Sensitivitätsgleichungen deckungsgleich sind.

Ausgangspunkt dieser Betrachtungen ist ein lineares, zeitinvariantes Differentialgleichungssystem, welches in Zustandsraumdarstellung vorliegt. Angenommen, die Koeffizienten der Zustandsraummatrizen lassen sich über einen Parameter p modifizieren, ergibt sich die lineare Zustandsraumdarstellung:

$$\begin{aligned} \dot{\underline{x}} &= \underline{\mathbf{A}}(p)\,\underline{x} + \underline{\mathbf{B}}(p)\,\underline{u} \\ \underline{y} &= \underline{\mathbf{C}}(p)\,\underline{x} + \underline{\mathbf{D}}(p)\,\underline{u} \end{aligned} \tag{5.25}$$

mit dem Anfangszustand $\underline{x}_0$.

Die Sensitivitätsgleichung erhält man durch Anwenden der Differentiationsregeln auf die Zustandsraumdarstellung. Die Differentiation nach dem unabhängigen Parameter p ergibt:

$$\begin{aligned} [\dot{\nabla \underline{x}}] &= \frac{\partial}{\partial p}\underline{\mathbf{A}}\,\underline{x} + \underline{\mathbf{A}}\,\nabla\underline{x} + \frac{\partial}{\partial p}\underline{\mathbf{B}}\,\underline{u} \\ \frac{\mathrm{d}\underline{y}}{\mathrm{d}p} &= \frac{\partial}{\partial p}\underline{\mathbf{C}}\,\underline{x} + \underline{\mathbf{C}}\,\nabla\underline{x} + \frac{\partial}{\partial p}\underline{\mathbf{D}}\,\underline{u} \end{aligned} \tag{5.26}$$

In vielen Fällen ist der Anfangszustand $\underline{x}_0$ konstant und somit unabhängig vom Parameter p. Hieraus folgt für den Anfangszustand der Sensitivitäten: $\nabla\underline{x}_0 = \underline{0}$.

Fasst man die ursprünglichen Modellgleichungen und die Sensitivitätsgleichungen zusammen, erhält man ein erweitertes lineares Differentialgleichungssystem. Dieses besitzt die Form:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\left[\begin{array}{c}\underline{x}\\\hline \nabla\underline{x}\end{array}\right] &= \left[\begin{array}{c|c}\underline{\mathbf{A}} & \underline{\mathbf{0}}\\\hline \frac{\partial}{\partial p}\underline{\mathbf{A}} & \underline{\mathbf{A}}\end{array}\right]\left[\begin{array}{c}\underline{x}\\\hline \nabla\underline{x}\end{array}\right] + \left[\begin{array}{c}\underline{\mathbf{B}}\\\hline \frac{\partial}{\partial p}\underline{\mathbf{B}}\end{array}\right]\underline{u} \\ \left[\begin{array}{c}\underline{y}\\\hline \nabla\underline{y}\end{array}\right] &= \left[\begin{array}{c|c}\underline{\mathbf{C}} & \underline{\mathbf{0}}\\\hline \frac{\partial}{\partial p}\underline{\mathbf{C}} & \underline{\mathbf{C}}\end{array}\right]\left[\begin{array}{c}\underline{x}\\\hline \nabla\underline{x}\end{array}\right] + \left[\begin{array}{c}\underline{\mathbf{D}}\\\hline \frac{\partial}{\partial p}\underline{\mathbf{D}}\end{array}\right]\underline{u} \end{aligned} \tag{5.27}$$

Dieses Differentialgleichungssystem lässt sich leicht auf den vektoriellen Fall mit dem unabhängigen Parametervektor $\underline{p}$ erweitern. Die Differentiation von (5.25) nach jedem Element p_i des Parametervektors führt auf die erweiterte lineare Zustandsraumdarstellung:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\left[\begin{array}{c}\underline{x}\\\hline \nabla_{p_1}\underline{x}\\ \vdots \\ \nabla_{p_n}\underline{x}\end{array}\right] &= \left[\begin{array}{c|ccc}\underline{\mathbf{A}} & \underline{\mathbf{0}} & \cdots & \underline{\mathbf{0}}\\\hline \frac{\partial}{\partial p_1}\underline{\mathbf{A}} & \underline{\mathbf{A}} & \cdots & \underline{\mathbf{0}}\\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial}{\partial p_n}\underline{\mathbf{A}} & \underline{\mathbf{0}} & \cdots & \underline{\mathbf{A}}\end{array}\right]\left[\begin{array}{c}\underline{x}\\\hline \nabla_{p_1}\underline{x}\\ \vdots \\ \nabla_{p_n}\underline{x}\end{array}\right] + \left[\begin{array}{c}\underline{\mathbf{B}}\\\hline \frac{\partial}{\partial p_1}\underline{\mathbf{B}}\\ \vdots \\ \frac{\partial}{\partial p_n}\underline{\mathbf{B}}\end{array}\right]\underline{u} \\ \left[\begin{array}{c}\underline{y}\\\hline \nabla_{p_1}\underline{y}\\ \vdots \\ \nabla_{p_n}\underline{y}\end{array}\right] &= \left[\begin{array}{c|ccc}\underline{\mathbf{C}} & \underline{\mathbf{0}} & \cdots & \underline{\mathbf{0}}\\\hline \frac{\partial}{\partial p_1}\underline{\mathbf{C}} & \underline{\mathbf{C}} & \cdots & \underline{\mathbf{0}}\\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial}{\partial p_n}\underline{\mathbf{C}} & \underline{\mathbf{0}} & \cdots & \underline{\mathbf{C}}\end{array}\right]\left[\begin{array}{c}\underline{x}\\\hline \nabla_{p_1}\underline{x}\\ \vdots \\ \nabla_{p_n}\underline{x}\end{array}\right] + \left[\begin{array}{c}\underline{\mathbf{D}}\\\hline \frac{\partial}{\partial p_1}\underline{\mathbf{D}}\\ \vdots \\ \frac{\partial}{\partial p_n}\underline{\mathbf{D}}\end{array}\right]\underline{u} \end{aligned} \tag{5.28}$$

Betrachtet man die Systemmatrix des erweiterten Differentialgleichungssystems (5.27) bzw. (5.28) im vektoriellen Fall, so fällt auf, dass die Ableitungen $\frac{\partial}{\partial p_i}\underline{\mathbf{A}}$ im linken unteren Quadranten keinerlei Einfluss auf die Lage der Eigenwerte ausüben. Diese werden

ausschließlich von der Matrix $\underline{\mathbf{A}}$ auf der Diagonalen der Systemmatrix bestimmt. Die Eigenwerte des ursprünglichen Modells werden durch die Erweiterung um die Sensitivitätsgleichungen $n-$mal dupliziert. Die Sensitivitätsgleichungen besitzen demnach keinerlei Einfluss auf die maximale Integrationsschrittweite eines numerischen Integrationsverfahrens, d. h. die Simulation des erweiterten Modells kann mit den gleichen Integrationseinstellungen, wie für das ursprüngliche Modell, durchgeführt werden.

Dieser Sachverhalt trifft nur auf lineare Gleichungssysteme zu. Für Systeme mit nicht allzu stark ausgeprägten nichtlinearen Eigenschaften ist jedoch ein ähnliches Verhalten zu erwarten.

5.4.5 Linearisierung von nichtlinearen dynamischen Modellen

Die Linearisierung von dynamischen nichtlinearen Differentialgleichungssystemen hat eine große Bedeutung, da man hierüber die nichtlineare Systembeschreibung auf eine lineare Zustandsraumdarstellung zurückführen kann. Die große Bandbreite an Verfahren der linearen Systemtheorie steht so zu einem gewissen Grad auch den nichtlinearen Systemen zur Verfügung. Ein nichtlineares Modell der Form:

$$\begin{aligned} \dot{\underline{x}} &= \underline{\psi}(\underline{x}, \underline{u}, t) \\ \underline{y} &= \underline{\xi}(\underline{x}, \underline{u}, t) \end{aligned} \tag{5.29}$$

kann durch die Linearisierung um einen Arbeitspunkt $\underline{u}_0$, $\underline{x}_0$ und $\underline{y}_0$ in ein lineares Zustandsraummodell überführt werden, die in der näheren Umgebung des Arbeitspunktes das Verhalten des Systems mit geringer Abweichung wiedergibt.

Das linearisierte Modell beschreibt demnach die Abweichung des Systems vom gewählten Arbeitspunkt. Die Abweichungen $\Delta\underline{u}$, $\Delta\underline{x}$ und $\Delta\underline{y}$ können über:

$$\begin{aligned} \underline{u} &= \underline{u}_0 + \Delta\underline{u} \\ \underline{x} &= \underline{x}_0 + \Delta\underline{x} \\ \underline{y} &= \underline{y}_0 + \Delta\underline{y} \end{aligned} \tag{5.30}$$

angegeben werden.

Die Linearisierung lässt sich über eine Taylor-Reihenentwicklung mit Abbruch nach dem ersten Glied herleiten (siehe hierzu [Föl94, Har02]). Durch die partielle Differentiation der Zustands- und Ausgangsgleichung ψ und ξ nach dem Zustands- und Eingangsvektor $\underline{x}$ und $\underline{u}$ erhält man die Jacobi-Matrizen $\underline{\mathbf{A}}$, $\underline{\mathbf{B}}$, $\underline{\mathbf{C}}$ und $\underline{\mathbf{D}}$:

$$\begin{aligned} \underline{\mathbf{A}} &= \left.\frac{\partial \underline{\psi}}{\partial \underline{x}}\right|_{\substack{\underline{x}=\underline{x}_0 \\ \underline{u}=\underline{u}_0}} & \underline{\mathbf{B}} &= \left.\frac{\partial \underline{\psi}}{\partial \underline{u}}\right|_{\substack{\underline{x}=\underline{x}_0 \\ \underline{u}=\underline{u}_0}} \\ \underline{\mathbf{C}} &= \left.\frac{\partial \underline{\xi}}{\partial \underline{x}}\right|_{\substack{\underline{x}=\underline{x}_0 \\ \underline{u}=\underline{u}_0}} & \underline{\mathbf{D}} &= \left.\frac{\partial \underline{\xi}}{\partial \underline{u}}\right|_{\substack{\underline{x}=\underline{x}_0 \\ \underline{u}=\underline{u}_0}} \end{aligned} \tag{5.31}$$

Die Differentialgleichung des linearisierten Modells erhält man schließlich über die Zustandsraumdarstellung:

$$\begin{aligned} [\dot{\Delta\underline{x}}] &= \underline{\mathbf{A}}\,\Delta\underline{x} + \underline{\mathbf{B}}\,\Delta\underline{u} \\ \Delta\underline{y} &= \underline{\mathbf{C}}\,\Delta\underline{x} + \underline{\mathbf{D}}\,\Delta\underline{u} \end{aligned} \tag{5.32}$$

Die Umrechnung auf die ursprünglichen Ein-, Ausgangs- und Zustandsgrößen $\underline{u}$, $\underline{y}$ und $\underline{x}$ geschieht über die Gleichungen (5.30).

Die Berechnung der Jacobi-Matrizen in (5.32) kann mit Hilfe der *Algorithmischen Differentiation* oder über *Differenzenquotienten* erfolgen. Beide Methoden haben dabei die bereits genannten spezifischen Vor- und Nachteile:

Die Anwendung der AD führt zu exakt berechneten Jacobi-Matrizen. Demgegenüber liefern Differenzenquotienten nur eine Näherung der Lösung. Sie lassen sich jedoch sehr einfach implementieren und können auch auf Black-Box-Funktionen angewendet werden.

Im Normalfall sind exakte Ergebnisse bei technischen Problemen sehr zu begrüßen. In diesem Zusammenhang muss sich der Anwender jedoch selbst die Frage beantworten, ob er eine exakte Linearisierung überhaupt durchführen will. Bei der Differentiation für z. B. ein Optimierungsproblem sind exakte Ableitungen für die Konvergenz des Verfahrens unabdingbar. Bei der Linearisierung hingegen soll das physikalische Verhalten eines Systems in einem bestimmten Arbeitsbereich wiedergegeben werden. Ein Konvergenzproblem wie bei einer Optimierung liegt hier nicht vor.

Es gibt zwei Argumente, weshalb Differenzenquotienten zur Linearisierung besser geeignet sein können.

- *Differenzenquotienten verhalten sich bei gut linearisierbaren Modellen gutmütig:* Bei der Linearisierung eines nichtlinearen Systems geht man implizit davon aus, dass sich das System in der Umgebung des Arbeitspunktes näherungsweise linear verhält. Die Differenzenquotienten verwenden als Ansatzfunktionen Polynome. Das näherungsweise lineare Verhalten des Ausgangsmodells kann über diese Polynome sehr genau nachgebildet werden. Geht man sogar von einem exakt linearem Verhalten in der Umgebung des Arbeitspunktes aus, so liefern die Differenzenquotienten, sieht man von Auslöschungseffekten ab, auch exakte Ergebnisse.

  Je schwächer ausgeprägt die Nichtlinearitäten im Ausgangsmodell sind, desto genauer kann es über Differenzenquotienten linearisiert werden und desto größer können die Schrittweiten gewählt werden, wodurch wiederum die Auslöschungseffekte reduziert werden.

- *Differenzenquotienten besitzen ein mittelndes Verhalten:* Bei der Linearisierung über die AD kann es bei Unstetigkeiten zu nicht vorhersagbaren Effekten kommen. Wird der Arbeitspunkt direkt auf eine Unstetigkeiten gelegt, ist es von Implementierungsdetails abhängig, was die Differentiation liefert. Differenzenquotienten erreichen hier eine Mittelung und bilden, eine geeignete Schrittweite vorausgesetzt, das prinzipielle Verhalten ab.

  Z. B. kommt es bei Hydraulikventilen aufgrund der Steuerkanten bei der Nullstellung des Steuerschiebers zu einer Umschaltung der Volumenströme. Eine Linearisierung um die Mittelstellung mit der AD führt hier meist zu unbrauchbaren Ergebnissen.

In Abbildung 5.8 ist beispielhaft der Volumenstrom eines Ventils über die normierte Steuerschieberstellung angetragen. Das Ventil besitzt eine positive Überdeckung und ein asymmetrisches Verhalten bzgl. positiver und negativer Steuerschieberstellung. Die exakte Differentiation um den Nullpunkt mittels AD liefert hier den absolut unbrauchbaren Gradienten von 0. Die beiden Differenzenquotienten mitteln den Gradienten über den Arbeitsbereich und geben zumindest das grundsätzliche Verhalten des Ventils wieder. Die

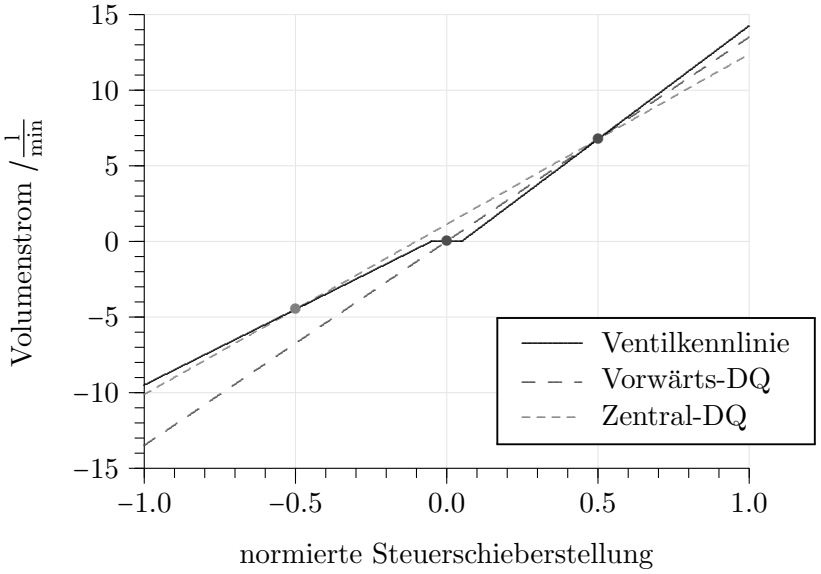


Abbildung 5.8: Ventilkennlinie bei positiver Überdeckung der Steuerkanten

Totzone in der Mittelstellung kann durch das lineare Modell prinzipbedingt nicht abgebildet werden.

Liefert ein über Differenzenquotienten linearisiertes Modell keine brauchbaren Ergebnisse, so ist dies häufig nicht auf die Verwendung von Differenzenquotienten zurückzuführen. Es muss hier vielmehr die Frage gestellt werden, ob eine Linearisierung bei diesem Modell überhaupt zulässig ist.

5.4.6 Differentiation von Punkt- und abschnittsweise bewertenden Kriterien

Die in Kapitel 4.3.2 beschriebenen Punkt- und abschnittsweise bewertenden Zielkriterien, beurteilen das System anhand einzelner Zeitpunkte bzw. Zeitabschnitte. Eine Besonderheit dieser Kriterien ist, dass die Verschiebungen der Zeitpunkte bzw. die Veränderungen der Zeitabschnitte in die Zielgrößen einfließen. Diese Verschiebungen lassen sich bei der AD der Differentialgleichungen nicht ohne Weiteres berücksichtigen. Dies liegt in der Arbeitsweise der Integrationsverfahren begründet.

Die Integrationsverfahren zur Lösung gewöhnlicher Differentialgleichungen generieren eine Folge von Zeitpunkten t_k, zu denen das Modell ausgewertet wird (vgl. Seite 64). Die Wahl dieser Zeitpunkte ist stark von der verwendeten Schrittweitensteuerung des Integrationsverfahrens abhängig. Bei der Differentiation des Simulationsprogramms können zwei Fälle auftreten (vgl. Abschnitt 5.4.1 und 5.4.2):

- $\nabla t_k = 0$: Die Zeitpunkte sind nicht von den Entwurfsparametern abhängig. Dies ist bspw. bei der Integration der Sensitivitätsgleichungen oder bei der Verwendung fester Schrittweiten der Fall. Weiterhin kann über geeignete Maßnahmen für eine Unabhängigkeit der Zeitpunkte gesorgt werden (siehe Seite 66).

- $\nabla t_k \neq 0$: Aufgrund der AD des Integrationsverfahrens besitzen die Zeitpunkte eine Ableitung. Diese Ableitungen liegen in der Schrittweitensteuerung begründet und

können auch nur in diesem Zusammenhang interpretiert werden. Werden diese Ableitungen für die Differentiation der Kriterien herangezogen, so führt dies zu falschen Gradienten[10].

Die Berechnung der Kriterien darf sich nicht alleine auf die Integrationszeitpunkte t_k stützen. Dies gilt vor allem für die *Triggerfunktion* (4.6), da in den Funktionswert dieses Kriteriums lediglich der Zeitpunkt eingeht. Die folgende Betrachtung von Punkt- und abschnittsweise bewertenden Zielkriterien wird daher am Beispiel dieser Funktion untersucht. Es wird dabei von $\nabla t_k = 0$ ausgegangen, da so definierte Verhältnisse vorliegen.

Die *Triggerfunktion* ist in Gleichung (4.6) zeitkontinuierlich angegeben:

$$f_{hit} = \min\{t \mid y(t) - \bar{y} = 0\}$$

Eine Eins-zu-eins-Umsetzung dieser Funktion wird ein unbrauchbares Ergebnis liefern, da die diskreten Simulationszeitpunkte t_k mit dem Zeitpunkt t_{hit}, bei dem die Gleichheitsbedingung $y(t_{hit}) = \bar{y}$ exakt erfüllt ist, nur in Ausnahmefällen übereinstimmen.

Aus diesem Grund wird man bei Simulationen auf eine zeitdiskrete Formulierung, wie:

$$\hat{f}_{hit} = \min\{t_k \mid \operatorname{sgn}(y_k - \bar{y}) \neq \operatorname{sgn}(y_{k-1} - \bar{y})\} \tag{5.33}$$

zurückgreifen. Diese Funktion detektiert den Zeitpunkt anhand eines Vorzeichenwechsels. Sie liefert eine Zielgröße, die grundsätzlich zur Bewertung eines Systems geeignet ist. Differenziert man diese Funktion, so erhält man wegen $\nabla t_k = \underline{0}$ jedoch $\nabla \hat{f}_{hit} = \underline{0}$.

Die Ursache hierfür ist nicht bei der AD zu suchen; diese liefert den korrekten Gradienten für diese Funktion. Dies kann man sich verdeutlichen, wenn man die Systemausgänge y_k und y_{k-1} im „Kleinen“ auslenkt. Die Funktion (5.33) ändert hierdurch nicht ihren Wert und der Gradient verschwindet daher folgerichtig (siehe auch Abbildung 5.9). Vielmehr muss die Funktion so formuliert werden, dass eine minimale Auslenkung zu einer Veränderung des Funktionswertes führt. Dies kann durch eine Abschätzung von t_{hit} über eine Interpolation von $y(t)$ erreicht werden.

Mit den Stützstellen der Simulation y_k und y_{k-1} zu den Zeitpunkten t_k und t_{k-1} kann man eine lineare Approximation des Zeitpunktes durchführen:

$$\tilde{t}_{hit,k} = t_k - \frac{y_k - \bar{y}}{y_k - y_{k-1}}(t_k - t_{k-1}) \tag{5.34}$$

und man gelangt zu der folgenden Formulierung der Triggerfunktion:

$$\tilde{f}_{hit} = \min\left\{\tilde{t}_{hit,k} \mid t_{k-1} \leq \tilde{t}_{hit,k} \leq t_k\right\} \tag{5.35}$$

Über die Gradienten der Stützstellen ∇y_k und ∇y_{k-1} erfolgt eine Abschätzung des Gradient $\nabla \tilde{t}_{hit,k}$:

$$\nabla \tilde{t}_{hit,k} = \frac{(y_k - \bar{y})(\nabla y_k - \nabla y_{k-1}) - \nabla y_k (y_k - y_{k-1})}{(y_k - y_{k-1})^2}\,(t_k - t_{k-1}) \tag{5.36}$$

[10] Integrationsalgorithmen, die über eine sog. *Zero-Crossing-Detection* verfügen, sind grundsätzlich in der Lage, die korrekten Ableitungen für die Zeitpunkte und Zeitabschnitte der Kriterien zu ermitteln. Die Umsetzung erfordert jedoch eine Anpassung des Integrationsalgorithmus und wird daher nicht weiter betrachtet.

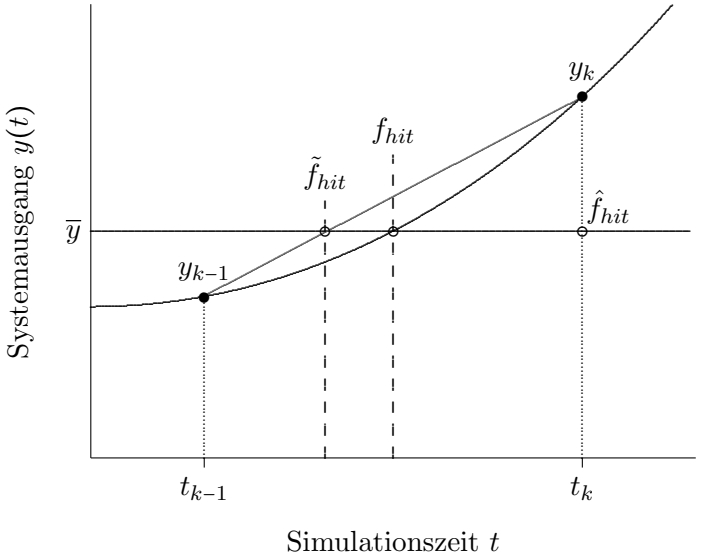


Abbildung 5.9: Arbeitsweise verschiedener Umsetzungen der Triggerfunktion

Weitere Punkt- und abschnittsweise bewertende Kriterien lassen sich in ähnlicher Weise formulieren. Beim Schwellwertkriterium (4.7) kann eine Verschiebung der Integrationsgrenzen ebenfalls über eine lineare Interpolation abgeschätzt werden. Da dieses Kriterium bereits in kontinuierlicher Weise von dem Systemausgang $y(t)$ abhängt, besitzt es im Normalfall bereits einen sinnvollen Gradienten. Die Interpolation der Zeitpunkte ist hier nur erforderlich, wenn die Verschiebungen der Integrationsgrenzen den wesentlichen Einfluss auf den Funktionswert ausüben.

Die Differentiation des Maximalwertkriteriums (4.5) ist nach [Dig04] als unproblematisch anzusehen. Prinzipiell kann eine Abschätzung des Maximalwertes bspw. über eine quadratische oder kubische Interpolation vorgenommen werden. Im Allgemeinen ist dies jedoch unnötig, da es ebenfalls in kontinuierlicher Weise von einem Systemausgang $y(t)$ abhängt.

Grundsätzlich lässt sich festhalten, dass die oben beschriebene Abschätzung von Zeitpunkten immer dann unerlässlich ist, wenn das Ergebnis des jeweiligen Kriteriums als ein Zeitpunkt oder ein Zeitraum interpretiert werden kann.

6 Selbstoptimierende geregelte Systeme

Selbstoptimierende Systeme zeichnen sich dadurch aus, dass sie im Betrieb selbstständig auf sich ändernde Einflüsse und geänderte Zielsetzungen reagieren und ihr Verhalten optimal an die neue Situation anpassen. Aufgrund der großen Bandbreite unterschiedlicher Fragestellungen und der Vielzahl der beteiligten technischen Domänen ergeben sich im Maschinenbau zahlreiche Möglichkeiten, intelligente Systeme zu entwerfen, die der Definition für selbstoptimierende Systeme nach [FGK+04] genügen (siehe Kapitel 6.1.3). Einige Beispiele hierfür sind:

- Selbstoptimierende Systeme, die dem Bereich der Logistik oder Wegeplanung zugeordnet werden können. Mit solchen Systemen kann z. B. eine Zeit-, Verbrauchs- oder Kosteneinsparung bei den individuellen Teilnehmern oder beim Gesamtsystem erreicht werden.

- Selbstoptimierendes Energiemanagement eines komplexen Gesamtsystems. Ein selbstoptimierendes Energiemanagement erlaubt es, den Energieverbrauch und den Leistungsbedarf verschiedener Teilsysteme, insbesondere bei Beschränkungen der Energieversorgung und übertragbaren Leistung, optimal aufeinander abzustimmen, so dass die Funktionalitäten der Teilsysteme erhalten bleiben. Die Ziele und Prioritäten der Teilsysteme werden gegeneinander abgewogen und das Gesamtsystem in einem für die jeweilige Situation „günstigen“ Kompromiss betrieben [KRB+06].

- Die Informationsverarbeitung eines maschinenbaulichen Erzeugnisses, die im Betrieb umkonfiguriert und optimiert wird. Diese Art der Selbstoptimierung zielt auf eine optimale Nutzung der vorhandenen und begrenzten Ressourcen wie Rechenleistung, Speicherplatz und Stromverbrauch oder auch dem Flächenbedarf bei Verwendung von FPGAs[1] ab. Hierdurch lassen sich bspw. die Kosten der Rechenhardware, aber auch deren Energieverbrauch reduzieren [OBR05, OZL10].

Die Umsetzung dieser verschiedenartigen Anwendungen erfordert in der Regel auch recht unterschiedliche Methoden, Verfahren sowie Software- und Hardwarearchitekturen. Der Fokus dieses Kapitels liegt auf der Betrachtung selbstoptimierender Regelungen. Es wird hier auf Systeme eingegangen, bei denen das dynamische Verhalten durch Variation der Regler bzw. Reglereinstellungen an die aktuellen Einflüsse und Ziele angepasst wird.

6.1 Grundlagen und Definitionen

Im diesem Abschnitt werden einige Grundlagen und Definitionen für selbstoptimierende Systeme des Maschinenbaus erläutert. Für weiterführende Informationen sei hier auf [FGK+04] und [ADG+08] verwiesen.

[1] Field Programmable Gate Array

6.1.1 Anforderungen und Ziele

Allgemein definieren die *Anforderungen* und *Ziele* bzw. *Zielsetzungen* ein Wunschverhalten, welches das System annehmen soll, oder sie fordern konkrete Eigenschaften, die zu erreichen oder zu vermeiden sind. Oft werden diese Begriffe synonym zueinander verwendet. Im Kontext selbstoptimierender Systeme kommt ihnen jedoch eine unterschiedliche Bedeutung zu. In [ADG+08] werden *Anforderungen* und *Ziele* die folgenden Bedeutungen zugewiesen:

- *Anforderungen* beziehen sich auf Systemeigenschaften, die in ihrer Ausprägung bereits zum Entwurfszeitpunkt festgelegt werden. Ausprägung bedeutet hier eine eindeutige, quantifizierbare Festlegung der jeweiligen Systemeigenschaft, bspw. durch Angabe fester Kennzahlen oder Schwellwerte. Die Anforderungen werden durch den Entwurf oder die Konstruktion fixiert und können nicht mehr variiert werden.

- Demgegenüber werden *Ziele* oder auch *Zielsetzungen* während des Entwurfs eines selbstoptimierenden Systems zwar vorgesehen, eine Ausprägung findet jedoch noch nicht statt. Es ist die Aufgabe der Selbstoptimierung eine konkrete Festlegung und Bewertung der *Ziele* zur Laufzeit vorzunehmen, um so das Systemverhalten an die aktuelle Situation anzupassen.

Die automatische Auswertung und Bewertung zur Laufzeit durch die Informationsverarbeitung des Systems macht eine Quantifizierung des Grades der *Zielerfüllung* erforderlich. Hierzu werden die oft nur allgemein und unscharf formulierten Ziele in greifbare *Zielgrößen* überführt. Die Zielgrößen stellen ein Maß für die Erfüllung des jeweiligen Ziels dar. Die Berechnung der Zielgrößen erfolgt durch *Bewertungsfunktionen*. Diese bilden das Systemverhalten auf die Zielgrößen ab und arbeiten auf der Basis von Simulationsdaten, Messwerten und beobachteten Daten aber auch auf von externen Systemen übermittelten Daten.

6.1.2 Interne, externe und inhärente Ziele

Die Ziele, die von einem System verfolgt werden, lassen sich nach [FGK+04] in interne, externe und inhärente Ziele unterteilen. Die Unterteilung in inhärente und externe Ziele orientiert sich daran, ob die Ausprägung des jeweiligen Ziels im System selbst stattfindet oder ob sie außerhalb von einem überlagerten oder benachbarten System gefordert wird.

- *Inhärente Ziele* beziehen sich auf Aspekte des Systems, die für umliegende Systeme und das Umfeld nicht direkt von Bedeutung sind. Sie beziehen sich meist auf den Entwurfszweck des Systems und sichern die Grundfunktionalität. Beschränkungen und Randbedingungen des Systems oder Ziele, die der Sicherheit und Zuverlässigkeit dienen, sind Beispiele für inhärente Ziele. Das System ist bspw. selbst dafür verantwortlich, dass Stellgrößenbeschränkungen oder maximale Bauteilbelastungen eingehalten werden. Die Ausprägung dieser Ziele wird vom System zur Laufzeit selbst vorgenommen.

- *Externe Ziele* sind für das Umfeld und die umliegenden Systeme von Interesse. Speziell bei miteinander interagierenden Systemen, hat die Art und Weise, wie ein (Teil-)System seine Aufgaben erfüllt, eine große Auswirkung auf die umliegenden Systeme. Durch die *externen Ziele* kann das Verhalten und somit die Auswirkungen

auf andere Systeme gezielt beeinflusst werden. Sie werden daher von überlagerten oder benachbarten Systemen oder auch direkt vom Benutzer an das System herangetragen. Die Ausprägung findet außerhalb der Systemgrenzen statt. Externe Ziele dienen als Schnittstelle zu anderen Systemen. Durch den Austausch von Zielen wird eine sehr abstrakte und generische Schnittstelle zwischen den verschiedenen interagierenden Systemen erreicht. Beispiele für externe Ziele sind der Energieverbrauch und die Güte der Funktionserfüllung, die von außen einem System vorgegeben werden.

Die *internen Ziele* beschreiben die Zielausprägungen, die zu einem bestimmten Zeitpunkt von dem selbstoptimierenden System verfolgt werden. Sie werden durch Auswahl, Priorisierung oder Gewichtung der inhärenten und externen Ziele festgelegt und zu einem *internen Zielsystem* zusammengefasst. Dieses *interne Zielsystem* unterliegt der direkten Kontrolle durch das System. So stellt das Bilden des internen Zielsystems einen wesentlichen Schritt des Selbstoptimierungsprozesses dar.

6.1.3 Prozess der Selbstoptimierung

Die Anpassung des Systems an neue Zielvorgaben oder geänderte Betriebsbedingungen wird durch den Prozess der Selbstoptimierung vorgenommen. Der Prozess der Selbstoptimierung besteht aus drei wiederkehrenden Schritten [FGK+04, ADG+08]:

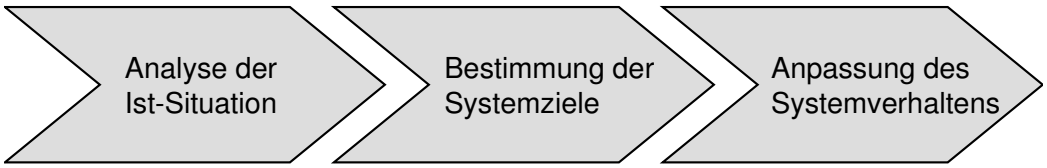


Abbildung 6.1: Die drei Schritte der Selbstoptimierung

Die drei Schritte der Selbstoptimierung werden im Folgenden erläutert.

1. Analyse der Ist-Situation: Als Entscheidungsgrundlage für den Selbstoptimierungsprozess werden im ersten Schritt, der *Analyse der Ist-Situation*, Informationen über den aktuellen Zustand und die Betriebsbedingungen des Systems erfasst und analysiert. Die Informationen beziehen sich dabei auf das System selbst, auf benachbarte Systeme sowie das Umfeld des Systems. Eine wichtige Aufgabe ist die Bestimmung und Prüfung des Erfüllungsgrades der verfolgten Ziele. Der Erfüllungsgrad dient der Beurteilung der aktuellen Situation und des verwendeten internen Zielsystems.

Bei mechatronischen Systemen kommen zur *Analyse der Ist-Situation* besonders regelungstechnische Verfahren zum Einsatz. Vor allem sind in diesem Schritt Filtertechniken, Beobachter und Schätz- und Identifikationsverfahren von Interesse. Diese Techniken erlauben eine Schätzung des Systemzustandes, des Betriebspunktes sowie eine Identifikation von System- und Störmodellen. Darüber hinaus lassen sich bestimmte Informationen auch indirekt durch Kommunikation mit anderen Systemen ermitteln. Auf diese Weise werden die Beobachtungen anderer Systeme mit in den Selbstoptimierungsprozess einbezogen. Lernverfahren eröffnen weitere interessante Möglichkeiten im Falle sich wiederholender Situationen, z. B. bei wiederkehrenden Störungen oder Anregungsformen. Informationen über

solche Situationen können mit jeder Wiederholung erlernt, verfeinert und für die erneute Verwendung in Datenspeichern hinterlegt werden.

2. Bestimmung der Systemziele: Im zweiten Schritt, der *Bestimmung der Systemziele*, wird auf Basis der im ersten Schritt gewonnenen Informationen die Anpassung des internen Zielsystems vorgenommen. Gesucht wird ein für die aktuelle Situation günstiger Kompromiss, der die verschiedenen, teilweise konkurrierenden Ziele berücksichtigt. Dieser Kompromiss wird auf das interne Zielsystem abgebildet. Dies geschieht zum einen durch Priorisierung oder Gewichtung interner Ziele. Hierdurch wird der Einfluss bestimmter Ziele graduell erhöht und der Einfluss anderer Ziele verringert. Zum anderen kann das Zielsystem durch Auswahl anderer Ziele verändert werden. Bisher nicht berücksichtigte externe und inhärente Ziele können hinzugefügt und bestehende interne Ziele aus dem Zielsystem entfernt werden.

3. Anpassung des Systemverhaltens: Die Änderung am internen Zielsystem erfordert eine *Anpassung des Systemverhaltens* im dritten Schritt der Selbstoptimierung. Diese Verhaltensanpassung erfolgt bei mechatronischen Systemen durch Parameter- und Strukturanpassungen innerhalb der steuernden und regelnden Informationsverarbeitung. Parameteranpassung bezeichnet die Veränderung einiger Systemparameter, z. B. Reglerparameter. Bei einer Strukturanpassung werden Elemente der Regelung und Steuerungen ausgetauscht, entfernt oder hinzugefügt.

Einer graduellen Veränderung des internen Zielsystem kann meist durch eine Anpassung der Systemparameter begegnet werden. Die Auswahl neuer Ziele kann aber eine Rekonfiguration der Reglerstruktur zwingend erforderlich machen. So ist es bei Änderung der Regelungsaufgabe oft notwendig, die Regelung auf eine neue Regelgröße umzuschalten und den Regler entsprechend auszutauschen, wie z. B. bei der Umschaltung von einer Positions- auf eine Geschwindigkeitsregelung.

6.1.4 Einflussmöglichkeiten

Die Anpassung eines technischen Systems durch den Selbstoptimierungsprozess erfordert Einflussmöglichkeiten – „Stellschrauben“ – mit denen das Verhalten während des Betriebs beeinflusst werden kann. Einflussmöglichkeiten sind bei mechatronischen Systemen vor allem in der steuernden und regelnden Informationsverarbeitung zu finden. Die hohe Flexibilität von Software ermöglicht es, auf eine einfache Art und Weise eine Variation der Regelungsalgorithmen vorzunehmen.

Das Verhalten eines regelungstechnischen Systems wird über zwei fundamentale Bestandteile kontrolliert: einem Sollwertgenerator und einem Regler (siehe Abbildung 6.2). Der Sollwertgenerator liefert Zeitsignale, die dem Regelkreis aufgeschaltet werden. Diese Signale beschreiben Sollwerte für den Regler, bspw. eine Referenztrajektorie für die zukünftige gewünschte Bewegung des Systems, aber auch Signale zur Kompensation bekannter Störungen können im Sollwertgenerator berechnet und auf die Aktoren des Systems aufgeschaltet werden. Dieses Prinzip wird als *Steuerung*[2] bezeichnet. Die *Regelung*[3] unterscheidet sich von der *Steuerung* durch die Rückführung von Streckengrößen. Die Aufgabe

[2] engl.: feed-forward-control
[3] engl.: feed-back-control

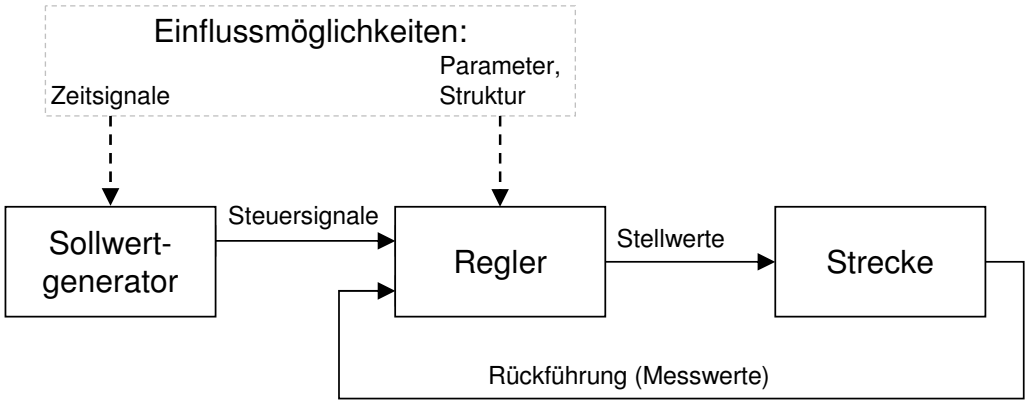


Abbildung 6.2: Einflussmöglichkeiten für eine selbstoptimierende Regelung

des Reglers ist es, auf unbekannte bzw. nicht berücksichtigte Einflüsse zu reagieren. Mit Hilfe der Rückführung werden die Auswirkungen der unbekannten Einflüsse erfasst und in Reaktion darauf neue, korrigierende Stellwerte berechnet.

Eine Anpassung des Systemverhaltens kann sowohl über eine Variation der Steuersignale als auch des Reglers erfolgen. Es stehen die folgenden drei Möglichkeiten der Einflussnahme zur Verfügung:

- **Anpassung der Parameter:** Unter Parametern werden im Zusammenhang regelungstechnischer Systeme meist konstante Größen, wie Verstärkungsfaktoren oder Zeitkonstanten verstanden. Diese Größen werden bei klassischen Systemen während der Auslegung eingestellt und erfahren anschließend im Betrieb keinerlei Wertänderung. Im Kontext selbstoptimierender aber auch adaptiver Systeme werden jedoch Parameteränderungen zur Laufzeit genutzt, um das dynamische Verhalten der jeweiligen regelungstechnischen Komponente an aktuelle Systemzustände oder Situationen anzupassen. Das Schwingungsverhalten des Systems kann dabei sprunghaft aber auch kontinuierlich, durch ein graduelles Verstellen der Parameter, angepasst werden.

- **Anpassung der Struktur:** Strukturänderungen sind unabdingbar, wenn sich das Regelungsziel oder die Regelungsaufgabe ändern. Eine Strukturänderung liegt vor, wenn einer Regelung neue Elemente hinzugefügt oder bestehende entfernt werden, aber auch wenn der Signalfluss zwischen den Elementen neu arrangiert wird. Mit ihr geht meist auch eine Änderung der Parameter einher. Strukturänderungen bewirken im Allgemeinen eine sprunghafte Änderung des Systemverhaltens. Aus dem Umschaltvorgang selbst resultieren im Allgemeinen Unstetigkeiten in den Stellwerten, die in den meisten Fällen unerwünscht sind. Diesen muss mit speziellen Maßnahmen zur Reglerumschaltung begegnet werden. Strukturänderungen erfolgen bspw. bei der Umschaltung zwischen verschiedenen Regelungsaufgaben wie der Wechsel zwischen einer Positions-, einer Geschwindigkeits- oder einer Kraftregelung. Bei stark nichtlinearen Systemen kann aber auch der Wechsel zwischen verschiedenen Betriebspunkten eine Strukturumschaltung erforderlich machen.

- **Anpassung der Steuersignale:** Eine weitere Einflussmöglichkeit stellen zeitliche Steuersignale, wie Sollwertverläufe und Störgrößenkompensationssignale, dar. Diese werden im Signalgenerator berechnet und auf das System aufgeschaltet. Bei der

Aufschaltung von Zeitsignalen können dynamische Effekte des Systems aber auch bekannte innere und äußere Störungen berücksichtigt werden. Hierdurch wird die Regelgüte im Allgemeinen stark verbessert. Der Sollwertgenerator weist verschiedene Freiheitsgrade auf, die als Einflussmöglichkeit für eine selbstoptimierende Regelung genutzt werden können. Eine Bewegungsaufgabe kann auf unterschiedliche Art und Weise umgesetzt werden, z. B. kann eine gewünschte Position schnell, aber mit hohem Energieaufwand oder langsam, mit niedrigem Energieaufwand angefahren werden. Ebenso können Störungen für eine optimale Regelgüte vollständig oder zur Entlastung der Aktorik auch nur teilweise kompensiert werden. Eine Aufschaltung nur in den niedrigen oder mittleren Frequenzbereichen bewirkt in vielen Fällen eine Senkung des Leistungs- und Energiebedarfs. Die Variation der Signale kann durch Filterung erreicht werden. Hierdurch lassen sie sich in ihrer Amplituden- und Frequenzcharakteristik verändern.

Sollwertverläufe können meist direkt aus den Bewegungsaufgaben des Systems generiert werden. Für eine Störgrößenkompensation muss jedoch der Verlauf der zukünftigen Störeinflüsse bekannt sein oder zumindest abgeschätzt werden können. Mit ihrer Hilfe lassen sich Signale bestimmen, welche auf die Aktorik aufgeschaltet werden und so die Auswirkungen der störenden Einflüsse kompensieren. Die direkte Messung von äußeren Einflüssen, speziell aus der Umgebung des Systems, ist oft schwierig und erfordert in vielen Fällen zusätzliche, vorausschauende Sensoren. Häufig werden daher Beobachter, wie Luenberger Beobachter oder Kalmanfilter, zur Abschätzung der Störeinflüsse eingesetzt. Die Vernetzung von Systemen zu VMS eröffnet neue Möglichkeiten, diese Einflüsse zu erfassen. Bestimmte Störungen wirken sich mit einer zeitlichen Verzögerung auf mehrere Systeme eines VMS aus. Durch Kommunikationsmechanismen können Daten hierüber zwischen den Systemen ausgetauscht und für eine Berechnung der Zeitsignale genutzt werden. Beispiele hierfür finden sich in [MHO+04b, MVH05, TMV06, VT08].

6.2 Adaptive und selbstoptimierende Regelungen

Neben selbstoptimierenden Reglern sind auch adaptive Regelungsverfahren in der Lage, sich selbstständig an sich ändernde Situationen anzupassen. Adaptive Regler wurden entwickelt um Situationen zu bewältigen, bei denen sich Parameter wie Trägheiten, Dämpfungen oder Betriebsbedingungen wie äußere Lasten oder Kräfte dramatisch ändern. Ein klassisches Beispiel für Systeme mit sich ändernden Parametern ist eine gelenkte Rakete. Sie verliert durch das Verbrennen von Treibstoff Masse, ebenso verändert sich der Luftwiderstand mit zunehmender Flughöhe. Bei diesen sich stark ändernden Einflüssen arbeiten klassische statische Regler nicht mehr zufriedenstellend während des gesamten Betriebs.

Adaptive Regler erfassen diese veränderten Betriebsbedingungen und passen sich daran an, mit dem Ziel, das Regelungsverhalten zu erhalten. Sie sorgen also für ein konstantes Verhalten, welches bezüglich eines festen, zum Entwurfszeitpunkt definierten Ziels optimal ist. Selbstoptimierende Regler gehen in diesem Punkt über adaptive Regler hinaus, da hier auch Änderungen an den einzelnen Kriterien und am Zielsystem durchgeführt werden.

Selbstoptimierende Regler besitzen eine enge Verwandtschaft mit adaptiven Reglern, so dienen einige Ansätze der adaptiven Regelungen als Basis für die Entwicklung selbst-

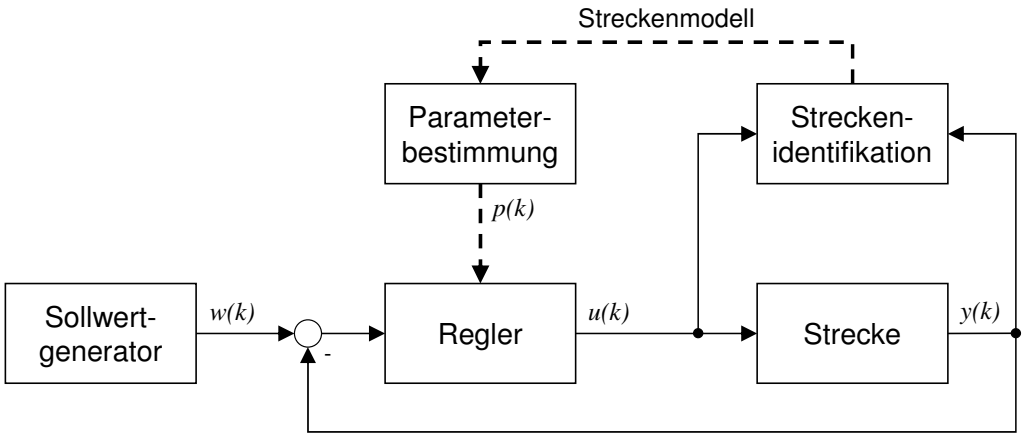


Abbildung 6.3: Adaptiver Regler: *Model identification adaptive control* (*MIAC*)

optimierender Regler. Insbesondere die Klasse der *Model identification adaptive control* (*MIAC*) kommt den Vorstellungen an einen selbstoptimierenden Regler bereits sehr nahe. Wegen der Fähigkeit Änderungen des Systemverhaltens im Betrieb zu erfassen und das Reglerverhalten hierauf anzupassen, wird ein derartiger Regler oft auch als *self-tuning controller* bezeichnet (siehe z. B. [SB89, ILM92]).

Abbildung 6.3 zeigt den Aufbau dieses Reglers. Den Ausgangspunkt bildet ein konventioneller, innerer Regelkreis, bestehend aus Regler und Strecke, sowie ein Sollwertgenerator. Die Adaption wird über einen zweiten überlagerten „Regelkreis“ erreicht, dessen Aufgabe in der Anpassung des Verhaltens besteht. Mit Hilfe einer Streckenidentifikation wird zur Laufzeit das Verhalten der Regelstrecke erfasst. Durch die ständige Aktualisierung des Modells werden Änderungen des Systemverhaltens und der äußeren Einflüsse erkannt und in ein mathematisches Modell überführt. Die Parameterbestimmung führt auf Basis dieses Modells die Berechnung der neuen, auf die Situation angepassten Reglerparameter durch. Diese werden anschließend im Regler eingestellt.

Sowohl bei der Parameterbestimmung als auch bei dem Identifikationsprozess kann auf eine Vielzahl unterschiedlicher Methoden und Verfahren zurückgegriffen werden. Zur Identifikation des Systemverhaltens können Verfahren wie bspw. die Methode der kleinsten Fehlerquadrate (method of least squares), die Maximum-Likelihood-Methode oder auch erweiterte Kalmanfilter (EKF) zum Einsatz kommen (siehe z. B. [Ise92b, Ise92a]).

Einige Ansätze zur Parameterbestimmung verwenden feste Einstellregeln oder analytische Methoden, um aus dem identifizierten Modell direkt die Reglerparameter zu ermitteln. Die Berechnung kann aber auch auf Basis von Optimierungsverfahren erfolgen. Eine Online-Optimierung sichert zum einen die Optimalität der Ergebnisse, zum anderen besteht hier der größte Gestaltungsspielraum bei der Definition der Regelungsziele. So lassen sich auch Ziele berücksichtigen, die mit festen Einstellregeln nicht formuliert werden können.

Da sich die für die Selbstoptimierung notwendige Anpassung des Zielsystems leicht auf die Optimierungskriterien der Online-Optimierung übertragen lässt, können solche adaptiven Regler relativ einfach als Basis für selbstoptimierende Regler herangezogen werden. Abbildung 6.4 zeigt den um einen Selbstoptimierungsprozess erweiterten adaptiven Regler. Die Mehrzieloptimierung realisiert hier den dritten Selbstoptimierungsschritt (vgl. [FGK+04, Obe08]).

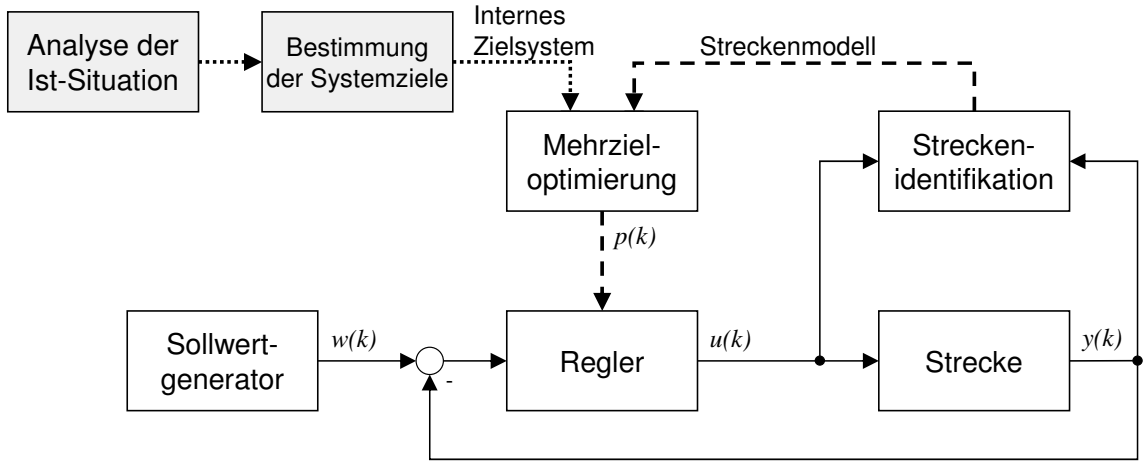


Abbildung 6.4: Selbstoptimierender adaptiver Regler

Eine große Einschränkung für die Anwendung der Mehrzieloptimierung ergibt sich aus dem benötigten Rechenaufwand. Die Berechnung einer kompletten Paretomenge kann zur Laufzeit mit aktuell verfügbarer Rechenleistung nur für kleine oder spezielle Probleme innerhalb der für die Adaption notwendigen Zeitintervalle durchgeführt werden. Die Auswahl der Optimierungsverfahren beschränkt sich daher auf Lösungsansätze, die, anstelle die komplette Paretomenge zu berechnen, lediglich einen einzelnen Paretopunkt gezielt anfahren (vgl. Kapitel 3.4).

In [HMS04] wurde bspw. ein selbstoptimierender adaptiver Regler realisiert, der zur Lösung des Mehrzielproblems eine spezielle onlinefähige Version des Mehrzieloptimierungsverfahrens MOPO einsetzt. Das Mehrzieloptimierungsproblem ist als ein Goal-Attainment-Problem formuliert, dessen Lösung unter sich ständig ändernden Randbedingungen mittels eines Gradientenverfahrens sukzessive angenähert wird. Dieses Verfahren wird in Kapitel 9.4.2 näher beschrieben.

Boecker und andere stellen in [BSKF06] den Bezug zwischen selbstoptimierenden und adaptiven Reglern her und definieren selbstoptimierende Regler als eine Erweiterung klassischer und adaptiver Regler. Abbildung 6.5 zeigt diese Erweiterung klassischer Regelungsansätze zu einer selbstoptimierenden Regelung. Eine selbstoptimierende Regelung besteht demnach aus drei aufeinander aufbauenden Stufen:

Idealisiertes System: Im Zentrum befindet sich die Regelstrecke, also das technische System, welches äußeren und inneren Einflüssen unterliegt. Die Reaktionen des Systems werden mittels eines herkömmlichen Reglers kompensiert. Sind die Einflüsse bekannt, können sie auch mit Hilfe eines geeigneten Modells durch eine Steuerung im Sollwertgenerator berücksichtigt werden. Sollwertgenerator und Regler sind Teil einer klassischen Regelung, die das Ziel hat, dem System ein gewünschtes Verhalten aufzuprägen. Viele Effekte des ursprünglichen Systemverhaltens werden durch Einsatz des Reglers kompensiert. Blickt man von außen auf das geregelte System, so sind diese Effekte vernachlässigbar. Die regelungstechnischen Maßnahmen führen zu einem *idealisierten System*.

Optimales System: Die klassische Regelung kann durch Adaptionsalgorithmen zu einer adaptiven Regelung erweitert werden. Treten Veränderungen bei den Einflüssen oder im

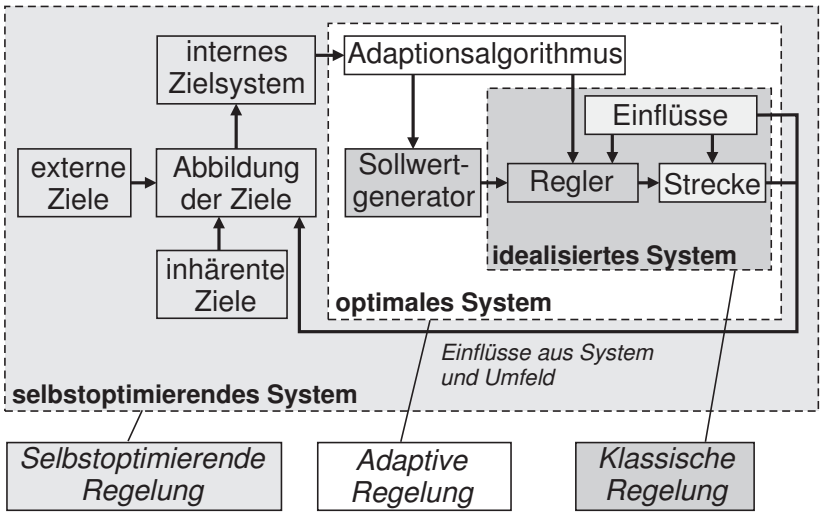


Abbildung 6.5: Erweiterung klassischer und adaptiver Regelungen zur selbstoptimierenden Regelung [BSKF06]

Verhalten der Strecke auf, sorgt sie für eine Anpassung des Reglers oder des Sollwertgenerators. Ziel ist der Erhalt eines vergleichbaren Regelungsverhaltens. Dieses Verhalten ist bezüglich eines fest eingestellten Ziels optimal; die Erweiterung um Mechanismen der Regleradaption führt also zu einem *optimalen System*.

Selbstoptimierendes System: Klassische oder adaptive Regelungen sind geeignet, um auf Veränderungen des Systems oder der auf das System wirkenden Einflüsse zu reagieren. Ändern sich jedoch die Anforderungen an das System, so ist die adaptive Regelung wegen ihrer Beschränkung auf ein festes Optimierungsziel nicht mehr ausreichend. An diesen Punkt setzten *selbstoptimierende Regelungen* an. Sie verwenden an Stelle eines festen Optimierungsziels ein internes Zielsystem, welches die aktuellen Anforderungen in Form mehrerer Einzelziele beschreibt. Im internen Zielsystem lassen sich sowohl Änderungen von externen oder inhärenten Zielen als auch Veränderungen der Einflüsse aus System oder Umfeld berücksichtigen. Hierdurch ergeben sich neue Freiheitsgrade: die Zusammensetzung des Zielsystems sowie die Priorisierung der Einzelziele. Diese werden vom Selbstoptimierungsprozess genutzt, um Anpassungen an dem unterlagerten optimalen System durchzuführen. Selbstoptimierende Regelungen nutzen das variable Zielsystem, um auf sich ändernde Einflüsse aus System, Umfeld sowie geänderte externe Vorgaben zu reagieren.

Die drei dargestellten Stufen ermöglichen einen schrittweisen Entwurf des selbstoptimierenden Systems. Die Komplexität des Systems steigert sich dabei mit jeder Ausbaustufe von dem einfachen Regler über den adaptiven bis hin zum anspruchsvollen selbstoptimierenden Regler.

Die Erweiterung eines klassischen Reglers um Adaption und Selbstoptimierung ist dabei nicht als starr anzusehen. Je nach System und Regelungsansatz sind Variationen möglich oder notwendig. So lassen sich einige anspruchsvollere Regelungsansätze nicht ohne Weiteres in einen klassischen und einen adaptiven Teil zerlegen, wie z. B. modellbasierte prädiktive Regelungen (siehe z. B. [DP04]). Ebenso ist das Vorhandensein einer Adaptionsstrategie – im Sinne eines adaptiven Reglers – nicht Voraussetzung für einen selbstoptimierenden Regler.

6.3 Selbstoptimierende Regelungen auf Basis paretooptimaler Konfigurationen

Das Hauptmerkmal eines selbstoptimierenden Reglers besteht in dem selbstständigen Anpassen des internen Zielsystems. Es ist naheliegend, für die Berechnung der, bezüglich dieses Zielsystems, optimalen Systemkonfigurationen mathematische Optimierungsverfahren einzusetzen. Da das Zielsystem, wie oben bereits erwähnt, stets aus mehreren Einzelzielen besteht, führt dieser Lösungsansatz unweigerlich auf ein Mehrzieloptimierungsproblem. Das Ergebnis dieser Optimierung ist eine Paretomenge, welche die möglichen, optimalen Systemeinstellungen enthält (vgl. Kapitel 3.3). In diesem Abschnitt wird das Konzept eines Selbstoptimierungsprozess vorgestellt, der auf Basis dieser Systemeinstellungen arbeitet, konkrete Ausprägungen folgen dann im Kapitel 9.

6.3.1 Selbstoptimierungsprozess

Ein Ansatz, eine selbstoptimierende Regelung auf Basis von paretooptimalen Systemkonfigurationen zu realisieren, besteht darin, zur Laufzeit einen für die aktuelle Situation günstigen Punkt aus der Paretomenge zu wählen und entsprechend im Regler einzustellen. Durch die Auswahl des Paretopunktes wird eine Anpassung des Systemverhaltens an geänderte Ziele und Betriebsbedingungen erreicht.

In [FGK+04] wird der Einsatz von Optimierungsverfahren weitgehend dem dritten Schritt der Selbstoptimierung, der *Anpassung des Systemverhaltens*, zugeordnet. Die Optimierung dient hier der Berechnung der benötigten optimalen Systemparameter. Dies trifft bspw. auf den um Selbstoptimierung erweiterten, adaptiven Regler zu, welcher im vorangegangenen Abschnitt 6.2 beschrieben wurde.

Im Gegensatz hierzu wird in dieser Arbeit die modellbasierte Mehrzieloptimierung vom Selbstoptimierungsprozess entkoppelt und als eigenständiger Prozess angesehen. Der Optimierungsprozess greift nicht direkt in das Verhalten des Systems ein, sondern liefert dem Selbstoptimierungsprozess vielmehr eine Entscheidungsgrundlage, die in Form von Paretomengen zur Verfügung gestellt wird. Abbildung 6.6 zeigt dieses Zusammenwirken zwischen der Mehrzieloptimierung und dem Selbstoptimierungsprozess.

Die Paretomengen werden dem Selbstoptimierungsprozess übermittelt. Sie enthalten zwei Arten von Informationen, die hier genutzt werden können:

- Der **Bildraum** einer Paretomenge liefert Aussagen über die Quantität der einzelnen Ziele sowie das Verhältnis zwischen den Zielen. Dies kann im Schritt *Bestimmung der Systemziele* zur Ausprägung des internen Zielsystems genutzt werden. Auf Basis der aktuellen Situation wird aus der Bildmenge der Punkt ermittelt, der den aktuellen inhärenten und externen Zielen am besten gerecht wird. Der gewählte Paretopunkt lässt sich anschließend direkt in das interne Zielsystem überführen.

- Der **Urbildraum** enthält die zugehörigen optimalen Systemkonfigurationen, wie z. B. Zahlenwerte der Reglerparameter. In der *Anpassung des Systemverhaltens* wird der zum Zielsystem passende Punkt des Urbildraums gewählt und die Regelungstechnik des Systems entsprechend konfiguriert.

Der Schritt *Bestimmung der Systemziele* „arbeitet“ somit im Bildraum des Systems; der Schritt *Anpassung des Systemverhaltens* nimmt die Abbildung auf den Urbildraum vor.

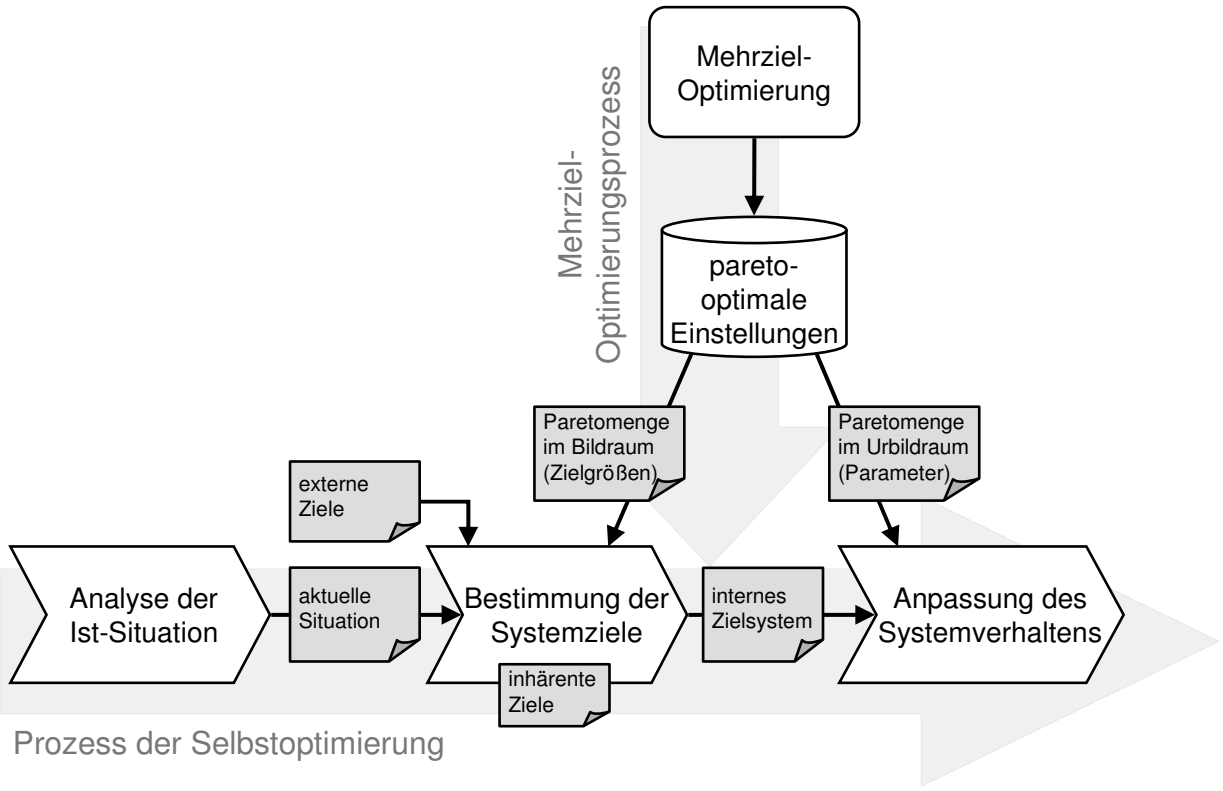


Abbildung 6.6: Prozess der Selbstoptimierung auf Basis paretooptimaler Systemeinstellungen

6.3.2 Wissensbasis

Bei dem Ansatz, eine selbstoptimierende Regelung auf Basis paretooptimaler Konfigurationen aufzubauen, müssen die Paretomengen zur Auswahl einer geeigneten Systemkonfiguration komplett vorliegen. Die Berechnung einer Paretomenge gestaltet sich jedoch wesentlich aufwändiger, als das Lösen eines einzelnen Optimierungsproblems. Im Allgemeinen kann sie mit aktuell verfügbarer Rechenhardware zur Laufzeit nicht innerhalb kurzer Zeitabstände bestimmt werden. Eine Online-Mehrzieloptimierung mit der Absicht, die Ergebnisse direkt in die Regelung einzustellen, ist daher meist nicht praktikabel. Für die Berechnung der Paretomengen ergeben sich daher zwei Möglichkeiten:

- Die Berechnung erfolgt zur Laufzeit für neu auftretende Betriebssituationen. Die Ergebnisse der Optimierung werden gespeichert und stehen so für zukünftige, vergleichbare Situationen zur Verfügung. Die neuen Betriebssituationen müssen erfasst werden, z. B. durch Identifikations- oder Beobachtertechniken.

- Die Paretomengen werden vorab beim Entwurf des Systems ermittelt. In vielen Anwendungsfällen ist eine Berechnung zur Laufzeit nicht erforderlich. Sind während des Betriebes keine oder nur geringe unbekannte Einflüsse auf das Streckenverhalten zu erwarten, ist eine Identifikation des Verhaltens mit der darauf aufsetzenden Berechnung der Paretomenge nicht erforderlich. Dasselbe gilt für Systeme deren Paretofronten sich sehr robust gegenüber Änderungen verhalten. Optimale Systemeinstellungen können in diesen Fällen bereits zum Entwurfszeitpunkt berechnet werden.

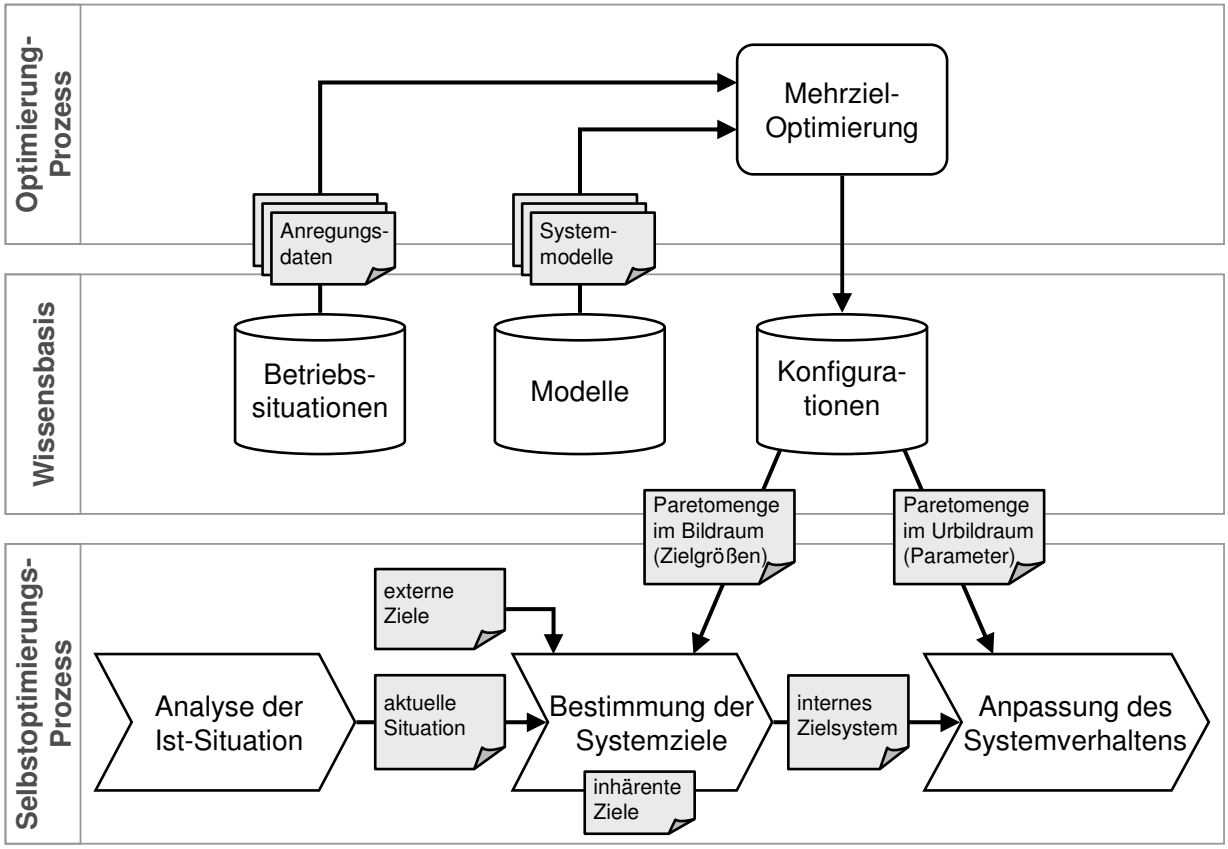


Abbildung 6.7: Wissensbasis für selbstoptimierende Regelungen

In beiden Fällen ist der Selbstoptimierungsvorgang zeitlich von der rechenintensiven Optimierung entkoppelt. Er wird so in die Lage versetzt, ohne Durchführung aufwändiger Berechnungen schnell auf geänderte Situationen zu reagieren. Die berechneten Paretomengen werden gespeichert und stehen so der Selbstoptimierung zu einem beliebigen späteren Zeitpunkt zur Verfügung. Zur Speicherung und zum Austausch der Paretomengen ist die Einführung einer zentralen Datenbank sinnvoll, die im Folgenden als *Wissensbasis* bezeichnet wird.

Das Zusammenwirken von Wissensbasis, Mehrzieloptimierungsprozess und Selbstoptimierungsprozess für eine selbstoptimierende Regelung ist in Abbildung 6.7 dargestellt. Die Wissensbasis ermöglicht hier einen gemeinsamen Zugriff von Optimierungsprozess und den Selbstoptimierungsschritten auf die Paretomengen.

Innerhalb eines selbstoptimierenden Systems existieren viele weitere Funktionen oder Prozesse, die Zugriff auf Daten und Informationen über das System und sein Umfeld benötigen. Die Aufgaben dieser Prozesse bestehen u. a. in:

- der Auslegung regelungstechnischer Komponenten, wie Regler, Steuerungen oder Beobachter.
- der Planung zukünftiger Aktionen.
- der Überwachung des technischen Systems.
- der Identifikation des System- oder Anregungsverhaltens.

- der Kommunikation, zum Austausch von Informationen mit anderen Systemen.

Die *Wissensbasis* bietet eine Möglichkeit, aktuelle Informationen über das System und sein Umfeld zu hinterlegen und den verschiedenen Prozessen zur Verfügung zu stellen.

Neben den optimalen Systemkonfigurationen werden bspw. auch mathematische Modelle des Systemverhaltens oder Informationen über typische Betriebssituationen, wie z. B. Anregungsdaten, benötigt. Die in der *Wissensbasis* hinterlegten Informationen umfassen, sind aber nicht beschränkt auf:

- **Modelle:** Hierunter sind situationsunabhängige Verhaltensbeschreibungen, wie Modelle des Systems und des Systemumfelds, aber ebenso Bewertungsmodelle zu verstehen. Je nach Anwendungsfall können zur Verhaltensrepräsentation parametrische Modelle, also Modelle in Form mathematischer Gleichungen oder Differentialgleichungen, aber auch nicht parametrische Modelle, wie z. B. Kennfelder, zum Einsatz kommen.

- **Situationen:** Situationsabhängige Daten umfassen u. a. Anregungsdaten im Zeit- oder Frequenzbereich, Anregungsmodelle sowie sonstige kontinuierliche oder diskrete Betriebszustände des Systems oder seines Umfeldes.

- **Konfigurationen:** Die optimalen Parametersätze und Reglerstrukturen werden zusammen mit den zugehörigen optimalen Zielgrößenwerten in Form von Paretomengen abgelegt. Für unterschiedliche Situationen können jeweils eigene Paretomengen notwendig sein.

Insgesamt besitzt die *Wissensbasis* die folgenden Eigenschaften und Vorteile für den Entwurf und den Betrieb der Informationsverarbeitung eines selbstoptimierenden Systems:

- **Zentrale Datenverwaltung**: Die Verteilung der Informationen an die verschiedenen Prozesse wird durch die zentrale Datenhaltung stark vereinfacht. Die Daten verschiedener Informationsquellen werden in der *Wissensbasis* zusammengeführt und stehen so sämtlichen Prozessen zur Verfügung. Eine redundante Datenhaltung wird vermieden.

- **Aktualisierbarkeit und Aktualität**: Die Daten innerhalb der *Wissensbasis* werden z. B. durch Identifikationsprozesse oder Optimierungsprozesse verändert. Die Aktualisierung der Daten kann über die *Wissensbasis* leicht koordiniert werden. Die veränderten Daten werden den verschiedenen Prozessen zeitgleich zur Verfügung gestellt. Die Prozesse arbeiten auf Basis der selben, aktuellen Daten. Unterschiedliche Informationsstände in den Prozessen werden vermieden.

- **Definierte Schnittstellen**: Der Zugriff auf das gespeicherte „Wissen“ erfolgt über definierte Schnittstellen. Der Informationsaustausch mit anderen Systemen wird erheblich vereinfacht.

- **Flexibilität**: Durch die Trennung von Daten und Anwendung kann der Informationsaustausch und die Verteilung der Information einfach auf neue Prozesse ausgedehnt werden. Zur Laufzeit erzeugte Prozesse lassen sich flexibel an die *Wissensbasis* anbinden.

6.3.3 Einordnung im OCM

Die Wissensbasis eines selbstoptimierenden Systems ist dem kognitiven Operator eines OCM zugeordnet (Abb. 6.8). Die Prozesse des kognitiven Operators erhalten so einen direkten Zugriff auf die Wissensbasis und deren Inhalte. Der direkte Zugriff des reflektorischen Operators auf die Wissensbasis ist aufgrund der harten Echtzeitanforderungen nicht möglich.

In vielen Fällen werden bestimmte Informationen, z. B. Daten über bekannte Störungen, im reflektorischen Operator benötigt. Um eine zeitliche Entkopplung des kognitiven und reflektorischen Operators zu erreichen und um den Echtzeitanforderungen zu genügen, werden diese Informationen aus der Wissensbasis zu definierten Zeitpunkten in einen Zwischenspeicher im reflektorischen Operator geladen. Eine komplette Spiegelung der Wissensbasis in einen solchen Speicher ist nicht erforderlich, es müssen hier nur die aktuell benötigten Informationen übertragen werden.

Die Mehrzieloptimierung kann für viele Betriebssituationen bereits zum Entwurfszeitpunkt des Systems durchgeführt werden. Müssen während des Betriebes keine weiteren Situationen berücksichtigt werden, so sind die Paretomengen initial bereits in der Wissensbasis hinterlegt und das OCM enthält keinen Optimierungsprozess. Ist eine Online-Optimierung vorgesehen, so ist sie im kognitiven Operator des OCM angeordnet.

Der Kommunikations- oder der Identifikationsblock sind Beispiele für weitere optionale Bestandteile des kognitiven Operators. Der Kommunikationsblock dient hier dem Austausch von „Wissen“ über das System oder das Umfeld mit anderen OCM. Der kognitive Operator kann um weitere, hier nicht näher aufgeführte Komponenten, ergänzt werden.

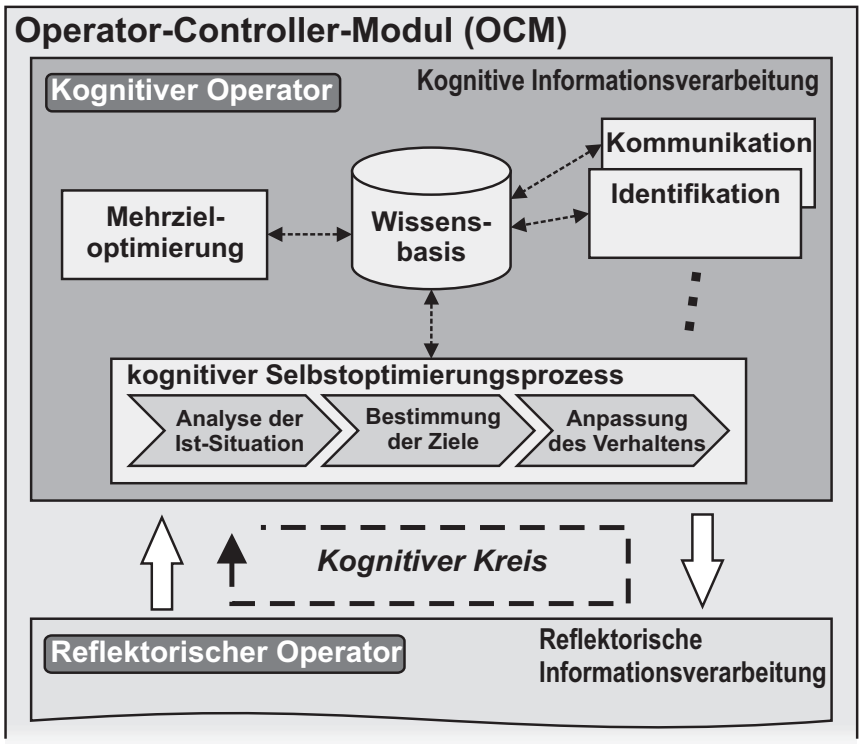


Abbildung 6.8: Einbettung der Wissensbasis und des Selbstoptimierungsprozesses in die Informationsverarbeitung

Viele Bestandteile eines selbstoptimierenden Systems, die den großen Unterschied zwischen einem selbstoptimierenden und einem konventionellen System ausmachen, wie z. B. Optimierungsprozesse oder die Wissensbasis, sind dem kognitiven Operator zugeordnet. Aus funktionalen Gesichtspunkten heraus und aus Gründen einer klaren Strukturierung ist der Selbstoptimierungsprozess ebenfalls dort einzuordnen. Die Unterteilung des Operators in reflektorischen und kognitiven Teil ist nach [OHG04a] der Trennung zwischen harten und weichen Echtzeitanforderungen geschuldet (vgl. Seite 10). Der Selbstoptimierungsprozess muss dementsprechend so entworfen werden, dass keine harten Echtzeitanforderungen an die Informationsverarbeitung gestellt werden.

Neben einem einfachen Selbstoptimierungsprozess, wie er im kognitiven Operator in Abbildung 6.8 dargestellt ist, sind auch kaskadierte Prozessstrukturen denkbar. Hierbei bestimmt ein übergeordneter Selbstoptimierungsprozess die Ziele für einen untergeordneten Selbstoptimierungsprozess.

Da der Selbstoptimierungsprozess das Verhalten des realen Systems beobachtet und letztendlich beeinflusst, sind der Controller und der reflektorische Operator zumindest am ersten und am dritten Schritt der Selbstoptimierung – der *Analyse der Ist-Situation* und der *Anpassung des Systemverhaltens* – in irgendeiner Form beteiligt. Diese Beteiligung läuft bspw. darauf hinaus, dass der Controller lediglich Messwerte zur Verfügung stellt, die vom reflektorischen Operator gepuffert und erst im kognitiven Operator analysiert werden. Gegebenenfalls findet aber bereits im Controller oder im reflektorischen Operator eine Vorverarbeitung statt. So kann die Situationsanalyse auf Ergebnisse von Beobachtern zurückgreifen, die gleichzeitig auch zur Regelung im Controller oder zur Überwachung des Systems im reflektorischen Operator benötigt werden.

7 Selbstoptimierung in hierarchischen mechatronischen Systemen

Die betrachteten selbstoptimierenden mechatronischen Systeme sind aus einzelnen Subsystemen, den VMS, AMS, MFG und MFM, zusammengesetzt. Das Verhalten jedes dieser Subsysteme wird über eine eigene intelligente, informationsverarbeitende Einheit, das Operator-Controller-Modul, beeinflusst. Während im vorangegangenen Kapitel 6 Prozesse und Vorgänge betrachtet wurden, die innerhalb eines einzelnen OCM ablaufen, werden in diesem Kapitel Methoden und Verfahren betrachtet, die OCM-übergreifend in der Makro-

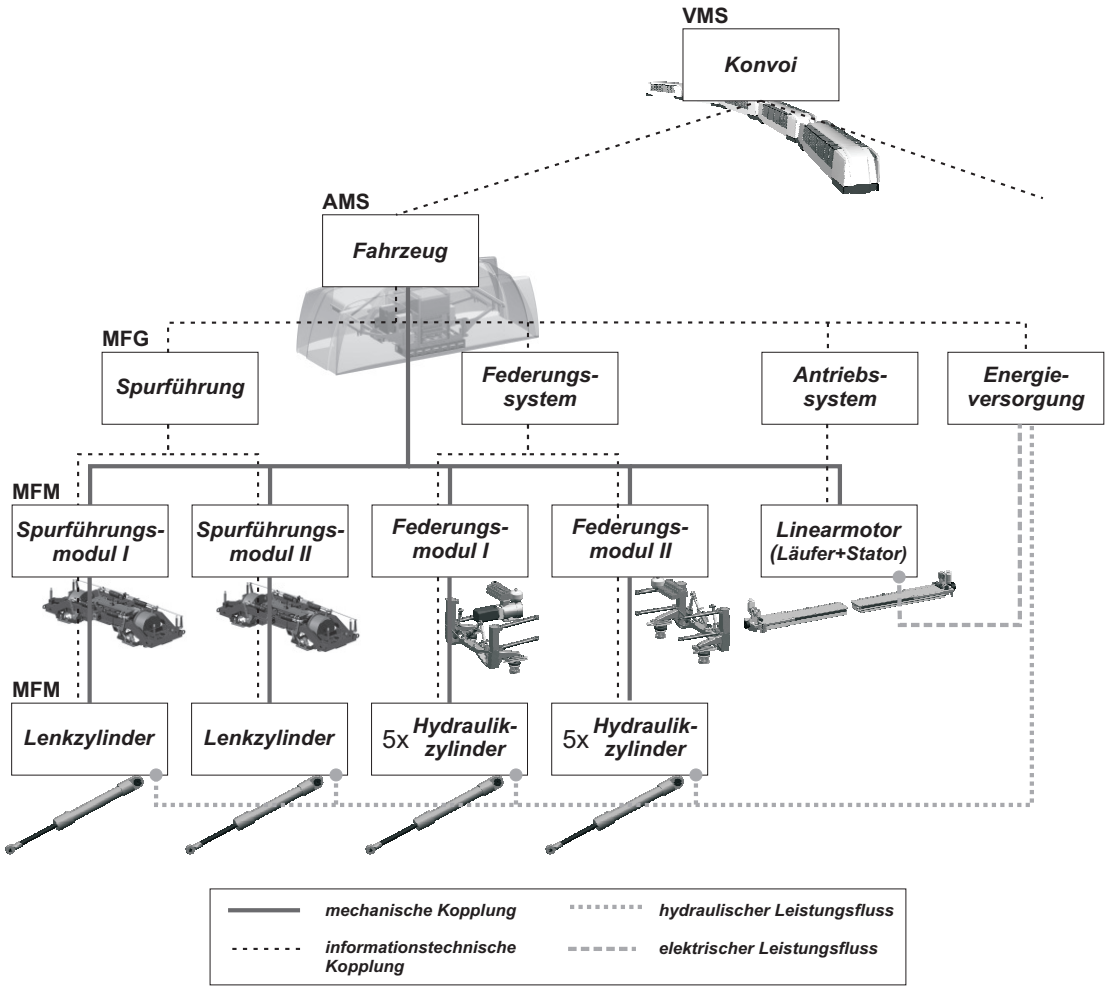


Abbildung 7.1: Hierarchische Aggregatstruktur des RailCabs[1]

[1] Die eingezeichneten mechanischen Kopplungen orientieren sich an den Teilbewegungsfunktionen und sind daher nicht vollständig. Tatsächlich sind bspw. die Linearmotoren über das Fahrwerk mit der Spurführung verbunden.

struktur der Informationsverarbeitung zum Einsatz kommen (vgl. Kapitel 2.2).

Der Fokus liegt auf Systemen bzw. Teilsystemen, bei denen eine weitgehend isolierte Betrachtung der einzelnen Module aufgrund starker gegenseitiger Kopplungen nicht vorgenommen werden kann. Insbesondere die Ebene der MFM ist oft geprägt von hochdynamischen physikalischen und regelungstechnischen Wechselwirkungen. Die Verhaltensänderung eines MFM führt hier direkt zu einer signifikanten Beeinflussung des Schwingungsverhaltens des gesamten Systems. Eine isolierte Betrachtung der Subsysteme kann hier nicht vorgenommen werden, vielmehr muss die Selbstoptimierung koordiniert im Verbund der Subsysteme bis hin zum gesamten betrachteten System erfolgen.[2]

Abbildung 7.1 zeigt die Struktur der Subsysteme am Beispiel des RailCabs. Diese Struktur ist durch einen modular-hierarchischen Aufbau gekennzeichnet, der sich aus der Funktionsstruktur ergibt. Jedes der drei MFG *Spurführung*, *Federungssystem* und *Antriebssystem*, realisiert eine Teilbewegungsfunktion des *Aufbaus*, des AMS. Die Selbstoptimierung erfolgt verteilt innerhalb des Systems, unter Ausnutzung der hierarchischen Struktur.

7.1 Anforderungen

Aus Sicht dieser modular-hierarchischen Architektur ergeben sich die folgenden Anforderungen an den verteilten Selbstoptimierungsprozess:

- **Modularität:** Die Komplexität eines verteilten selbstoptimierenden Systems erfordert eine ausgeprägte Modularität und starke Kapselung der einzelnen Operator-Controller-Module. Durch das Prinzip der Kapselung wird der interne Aufbau der regelungstechnischen Bestandteile und die Implementierung des OCM versteckt. Über definierte, möglichst abstrakte Schnittstellen findet ein Informationsaustausch mit den benachbarten OCM statt. Eine hohe Modularität ermöglicht einen teilweise unabhängigen Entwurf der einzelnen Module, beispielsweise durch verschiedene Entwicklergruppen. Die einzelnen Module lassen sich, im Sinne eines Bottom-Up-Entwurfs, bis zu einem gewissen Grad einzeln testen, ebenso wird die Wartbarkeit und der Austausch einzelner Module erleichtert.

- **Stabilität:** Der Selbstoptimierungsprozess muss stabile Lösungen für das Gesamtsystem liefern. Für eine ausgeprägte Modularität und eine starke Kapselung der einzelnen OCM, ist prinzipiell eine isolierte Betrachtung der einzelnen Subsysteme wünschenswert. Aufgrund der Wechselwirkungen ist ein einzelnes OCM ohne Informationen über die umliegenden Teilsysteme nicht in der Lage abzuschätzen, ob eine bestimmte Veränderung seines Verhaltens auch den gewünschten Effekt auf das Verhalten des Gesamtsystems hat. Dies gilt besonders für die unteren Strukturebenen. Eine Anpassung der regelungstechnischen Komponenten darf nicht ausschließlich aus den lokalen Informationen eines einzelnen Subsystems motiviert sein, sondern muss aus einer Betrachtung des gesamten Teilsystems resultieren.

- **Skalierbarkeit:** Der Selbstoptimierungsprozess innerhalb der hierarchischen Architektur sollte skalierbar bezüglich der Anzahl der beteiligten Subsysteme sein. Ska-

[2] Eine Bewertung der Wechselwirkungen kann über so genannte Koppelmaße erfolgen. Hestermeyer schlägt in [Hes06] das Relative Gain Array (RGA) als Koppelmaß für die Bewertung der Verkopplungen innerhalb der hierarchischen Aggregatstruktur vor (siehe auch [Bri66] und [SP96]).

lierbar bedeutet in diesem Zusammenhang, dass der Aufwand zur Ausführung der Selbstoptimierung nicht überproportional mit der Anzahl der Hierarchieebenen und mit der Anzahl der Subsysteme ansteigt. Hierdurch lässt sich der Selbstoptimierungsprozess sowohl auf kleine Systeme, mit nur zwei Subsystemen, als auch auf große Systeme mit sehr vielen Subsystemen anwenden. Der Aufwand umfasst u. a. den Rechenaufwand, den Kommunikationsaufwand und den Speicherplatzbedarf der einzelnen OCM.

Planungs- und Informationshorizont

Betrachtet man die informationstechnischen Systeme in der Aggregatstruktur, die OCM, so können zwei Trends ausgemacht werden. Zum einen müssen die OCM der höheren Strukturebenen einen wesentlich größeren Umfang an Subsystemen berücksichtigen. Die kumulierte Informationsmenge über die betrachteten Subsysteme, im Folgenden als Informationshorizont bezeichnet, wächst mit der Strukturebene stark an. Zum anderen ist bei Planungssystemen damit in der Regel auch ein Anstieg des Planungshorizontes verbunden. Als Planungssysteme seien hier Techniken verstanden, welche Aktionen des (Teil-)Systems für einen gewissen Zeitraum oder auch Wegstrecke, dem Planungshorizont, vorausschauend bestimmen. Bei geregelten Systemen kann demgegenüber nicht von einem Planungshori-

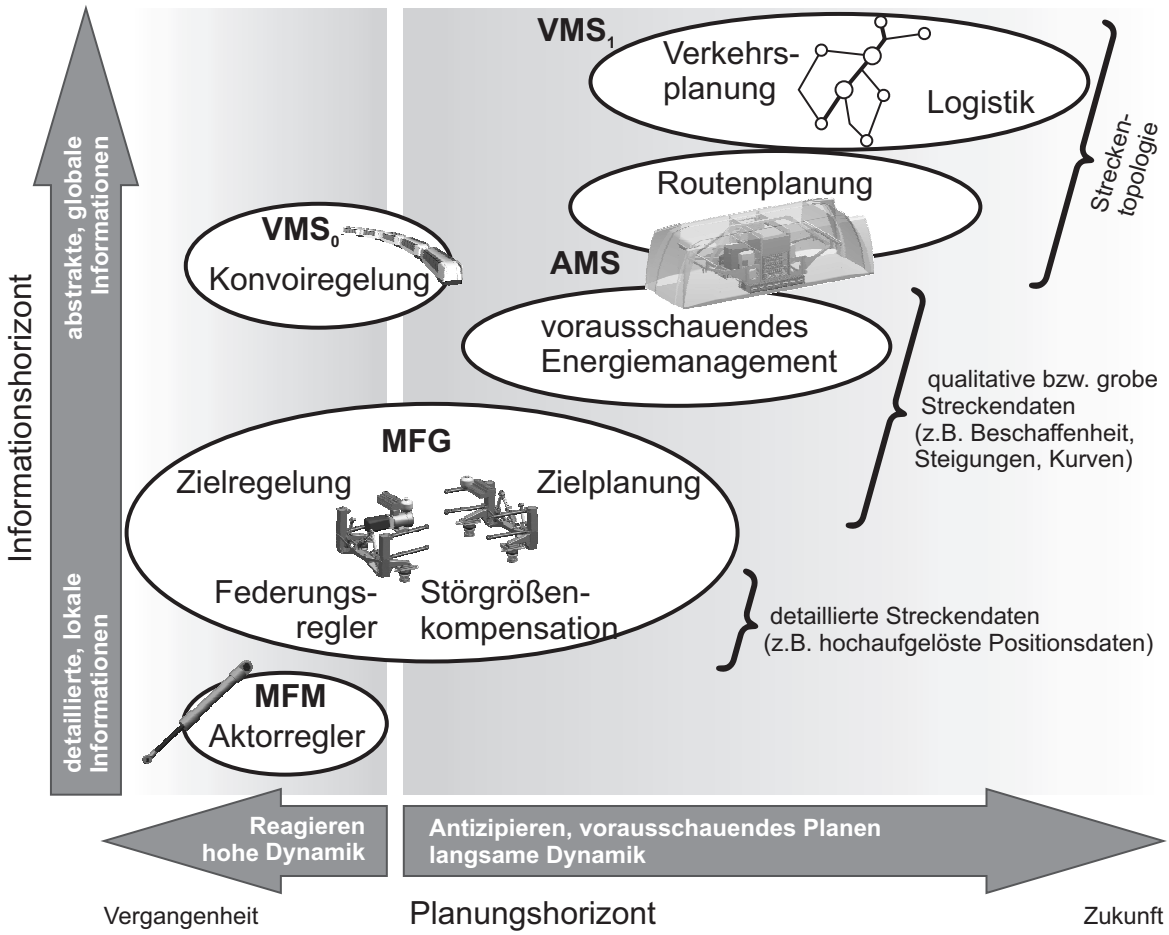


Abbildung 7.2: Einordnung einiger Regelungs- und Planungsverfahren aus verschiedenen Architekturebenen des RailCabs bezüglich Planungs- und Informationshorizont

zont gesprochen werden, da sie im Nachhinein auf Veränderungen ihres Umfeldes, z. B. auf Störungen oder Sollvorgaben, reagieren.

In Abbildung 7.2 sind einige Beispiele für Regelungs- und Planungsysteme des RailCabs aufgeführt.[3] Diese Beispiele stellen Ansatzpunkte für Selbstoptimierung in hierarchischen mechatronischen Systemen dar. Diese Systeme sind qualitativ bezüglich ihres Planungs- und Informationshorizontes eingeordnet.

Der Aktorregler, im Beispiel für einen Hydraulikzylinder, besitzt nur Informationen über den Zylinder selbst, über das Hydraulikventil und ggf. den Versorgungsdruck. Er hat dementsprechend nur eine sehr lokale Sichtweise, der Informationshorizont ist dementsprechend gering. Ein vorausschauendes Energiemanagement, welches die Leistungsflüsse für den vorausliegeden Streckenabschnitt plant, bezieht Informationen wie den aktuellen und geschätzten zukünftigen Leistungsbedarf der verschiedenen Verbraucher in seine Planung mit ein. Dementsprechend umfasst der Informationshorizont die Daten der Energieversorgung, des Energiespeichers und der Verbraucher wie bspw. der MFG. Auf der Ebene der Logistik oder der Verkehrsplanung ist der Informationshorizont auf ein Streckennetz mit einer großen Anzahl an Fahrzeugen ausgedehnt.

In dem Maße wie der Informationshorizont zunimmt, muss auch der Abstraktionsgrad steigen. Um die große Informationsmenge beherrschbar zu machen, muss diese auf das Wesentliche reduziert werden. Viele Daten, die auf den unteren Ebenen von Interesse sind, sind auf den höheren Ebenen zu spezifisch. Zum Beispiel ist die Durchflussmenge eines Hydraulikventils für den entsprechenden Zylinderregler von Bedeutung, für das Energiemanagement ist diese Information jedoch zu detailliert. Es interessiert hier vielmehr der Leistungsbedarf des gesamten Federungssystems. Um dies zu verdeutlichen sind im Bild 7.2 rechts exemplarisch die vom jeweiligen OCM benötigten Informationen über das Schienennetz eingetragen.

Aus diesem Sachverhalt lässt sich eine weitere Anforderung an die verteilte Selbstoptimierung ableiten:

- **Detaillierungsgrad:** Die Prozesse auf den verschiedenen Strukturierungsebenen arbeiten auf unterschiedlich abstrahierten Informationen. Der Datenaustausch zwischen den OCM muss dem Rechnung tragen. Speziell bei einem Datenaustausch mit der überlagerten Ebene ist dies von Bedeutung. Hier muss eine Vereinfachung bzw. Reduktion der auszutauschenden Daten stattfinden, um einen an das Problem angepassten Detaillierungsgrad zu erhalten.

  Die Forderung nach einem angepassten Detaillierungsgrad gilt im Grunde für alle Arten von Informationen. Sie betrifft gleichermaßen Daten aus dem Umfeld des Systems (z. B. Anregungsdaten oder Streckenverläufe), aktuelle Zustände und Messgrößen (z. B. der Geschwindigkeit oder den Ladegrad einer Batterie) oder auch Verhaltensbeschreibungen (wie mathematische Modelle).

[3] Diese Beispiele entstammen zum Teil verschiedenen Publikationen des SFB 614. Für weiterführende Informationen sei u. a. auf [MVH05, KRB+06, GWTD08a, GWTD08b, ADG+08, MAK+08, VT08, DHK+09] und [Klö09] verwiesen.

7.2 Hierarchische Wissensbasis

Um die Anforderungen an eine verteilte Selbstoptimierung zu erfüllen, wird in dieser Arbeit eine über die OCM-Hierarchie verteilte Wissensbasis vorgeschlagen. Die Grundstruktur bildet die Wissensbasis im kognitiven Operator des OCM aus Kapitel 6.3.2. Über Kommunikationskanäle wird in dem verteilten System ein Austausch von „Wissen“ ermöglicht. Dieses Konzept ist geeignet auch die unteren Ebenen, der MFM und MFG, die sich durch starke physikalische Abhängigkeiten auszeichnen, zu behandeln.

Die Kommunikation zwischen den Wissensbasen der OCM ist in Abbildung 7.3 zu sehen. Das „Rückgrat“ der verteilten Wissensbasis bildet ein hierarchisches Modell, dessen Bestandteile den einzelnen *Wissensbasen* zugeordnet sind. Der Informationsaustausch zwischen den einzelnen Bestandteilen dieses Modells findet im Wesentlichen über den Austausch von einfachen Verhaltensmodellen statt. Interne Größen wie die Struktur oder Parameter werden verborgen. Über die ausgetauschten Modelle kann das Verhalten benachbarter Systeme berücksichtigt und die Stabilität und Konvergenz für einen verteilten selbstoptimierenden Prozess gewährleistet werden. Durch die Vereinfachung bzw. Reduktion der Modelle wird der Detaillierungsgrad auf die jeweilige Strukturebene angepasst.

Auf Basis des hierarchischen Modells arbeitet eine hierarchische Optimierung. Diese ermittelt paretooptimale Einstellungen für das System. Die hierarchische Struktur wird bei der Optimierung des Gesamtsystems ausgenutzt, die Optimierung erfolgt verteilt über alle OCM. Im Wesentlichen werden hierbei abstrakte Größen, wie Entwurfspunkte zwischen den Systemen ausgetauscht. Die berechneten Einstellungen (Paretomenge) werden in der Wissensbasis hinterlegt und stehen so Planungsverfahren, Zielregelungen oder anderen Komponenten des OCM zur Verfügung.

Sowohl das hierarchische Modell als auch die Optimierung sind gut skalierbar auf eine

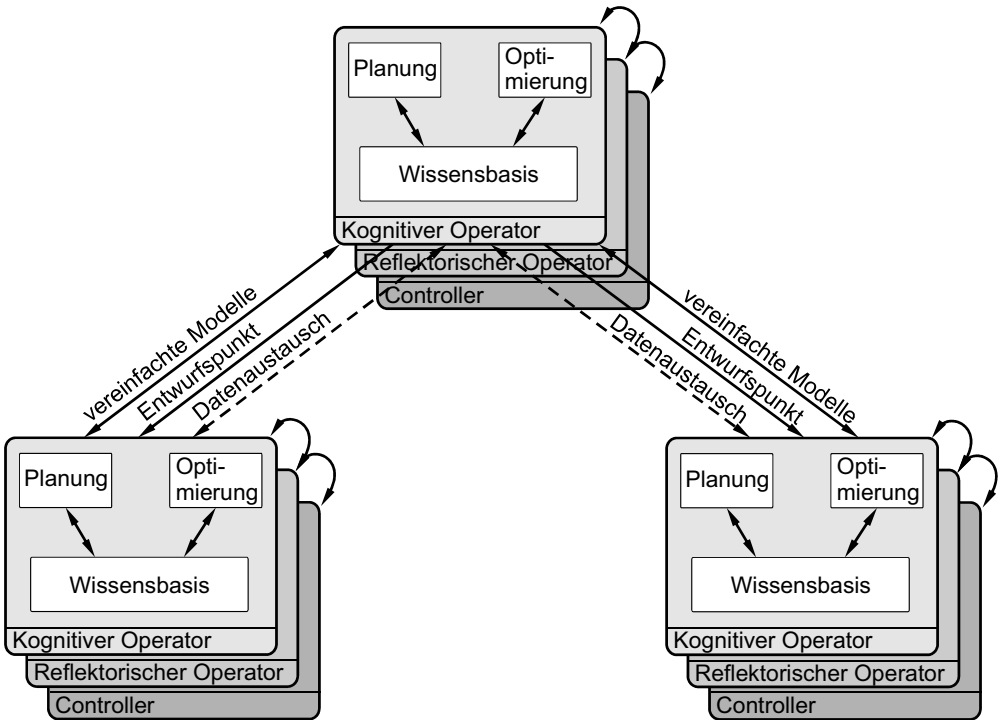


Abbildung 7.3: Kommunikation zwischen den Wissensbasen in der OCM-Architektur

große Anzahl an OCM. Der Datenaustausch findet im Wesentlichen auf der Ebene generischer Verhaltensmodelle und abstrakter Entwurfspunkte (z. B. Zielvorgaben) statt.

Dieses Konzept, ein selbstoptimierendes System über ein hierarchisches Modell und eine darauf aufsetzende hierarchische Optimierung zu realisieren, wurde erstmals in [MAK+08] und [SFB08] vorgestellt.

Gielen und andere stellen in [GME05, RGR07] ein vergleichbares Konzept für die hierarchische Modellierung, Optimierung und Synthese analoger Schaltkreise vor. Das hierarchische Modell setzt ebenfalls auf Modellreduktionstechniken auf. Die Synthese des Systems basiert auf Mehrzieloptimierungen der einzelnen Komponenten. Die Optimierung der jeweils höheren Ebene setzt auf den Paretomengen der unterlagerten Ebene auf. Dieser Vorgang wird von Gielen treffend als *Multi-Objective-Bottom-Up*-Entwurfsmethode bezeichnet.

Das hierarchische Modell und die darauf aufbauende hierarchische Optimierung werden in den folgenden beiden Abschnitten 7.3 und 7.4 näher vorgestellt.

7.3 Hierarchisches Modell

Die Grundidee des hierarchischen Modells soll im Folgenden kurz beschrieben werden. Jedes OCM besitzt a priori „Wissen“ über das Verhalten seines zugehörigen Subsystems. Dieses inhärente Wissen liegt in Form von mathematischen Modellen der Regler und der Regelstrecke vor. Das Verhalten der unterlagerten bzw. überlagerten Subsysteme ist dem OCM von vornherein nicht bekannt. Die Art des Modells richtet sich nach der jeweiligen Strukturebene und der Aufgabe des OCM. Auf den unteren Ebenen, die durch das Schwingungsverhalten geprägt sind, überwiegen vor allem Dynamikmodelle in Form von Differentialgleichungen.

Um das Verhalten unterlagerter Subsysteme zu berücksichtigen, findet zur Laufzeit ein Austausch von „Wissen“ zwischen einem OCM und seinen unterlagerten OCM in Form mathematischer Modelle statt. Eingebunden in das interne Modell ergibt sich so in jedem OCM ein Bild seines eigenen Verhaltens und dem Verhalten seiner Subsysteme. Dieses Prinzip ist in Abbildung 7.4 dargestellt. Das Modell wird im Folgenden als hierarchisches *Bottom-Up-Modell* bezeichnet.

Bei diesem Modellaustausch über die Hierarchieebenen hinweg wird die Modellierungstiefe mit Hilfe von Modellreduktions- oder Approximationstechniken angepasst. So stehen jeweils Modelle mit passendem Detaillierungsgrad für verschiedene Aufgaben wie Optimierung, Planung oder Auslegung von regelungstechnischen Komponenten zur Verfügung.

Durch den Austausch auf Basis vereinfachter Verhaltensmodelle werden die Abhängigkeiten zwischen den OCM reduziert und eine starke Kapselung der OCM erreicht; dynamische Abhängigkeiten werden erst zur Laufzeit ausgewertet.

Im Folgenden kennzeichnet ein hochgestellter Index i die Zugehörigkeit zum OCM i. Die dem OCM i unmittelbar unterlagerten Subsysteme werden über die Indexmenge $\mathcal{U}^i$ angegeben. Die Indizes aller in einem Teilbaum befindlichen OCM wird über die Indexmenge $\mathcal{V}^i$ angegeben, mit dem OCM i als oberstes Element des Teilbaums.

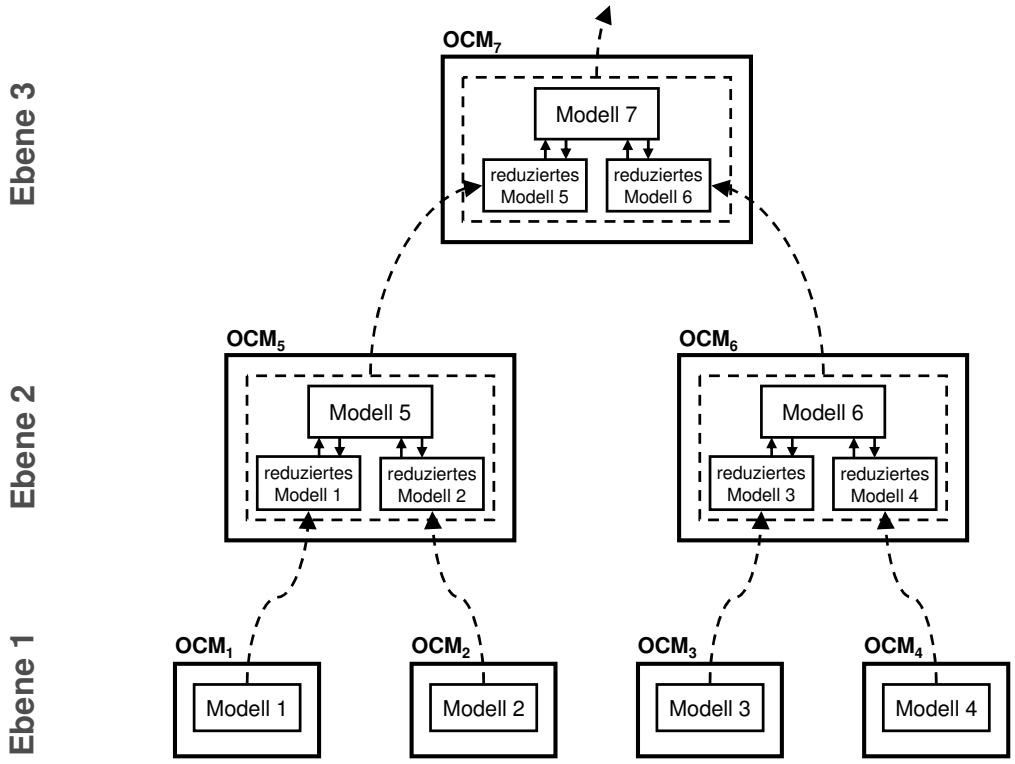


Abbildung 7.4: Austausch reduzierter Modelle im hierarchischen Bottom-Up-Modell

7.3.1 Basismodell

Das „Wissen“ eines OCM über sein eigenes Verhalten liegt in Form eines mathematischen Modells vor, welches das dynamische Verhalten des jeweiligen geregelten Subsystems beschreibt. Das Verhalten benachbarter OCM wird a priori nicht abgebildet. Diese Modelle bilden die Grundlage des hierarchischen Modells und werden aus diesem Grund als *Basismodelle* bezeichnet.

Die Modelle liegen als Differentialgleichungen vor und enthalten alle relevanten Effekte, die zur Beschreibung des jeweiligen Subsystems notwendig sind. Das Basismodell eines OCM i lässt sich über die gewöhnliche Differentialgleichung:

$$\dot{\underline{x}}^i = \underline{\psi}^i(\underline{x}^i, \underline{u}^i, \underline{v}^i, \underline{w}^i, \underline{p}^i, t) \tag{7.1}$$

mit den Ausgangsgleichungen:

$$\begin{aligned} \underline{y}^i &= \underline{\xi}^i(\underline{x}^i, \underline{u}^i, \underline{v}^i, \underline{w}^i, \underline{p}^i, t) \\ \underline{r}^i &= \underline{\vartheta}^i(\underline{x}^i, \underline{u}^i, \underline{v}^i, \underline{w}^i, \underline{p}^i, t) \\ \underline{q}^i &= \underline{\zeta}^i(\underline{x}^i, \underline{u}^i, \underline{v}^i, \underline{w}^i, \underline{p}^i, t) \end{aligned} \tag{7.2}$$

angeben.

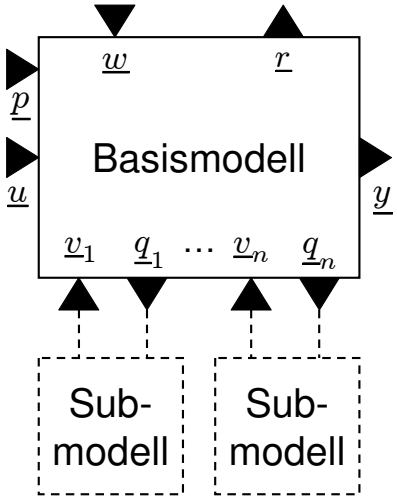


Abbildung 7.5: Schnittstellen des Basismodells

Die Bezeichner in den obigen Gleichungen sind:

$\underline{p}^i$	:	Vektor der veränderlichen Systemparameter, z. B. die Reglerparameter
$\underline{x}^i$	:	Zustandsvektor der Differentialgleichung
$\underline{u}^i$	:	Eingangsvektor des Basismodells
$\underline{y}^i$	:	Ausgangsvektor des Basismodells
$\underline{v}^i$	:	Eingangsvektor zur Kopplung von Submodellen; hauptsächlich physikalische Größen, z. B. Positionen und Geschwindigkeiten
$\underline{w}^i$	:	Eingangsvektor zur Kopplung an ein überlagertes Modell; beinhaltet sowohl physikalische Kopplungsgrößen als auch Stellsignale an den Regler
$\underline{q}^i$	:	Ausgangsvektor zur Kopplung von Submodellen; beinhaltet sowohl physikalische Kopplungsgrößen als auch Stellsignale an den unterlagerten Regler
$\underline{r}^i$	:	Ausgangsvektor zur Kopplung an ein überlagerte Modell; hauptsächlich physikalische Größen, z. B. Kräfte und Momente

Die Schnittstellen des Modells sind in Abbildung 7.5 dargestellt.

Das Basismodell hat zwei Verwendungszwecke:

- Intern im OCM wird das Basismodell genutzt, um andere Modelle für verschiedene Aufgaben wie Optimierungen oder Simulationen aufzubauen. Hierfür werden die Schnittstellen $\underline{u}$ und $\underline{y}$ genutzt.
- In der hierarchischen Architektur dient das Basismodell als Grundlage für den Austausch von Verhaltensmodellen. Das Basismodell wird hierzu über geeignete Mechanismen vereinfacht und an das überlagerte OCM übertragen. Dort wird es als Submodell an das Basismodell angebunden. Die Ankopplung von Submodellen geschieht über die Schnittstelle $\underline{v}_i$ und $\underline{q}_i$, die Anbindung an ein überlagertes Modell über die Schnittstelle $\underline{w}$ und $\underline{r}$

Das vereinfachte Modell eines unterlagerten Systems $j \in \mathcal{U}^i$ wird über die Differentialgleichung:

$$\begin{aligned} \dot{\underline{\tilde{x}}}^j &= \underline{\tilde{\psi}}^j(\underline{\tilde{x}}^j, \underline{w}^j, t) \\ \underline{r}^j &= \underline{\tilde{\vartheta}}^j(\underline{\tilde{x}}^j, \underline{w}^j, t) \end{aligned} \tag{7.3}$$

beschrieben. Die Anbindung der Submodelle geschieht über die Koppelbeziehungen:

$$\begin{aligned} \underline{q}^i &= \left(\underline{w}^j\right)_{j \in \mathcal{U}^i} \\ \underline{v}^i &= \left(\underline{r}^j\right)_{j \in \mathcal{U}^i} \end{aligned} \tag{7.4}$$

Die Gleichungen (7.1) und (7.2) sind abhängig von den Modellen der unterlagerten Subsysteme. Der Aufbau des hierarchischen Modells verläuft wie bereits beschrieben *bottom-up* durch die Strukturebenen der OCM-Architektur. Jedes OCM besitzt hierdurch ein Verhaltensmodell, welches die Gesamtheit der unterlagerten Subsysteme beschreibt.

Als Einstellmöglichkeiten können die Reglerparameter über den Vektor $\underline{p}^i$ gestellt werden. Ebenso lassen sich hierüber Parameter der Regelstrecke bspw. durch Identifikationsverfahren anpassen. Wie in (7.3) zu sehen, sind die Submodelle fix bezüglich ihrer Parameter. Aus Gründen der Kapselung erhält das Basismodell i keinen Zugriff auf die Parameter der Subsysteme $\mathcal{U}^i$. Diese werden in den unterlagerten OCM festgelegt und beim Erstellen des vereinfachten Modells eingefroren.

7.3.2 Modellaustausch

Der Modellaustausch geschieht auf der Grundlage vereinfachter Modelle. Die Art und Weise, wie eine Übertragung der vereinfachten Modelle realisiert werden kann, ist entscheidend von der Modellform abhängig. Je nach Architektur der Rechenhardware ist eine möglichst strukturierte und generische Modellform vorteilhaft. Hierdurch lässt sich das Modell einfach als Datenpaket an andere OCM übermitteln, was die Übertragung über ein Netzwerk erheblich erleichtert. Dieser Aspekt spielt eine untergeordnete Rolle, wenn Sender- und Empfänger-OCM z. B. über einen Shared-Memory-Zugriff verfügen. Das Modell kann hier leicht als beliebiges Objekt zwischen zwei OCM ausgetauscht werden.

Im Folgenden werden einige Modellformen betrachtet.

- **Lineare zeitinvariante Modelle:** LZI-Modelle können leicht über eine Linearisierung gewöhnlicher Differentialgleichungen der Form (5.29) bestimmt werden. Die Darstellung dieser Modelle geschieht im Mehrgrößenfall üblicherweise in Form der Zustandsraumdarstellung (5.25). Bei der Übertragung eines Basismodells an ein überlagertes OCM kommt die Zustandsraumdarstellung:

$$\dot{\underline{x}} = \underline{\mathbf{A}}\,\underline{x} + \underline{\mathbf{B}}\,\underline{w} \tag{7.5}$$

$$\underline{r} = \underline{\mathbf{C}}\,\underline{x} + \underline{\mathbf{D}}\,\underline{w} \tag{7.6}$$

  zum Einsatz.

  Diese Darstellungsformen kann eindeutig über das Tupel

$$\mathcal{M}_b = (\underline{w}_0, \underline{r}_0, \underline{\mathbf{A}}, \underline{\mathbf{B}}, \underline{\mathbf{C}}, \underline{\mathbf{D}}) \tag{7.7}$$

beschrieben werden. Hierbei geben $\underline{w}_0$ und $\underline{r}_0$ den Arbeitspunkt des Basismodells an. Eine Übertragung von $\mathcal{M}_b$ ist mit geringem Aufwand möglich.

LZI-Modelle zeichnen sich durch eine große Verfügbarkeit an unterschiedlichen Reduktionsmethoden aus (vgl. [Har02]). Nachteilig ist die Beschränkung auf die linearen Effekte. Insbesondere Nichtlinearitäten wie Begrenzungen spielen in der Technik eine große Rolle und können mit LZI-Modellen nicht abgebildet werden.

- **Lineare Differenzengleichungen:** Für lineare Differenzengleichungen gelten im Grunde dieselben Aussagen wie für LZI-Modelle. So kann eine Darstellung in der Zustandsraumdarstellung erfolgen:

$$\begin{aligned} \underline{x}_{k+1} &= \mathbf{A}_d\,\underline{x}_k + \mathbf{B}_d\,\underline{w}_k \\ \underline{r}_k &= \mathbf{C}_d\,\underline{x}_k + \mathbf{D}_d\,\underline{w}_k \end{aligned} \tag{7.8}$$

  Bezüglich der Übertragbarkeit und der verfügbaren Verfahren verhalten sich lineare Differenzengleichungen analog zu den LZI-Modellen.

- **Gewöhnliche Differentialgleichungen:** Der Austausch nichtlinearer gewöhnlicher Differentialgleichungen der allgemeinen Form (7.3):

$$\begin{aligned} \dot{\underline{x}} &= \underline{\psi}(\underline{x}, \underline{w}, t) \\ \underline{r} &= \underline{\vartheta}(\underline{x}, \underline{w}, t) \end{aligned}$$

  erfordert einen wesentlich höheren Aufwand. Im Grunde muss hier ein Berechnungsprogramm für die Funktionen $\underline{\psi}$ und $\underline{\vartheta}$ übertragen werden. Dies kann bspw. über spezielle Modellbeschreibungssprachen (z. B. Modelica [Fri03], O-DSL oder DSC [HHR94, Hah99]), in Form eines Interpreterprogramms oder mit gewissen Einschränkungen auch über ausführbaren Maschinencode[4] geschehen.

7.3.3 Modellvereinfachung

Die Ziele der Modellvereinfachung bestehen im Wesentlichen aus den folgenden beiden Punkten:

- Verringerung der Modellordnung
- Beschleunigung der Simulation

Die Verringerung der Modellordnung zielt darauf ab, dass das Modell mit den Ebenen nicht übermäßig anwächst. Beide Punkte gehen ein Stück weit einher. So führt ein kleineres Modell zu weniger Rechenoperationen, die in einem Simulationschritt ausgewertet werden müssen. Einen wesentlich größeren Einfluss übt jedoch die Simulationsschrittweite aus, da eine größere Schrittweite dazu führt, dass insgesamt weniger Simulationsschritte erforderlich sind. Durch bestimmte Reduktionsansätze lassen sich hochdynamische Anteile aus dem Modell entfernen, wodurch die Schrittweite oft in außerordentlichem Maße angehoben werden kann.

[4] Maschinencode beschreibt einen Algorithmus in Form von Prozessoranweisungen und ist somit hardwareabhängig. Die Übertragung erfordert eine genaue Kenntnis der Zielplattform. Gegebenenfalls muss Code für verschiedene Rechenplattformen vorgehalten werden.

Es existiert eine große Bandbreite an Verfahren, mit denen sich ein Modell vereinfachen lässt. So existieren leistungsfähige Verfahren der Ordnungsreduktion, aber auch Identifikationsverfahren und Optimierungsansätze können prinzipiell hierzu genutzt werden. Da das Reduktionsverfahren zur Laufzeit im OCM zum Einsatz kommen soll, müssen die Algorithmen automatisierbar sein und entsprechend schnell arbeiten. Ordnungsreduktionsverfahren sind speziell auf diesen Anwendungsfall zugeschnitten und in den meisten Fällen anderen Verfahren in Bezug auf Rechenzeit und Qualität des vereinfachten Modells überlegen.

Für lineare Systeme sind leistungsfähige Verfahren zur Ordnungsreduktion verfügbar, welche die oben genannten Anforderungen erfüllen. Eine Übersicht über diese Verfahren findet sich in [Har02]. Einige dieser Reduktionsverfahren sind im Anhang A.1 beschrieben.

Grundsätzlich können im hierarchischen Modell aber auch Verfahren für nichtlineare Systeme eingesetzt werden (z. B. das Verfahren von Lohmann [Loh94]). Aufgrund der besonderen Eigenheiten von nichtlinearen Systemen wie z. B.: Amplitudenabhängigkeit, unterschiedliches Lösungsverhalten je nach Anfangsbedingungen, Auftreten von Grenzzyklen, gestaltet sich die Entwicklung von Modellreduktionsverfahren wesentlich schwieriger als bei linearen Systemen. Die verfügbaren Verfahren eignen sich oft nur für bestimmte Problemklassen.

Das vereinfachte Modell soll das Verhalten aller unterlagerten Subsysteme berücksichtigen. Daher bezieht sich der Vorgang der Modellvereinfachung auf das vollständig ausgeprägte Basismodell (siehe Abbildung 7.6). Dieses besteht aus dem Basismodell i und den unterlagerten Submodellen $\mathcal{U}^i$ mit den Koppelbeziehungen (also Gl. (7.1), (7.2), (7.3) und (7.4)).

Der gesamte Prozess vom nichtlinearen hin zum reduzierten Modell $\mathcal{M}_b^i$ läuft, wie in Abbildung 7.7 dargestellt, in mehreren Schritten ab. Zunächst müssen der Parametersatz und Arbeitspunkt, für den das Modell generiert werden soll, festgelegt werden. Parameter und Arbeitspunkt werden vom lokalen OCM gesetzt[5].

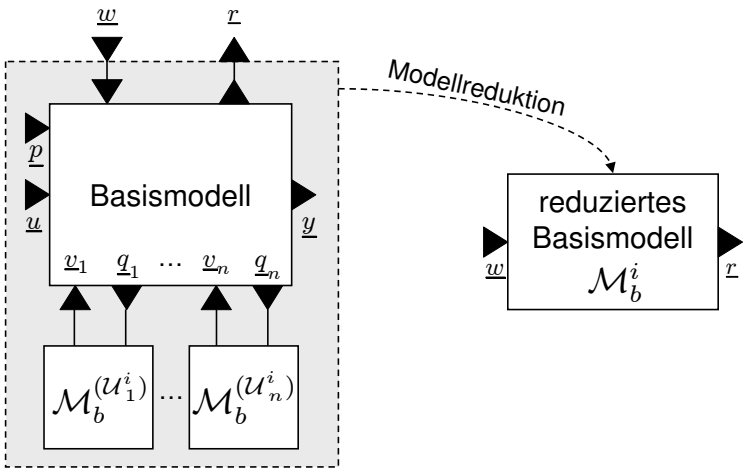


Abbildung 7.6: Schnittstellen des ursprünglichen und des reduzierten Basismodells

[5] Für den Arbeitspunkt ist in vielen Fällen die Bestimmung eines stationären Zustandes $\underline{x}_\infty$ bei einem vorgegebenen Eingang $\underline{u}_\infty$ mit $\underline{\psi}(\underline{x}_\infty, \underline{u}_\infty) = 0$ gefordert. Die Berechnung kann z. B. mit Hilfe von Verfahren zum Lösen von Quadratminimierungsproblemen, z. B. das *Gauß-Newton-Verfahren* oder das *Levenberg-Marquardt-Verfahren* [GMW81] effizient geschehen. Über die Lösbarkeit des Problems können jedoch keine allgemeingültigen Aussagen gemacht werden (vgl. [Föl94]). Sie muss im Einzelfall

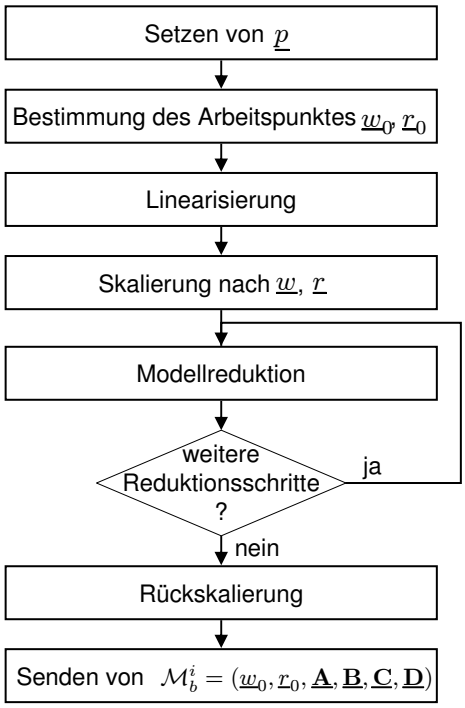


Abbildung 7.7: Ablaufdiagramm der Modellreduktion

Durch die Linearisierung um den Arbeitspunkt wird das gekoppelte Modell in eine lineare Modellbeschreibung überführt. Die Linearisierung stellt den ersten Vereinfachungsschritt dar, da hier die nichtlinearen Effekte des ursprünglichen Basismodells eliminiert werden.

Vor der eigentlichen Modellreduktion muss eine Skalierung bezüglich der Ein- und Ausgänge des Systems stattfinden. Diese Skalierung ist unbedingt erforderlich, wenn sich die Elemente der Vektoren $\underline{w}$ und $\underline{r}$ des linearisierten Systems in unterschiedlichen Größenordnungen bewegen. Die Skalierung wird über eine Lineartransformation des Modells erreicht, die Rückskalierung geschieht entsprechend über die Rücktransformation.

Über die Skalierungsfaktoren kann zudem eine Gewichtung realisiert werden, über die die Approximationsgüte der Eingangs-Ausgangspfade justiert werden kann. Dies ist bspw. sinnvoll, um Ausgangsgrößen, die das Bewegungsverhalten des überlagerten Systems beeinflussen, besonders genau zu approximieren, während der Verlauf von Größen, die nur zur Bewertung berechnet werden, wie z. B. der Leistungsbedarf, nur grundsätzlich wiedergegeben werden.

Anschließend erfolgt eine Reduktion der Modellordnung. Bei der Reduktion kann es sinnvoll sein, verschiedene Verfahren sequentiell auf das lineare System anzuwenden, wodurch die individuellen Stärken und Eigenschaften der verschiedenen Verfahren gezielt genutzt werden können.

7.3.4 Umgebungsmodell

Das Basismodell beschreibt das geregelte Verhalten des zugehörigen Teilsystems inklusive des Verhaltens der unterlagerten Subsysteme. Um das Basismodell an verschiedene Aufgaben, wie z. B. Simulationen oder Optimierungen, anzupassen, muss es in der Regel um

anhand eines konkreten Systems überprüft werden.

weitere Teilmodelle ergänzt werden. Für den Einsatz in einer modellbasierten Optimierung wird zusätzlich ein *Anregungs-* und *Bewertungsmodell* sowie ein *Umgebungsmodell* benötigt. Das resultierende Optimierungsmodell ist in Abbildung 7.8 dargestellt.

Anregungs- und *Bewertungsmodell* erweitern das Modell um Störanregungen und Referenzsignale sowie um Funktionen zur Berechnung der Zielgrößen (vgl. Kapitel 4.1). Das *Umgebungsmodell* beschreibt in stark vereinfachter Form das Verhalten der überlagerten Systembestandteile. Hierdurch wird das Verhalten der Umgebung und die Wechselwirkung der überlagerten Systeme berücksichtigt. Während Anregungen nicht beeinflusst werden können, also im regelungstechnischen Sinne nicht steuerbar sind, reagieren benachbarte Systeme jedoch in deterministischer Weise auf Aktionen des Systems. Hieraus können Instabilitäten oder andere negative Effekte resultieren, die erst bei der Ankopplung an das überlagerte System entstehen. Diese Effekte fließen über das Umgebungsmodell in die Optimierung ein.

Bei bestimmten Modellierungsansätzen ist das Umgebungsmodell ein notwendiger Bestandteil für eine Simulation, da das Basismodell in diesen Fällen ohne Umgebungsmodell nicht lauffähig ist. Ein Beispiel hierfür ist ein Basismodell, welches einen Kraftsteller beschreibt. Eine Simulation ist nur dann sinnvoll, wenn ein Gegenpart, eine Masse, existiert.

Das Umgebungsmodell für ein OCM i lässt sich über die Differentialgleichung:

$$\begin{aligned} \dot{\underline{x}}_u^i &= \underline{\psi}_u^i(\underline{x}_u^i, \underline{v}_u^i, t) \\ \underline{q}_u^i &= \underline{\zeta}_u^i(\underline{x}_u^i, \underline{v}_u^i, t) \end{aligned} \tag{7.9}$$

ausdrücken. Die Anbindung an das Basismodell, Gl. (7.1) und (7.2), geschieht über die Koppelbeziehungen:

$$\begin{aligned} \underline{w}^i &= \underline{q}_u^i \\ \underline{r}^i &= \underline{v}_u^i \end{aligned} \tag{7.10}$$

Bei einer Optimierung sind die umliegenden Systeme nicht Gegenstand der Optimierung. Es soll lediglich ihr grundsätzliches Verhalten berücksichtigt werden. In den meisten Fällen ist es daher ausreichend, stark reduzierte Umgebungsmodelle zu verwenden. Bei der Optimierung bspw. eines Aktors kann es ausreichend sein, diesen gegen z. B. ein Feder-Masse System arbeiten zu lassen, welches in etwa die Einbauverhältnisse des Aktors widerspiegelt.

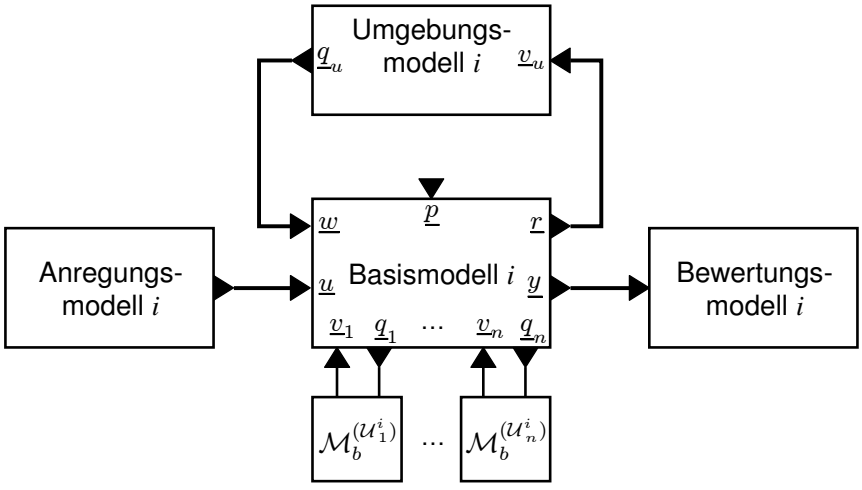


Abbildung 7.8: Basismodell erweitert zu einem Optimierungsmodell

Das Umgebungsmodell kann auf unterschiedliche Weise bereitgestellt werden. Bei Entwurf des Systems können neben den Basismodellen auch die Umgebungsmodelle erstellt und im OCM hinterlegt werden. Hierdurch lässt sich der Detaillierungsgrad speziell auf das Teilsystem abstimmen. Eine andere Möglichkeit ist eine automatische Generierung der Umgebungsmodelle. Dies kann ähnlich zu der Vorgehensweise beim hierarchischen Bottom-Up-Modell geschehen, jedoch muss hier der Generierungsablauf Top-Down von höheren Ebenen hin zu den unteren gerichtet sein.

7.3.5 Hierarchisches Umgebungsmodell

Die Generierung des hierarchischen Umgebungsmodells setzt auf dem bisher beschriebenen Bottom-Up-Modell auf. Über die bereits ausgeprägten Basismodelle ist das Verhalten eines kompletten Baums der Aggregatstruktur bekannt. Das oberste Element dieses Baumes bildet den Ausgangspunkt. Voraussetzung für dieses System ist, dass es auch ohne Eingaben von außen für sich lauffähig ist. Üblicherweise handelt es sich hierbei um ein autonomes System, ein AMS, oder ein teilautonomes System, z. B. ein MFG. Die Generierung der Umgebungsmodelle beginnt bei dieser obersten Ebene und verläuft anschließend Top-Down bis hin zu den untersten MFM, daher kann dieses Umgebungsmodell auch als Top-Down-Modell bezeichnet werden. Abbildung 7.9 zeigt diese Vorgehensweise beispielhaft für ein OCM der 2. Strukturebene.

Der Modellaustausch und die Modellvereinfachung kann auf dieselbe Art und Weise wie beim Bottom-Up-Modell ablaufen (vgl. Abschnitt 7.3.2 und 7.3.3). Jedoch gibt es einen großen Unterschied bei der Wahl der Schnittstellen und beim Bezugsrahmen, den die Umgebungsmodelle abbilden. In einem OCM i muss für jedes unterlagerte Subsystem $j \in \mathcal{U}^i$ ein eigenes Umgebungsmodell $\mathcal{M}_u^j$ generiert werden. Dieses Umgebungsmodell wird aus den verkoppelten Modellen: dem Basismodell, dem evtl. vorhandenen Umgebungsmodell der nächsthöheren Ebene $\mathcal{M}_u^i$ sowie allen benachbarten Submodellen $\mathcal{M}^k$ für alle $k \in \mathcal{U}^i$, $k \neq j$

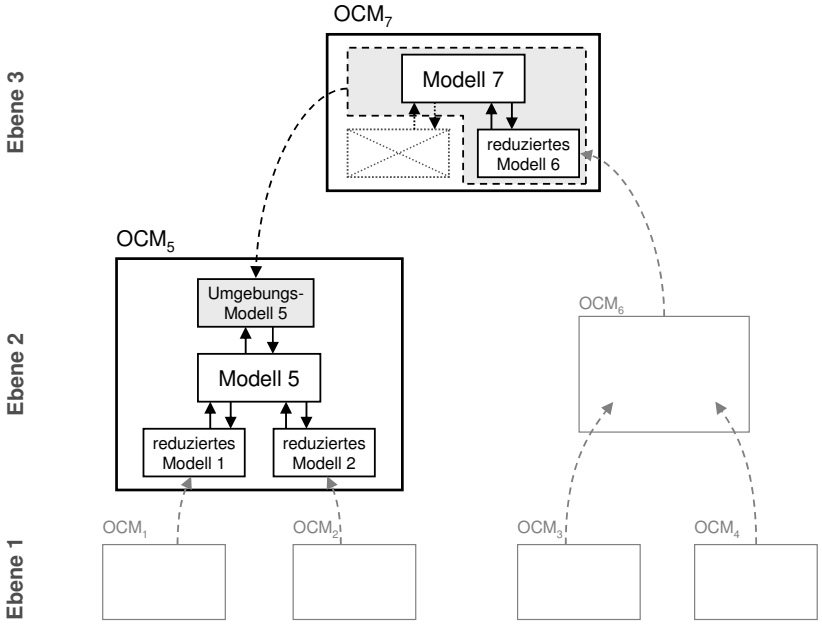


Abbildung 7.9: Austausch von Umgebungsmodellen im Top-Down-Modell

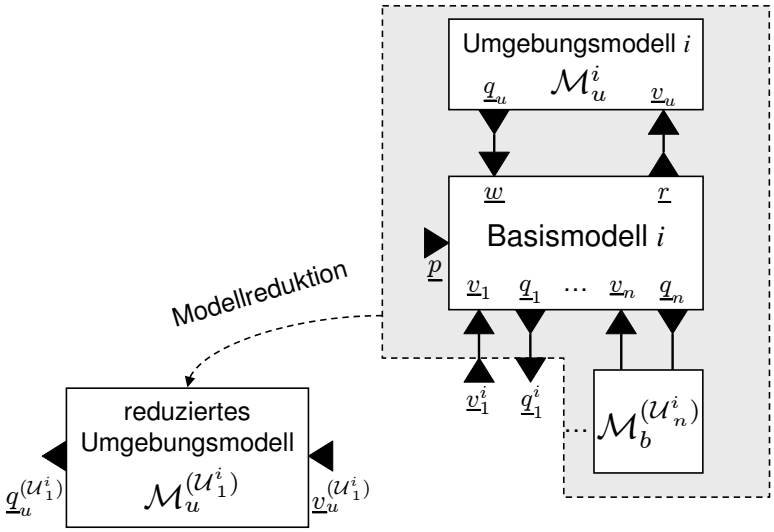


Abbildung 7.10: Erzeugung des reduzierten Umgebungsmodells für das erste unterlagerte Subsystem

gebildet.

Abbildung 7.10 zeigt die Schnittstelle und den Bezugsrahmen des Umgebungsmodells. Die Schnittstelle des Umgebungsmodells ergibt sich wie dargestellt aus dem Eingangs- und Ausgangsvektor über die das Submodell j normalerweise an das Basismodell angekoppelt wird. Die Ein- und Ausgänge $\underline{w}$ und $\underline{r}$ werden mit dem Umgebungsmodell aus der nächsthöheren Ebene gekoppelt und sind damit vollständig definiert. Das Verhalten benachbarter Teilbäume wird über die anderen Submodelle abgebildet und geht so in das Umgebungsmodell ein.

Das Optimierungsmodell in Abbildung 7.8 kann so auf jeder Ebene der OCM-Architektur aufgebaut werden. Es beinhaltet sowohl das physikalische als auch das regelungstechnische Verhalten der überlagerten Systeme. Um bspw. bei einer Optimierung nur das ungeregelte Verhalten, also nur die physikalischen Kopplungen, zu berücksichtigen, kann eine Deaktivierung der Regler über eine entsprechende Parametrierung erfolgen.

7.4 Hierarchische Mehrzieloptimierung

In diesem Abschnitt wird gezeigt, wie eine Mehrzieloptimierung auf der Grundlage des hierarchischen Modells durchgeführt werden kann. Die Aufgabe einer solchen Optimierung ist die Berechnung optimaler Systemeinstellungen für das hierarchische Gesamtsystem. Die Ergebnisse werden in Form von Paretomengen in der verteilten Wissensbasis gespeichert und stehen so anderen Prozessen zur Verfügung.

Prinzipiell besitzt jedes OCM eine eigene regelnde Informationsverarbeitung, deren Parameter im Rahmen einer Optimierung angepasst werden. Ebenso verfolgt jedes OCM eigene Ziele. Für jedes Modul kann daher ein Parametersatz $\underline{p}^i$ und Zielgrößen $\underline{f}^i$ angegeben werden.

Ziel der Optimierung ist die gleichzeitige Minimierung all dieser Zielgrößen. Eine solche Optimierung führt jedoch auf die Berechnung einer hochdimensionalen Paretomenge, die alle Zielgrößen aller Module einschließt. Dieses Problem kann nur mit extrem hohem Aufwand gelöst werden. Abhilfe kann durch eine Einschränkung des Optimierungsproblems auf eine Untermenge dieser umfassenden Lösung geschaffen werden. Dieses eingeschränkte Problem muss sich zum einen leicht formulieren lassen und zu brauchbaren Ergebnissen führen. Zum anderen gelten die Anforderungen, die im Abschnitt 7.1 an die verteilte Wissensbasis gestellt wurden, auch für die hierarchische Optimierung. Während die *Stabilität* und der *Detaillierungsgrad* bereits über die Abhängigkeiten im Bottom-Up-Modell und den Austausch reduzierter Modelle Berücksichtigung finden, müssen insbesondere die *Modularität* und die *Skalierbarkeit* durch den Optimierungsansatz gewährleistet sein.

Im Rahmen dieser Arbeit wurden zwei Ansätze untersucht, die diese Fragestellung auf unterschiedliche Art und Weise angehen und dabei die hierarchische Struktur des Modells ausnutzen: die *Top-Level*-Optimierung und die *Multi-Objective-Bottom-Up*-Optimierung.

Bei der *Top-Level-Optimierung* handelt es sich um einen zentralen Optimierungsansatz, der auf der obersten Ebene der betrachteten Hierarchie aufsetzt. Bei diesem Ansatz ist die Kapselung des OCM nicht so stark ausgeprägt wie bei dem zweiten Ansatz, der *Multi-Objective-Bottom-Up-Optimierung* (MOBU). Bei diesem Ansatz findet die Optimierung des Gesamtsystems dezentral über die OCM verteilt statt. Diese Art der Optimierung wurde in [GME05] und [MAK+08] unabhängig voneinander vorgestellt.

7.4.1 Top-Level-Optimierung

Bei dieser Methode ist der Optimierungsprozess dem Modul auf der obersten Hierarchieebene zugeordnet. Zumeist sind die Zielgrößen der obersten Ebene entscheidend für die Qualität des Gesamtsystems, dementsprechend werden hier nur diese Zielfunktionen betrachtet. Die Zielgrößen der unterlagerten Systeme werden als Nebenbedingungen in das Optimierungsproblem eingeführt.

Aus Optimierungssicht ist dies ein beschränktes, nichtlineares Optimierungsproblem. Mit den Zielfunktionen des obersten Moduls $\underline{f}^0$ und den Zielfunktionen $\underline{f}^k$ der unterlagerten Module kann dieses Optimierungsproblem folgendermaßen angegeben werden:

$$\begin{aligned} &\min_{\underline{p}^i} \underline{f}^0\left(\left(\underline{p}^i\right)_{i\in\mathcal{V}^0}\right) \\ &\underline{f}^k\left(\left(\underline{p}^i\right)_{i\in\mathcal{V}^k}\right) < \underline{b}^k \\ &\text{für alle} \quad k \in \mathcal{V}^0\,,\, k \neq 0 \end{aligned} \tag{7.11}$$

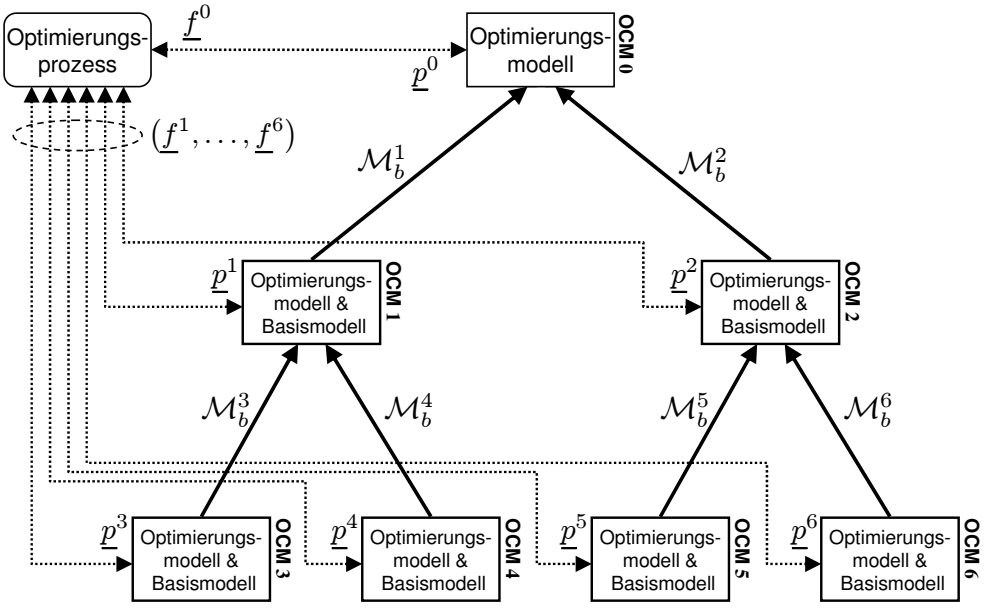


Abbildung 7.11: Struktur der Top-Level-Optimierung

Die Indexmenge $\mathcal{V}^k$ enthält hierbei den Index des betrachteten Moduls k und die Indizes aller in dem Teilbaum unterhalb von k befindlichen Module. Die Nebenbedingungen $\underline{f}^k$ hängen nur von den Parametern der Module in der Indexmenge $\mathcal{V}^k$ ab. Diese Abhängigkeit folgt aus dem Modellaustausch, der von den unteren hin zu den oberen Ebenen verläuft. Der Vektor $\underline{b}^k$ bezeichnet die oberen Schranken für die Zielgrößen des k-ten Moduls. Mit Hilfe dieser Schranken kann eine Mindestgüte für jedes Modul vorgegeben werden. Die Wahl geeigneter Schranken ist unter Umständen problematisch, da allzu restriktiv gewählte Schranken zu unlösbaren Optimierungsproblemen führen können.

Die Berechnung der Zielfunktionen und der Nebenbedingungen $\left(\underline{f}^i\right)_{i\in\mathcal{V}^0}$ läuft in zwei Schritten ab:

1. Setzen der Parameter $\left(\underline{p}^i\right)_{i\in\mathcal{V}^0}$ und Bestimmung der Submodelle $\left(\mathcal{M}_b^k\right)_{\substack{k\in\mathcal{V}^0\\k\neq 0}}$.

2. Parallele Berechnung der Zielgrößen und Nebenbedingungen $\left(\underline{f}^i\right)_{i\in\mathcal{V}^0}$

Nach dem ersten Schritt sind die Basismodelle aller Subsysteme bekannt. Im zweiten Schritt können die Zielfunktion und die Nebenbedingungen mit Hilfe der Optimierungsmodelle bestimmt werden. Es findet vor jeder Zielfunktionsauswertung ein Modellaustausch zwischen den Subsystemen statt, um die Änderungen in den Entwurfsparametern der unterlagerten Subsysteme zu berücksichtigen. Bild 7.11 zeigt den Kommunikationsfluss bei diesem zentralen Optimierungsansatz an einem Beispiel mit 7 Subsystemen auf 3 Ebenen.

Die Entwurfsparameter müssen bei diesem zentralen Ansatz vom Optimierer direkt an alle unterlagerten Subsysteme übermittelt werden. Dies erfordert zum einen Kenntnis von der Struktur des gesamten Systems, zum anderen müssen die Parameter der einzelnen Subsysteme dem Optimierer offen gelegt werden. Insbesondere eine hohe Anzahl an Parametern kann sich negativ auf die Konvergenzrate auswirken.

7.4.2 Multi-Objective-Bottom-Up-Optimierung

Bei diesem Optimierungsansatz wird das gesamte Problem in mehrere kleinere, weniger komplexe Optimierungen zerlegt, die dezentral in der OCM-Architektur verteilt werden. Jedes OCM löst dabei sein eigenes Optimierungsproblem. Anders als bei dem vorangegangenen Top-Level-Ansatz ist nicht der gesamte zulässige Parameterraum für die Optimierung der Teilsysteme freigegeben. Vielmehr beschränkt sich die Optimierung auf die Auswahl von Paretopunkten für die Subsysteme. Die Optimierung „arbeitet“ quasi auf den Paretomengen der direkt unterlagerten Systeme. Wegen der Abhängigkeit von diesen Paretomengen verläuft die Optimierung des gesamten Systems Bottom-Up durch die Strukturierungsebenen. Mit der Optimierung einer höheren Ebene kann erst begonnen werden, wenn die Paretomengen der Ebene darunter vorliegen.

Der Optimierungsprozess innerhalb eines OCM ist in Abbildung 7.12 dargestellt. Der Optimierer eines OCM i berücksichtigt nur die „lokalen“ Zielfunktionen und Nebenbedingungen $\underline{f}^i$ und $\underline{g}^i$. Seine Einflussmöglichkeiten bestehen zum einen in den „lokalen“ Entwurfsparametern $\underline{p}^i$ und zum anderen in den Zielvorgaben $\underline{\alpha}^j$, die an die unterlagerten OCM übermittelt werden. Über diese Zielvorgaben werden die Ziele des OCM auf die Ziele der unterlagerten OCM abgebildet. Die Kommunikation mit den direkt unterlagerten OCM $\mathcal{U}^i$ beschränkt sich auf die Übertragung der Zielvorgaben α^j und der reduzierten Modelle $\mathcal{M}_b^j$ mit $j \in \mathcal{U}^i$. Die Indexmenge $\mathcal{U}^i$ enthält hier die Indizes der dem OCM i direkt unterlagerten Module.

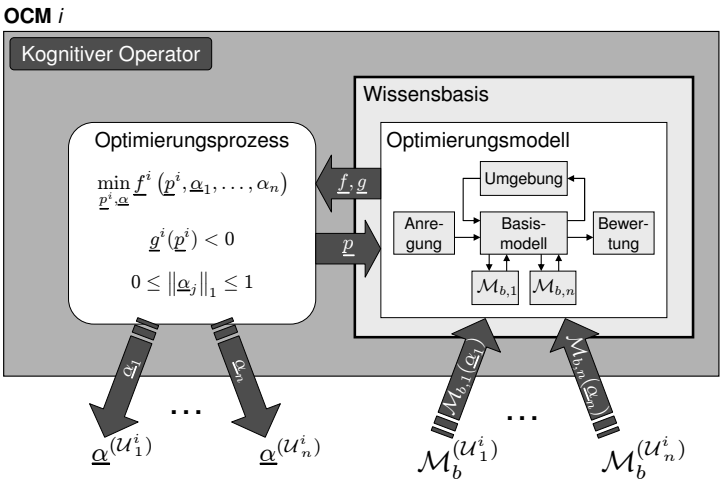


Abbildung 7.12: Optimierungsprozess der MOBU-Optimierung

Das Optimierungsproblem für das OCM i kann somit formuliert werden als:

$$\begin{aligned} &\min_{\underline{p}^i,\underline{\alpha}^j} \underline{f}^i\left(\underline{p}^i,\left(\underline{\alpha}^j\right)_{j\in\mathcal{U}^i}\right) \\ &\underline{g}^i\left(\underline{p}^i,\left(\underline{\alpha}^j\right)_{j\in\mathcal{U}^i}\right) \le \underline{0} \end{aligned} \tag{7.12}$$

Der Vektor α^j adressiert einen Punkt der Paretomenge des unterlagerten OCM j. In diesem OCM findet über die Urbildmenge eine Abbildung: $\underline{\alpha}^j \rightarrow \underline{p}^j, (\underline{\alpha}^k)_{k\in\mathcal{U}^j}$ auf die lokalen Parameter und die Ziele der OCM $\mathcal{U}^j$ der nächstniedrigeren Ebene statt (siehe Abbildung 7.13).

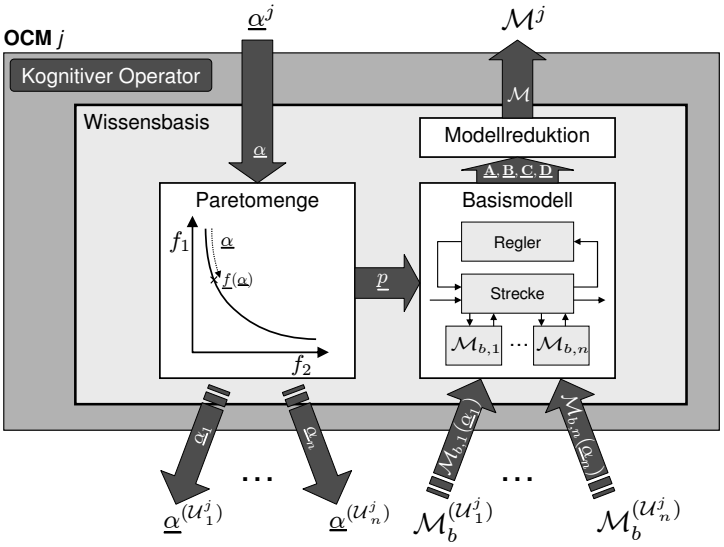


Abbildung 7.13: Kommunikationsfluss im kognitiven Operator

Die Kommunikation zwischen den OCM dient bei der MOBU-Methode der vollständigen Ausprägung des Optimierungsmodells des aktuell zu optimierenden OCM (siehe Abbildung 7.14). Die Berechnung der Zielgrößen und Nebenbedingungen $\underline{f}$ und $\underline{g}$ verläuft in drei Schritten:

1. Propagieren der Zielvorgaben $\underline{\alpha}$ und Bestimmung der Parameter $\underline{p}$ in jedem beteiligten OCM (Top-Down)

2. Ausprägung der Basismodelle und anschließende Parametrierung des Optimierungsmodells (Bottom-Up).

3. Simulation des Optimierungsmodells und Berechnung der Zielgrößen und Nebenbedingungen $\underline{f}$ und $\underline{g}$

Der Informationsaustausch zwischen den OCM ist auf die Übermittlung von Zielvorgaben und Verhaltensmodellen beschränkt. Ein direkter Zugriff auf die Entwurfsparameter oder die Zielgrößen anderer OCM findet nicht statt. Die Übertragung von Parametern wird durch die wesentlich abstraktere Vorgabe von Zielen realisiert. Hierdurch wird die ausgetauschte Information verallgemeinert und so eine starke Kapselung der OCM erreicht.

Der MOBU-Ansatz überträgt die ingenieurmäßige Vorgehensweise bei der Auslegung eines Systems auf die Optimierung. Ein solches kaskadiertes regelungstechnisches System wird üblicherweise beginnend mit der innersten Kaskade, also der untersten Ebene, ausgelegt. Über zahlreiche Iterationen muss das Verhalten der unterlagerten Systeme an das Gesamtsystem angepasst werden. Anders als bei diesem klassischen Entwurf besteht beim MOBU-Entwurf von vornherein die Möglichkeit, die Subsysteme durch Auswahl eines neuen Entwurfspunktes an das übergeordnete Systemverhalten anzupassen. Die Auslegung der höheren Ebenen setzt nicht auf einer einzelnen Lösung, sondern auf einer Lösungsmenge für die unteren Systeme auf.

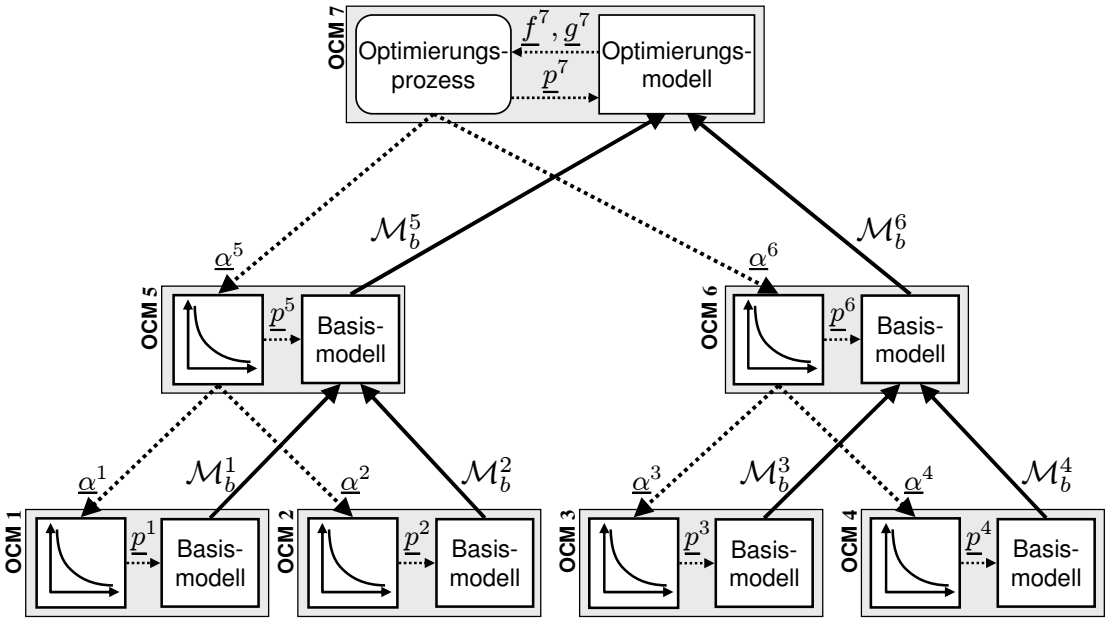


Abbildung 7.14: Zusammenspiel der MOBU-Optimierung und dem hierarchischen Modell

7.4.3 Gegenüberstellung der Optimierungsmethoden

Der Lösungsraum ist bei der MOBU-Methode wesentlich stärker eingeschränkt als bei der Top-Level-Methode. Abbildung 7.15 stellt beispielhaft die Lösungsräume der beiden Optimierungsmethoden gegenüber. In dem Diagramm sind die Bildräume der verschiedenen Module zu sehen. Die schwarzen Linien kennzeichnen die Paretofronten der jeweiligen OCM. Sie geben den Lösungsraum der MOBU-Methode an. Die grau hinterlegten Flächen kennzeichnen die zulässigen Bereiche, die den Lösungsraum der Top-Level-Optimierung darstellen.

Wegen des größeren Lösungsraums lassen sich für die Ziele der obersten Ebene bei der Top-Level-Optimierung bessere Ergebnisse erwarten. Für die Ziele aller weiteren OCM werden lediglich Schranken berücksichtigt. Dies kann im Extremfall dazu führen, dass ein Subsystem isoliert betrachtet eine unbrauchbare Regelgüte, z. B. ein instabiles Verhalten, aufweist. Der Test des einzelnen Subsystems auf einem Prüfstand gestaltet sich mit solchen Einstellungen schwierig.

Bei der MOBU-Optimierung werden die Ziele auf jeder Ebene unter Berücksichtigung der unterlagerten Systeme optimiert. Das System lässt sich so Schritt für Schritt aufbauen und testen. Die Ergebnisse sind auf jeder Ebene für sich plausibel.

Für die grobe Abschätzung der Skalierbarkeit bzgl. Rechenzeit sind vor allem die folgenden drei Aspekte der beiden Methoden relevant:

- Der Aufwand für das Setzen der Parameter und das Übertragen der Modelle
- Der Aufwand für die Berechnung der Zielfunktionen
- Die Ordnung des Optimierungsproblems

Beide Optimierungsansätze setzen auf dem hierarchischen Modell auf und sind bezüglich der Übertragung der Parameter und der Modelle vergleichbar. Der Aufwand hierfür steigt linear mit der Anzahl der Strukturebenen an.

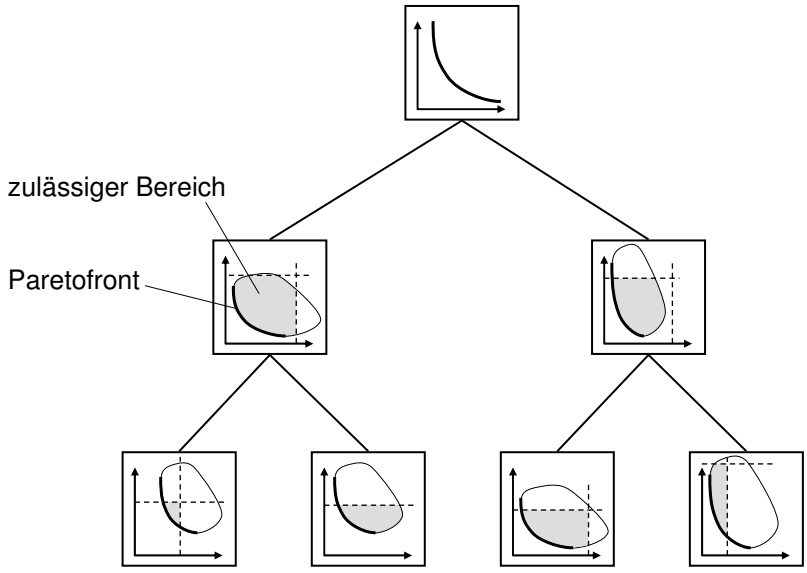


Abbildung 7.15: Gegenüberstellung der beiden Ansätze: Die Top-Level-Optimierung arbeitet auf den zulässigen Bereichen der unterlagerten Ebenen, die MOBU-Optimierung ist auf die Paretofronten beschränkt.

Die Berechnungen der Zielfunktionen und Nebenbedingungen können bei der Top-Level-Methode vollständig parallel ausgeführt werden. Diese Methode berücksichtigt im Optimierungsverfahren jedoch alle direkt und indirekt unterlagerten Subsysteme. Dies führt dazu, dass die Ordnung des Optimierungsproblems, in diesem Fall die Anzahl der Parameter und Nebenbedingungen, mit den Strukturebenen exponentiell[6] ansteigt.

Die MOBU-Methode betrachtet auf jeder Ebene nur den Zielfunktionsvektor des aktuellen Systems. Der Einsatz von Modellreduktionsansätzen kann auf den höheren Ebenen sogar dazu führen, dass der Rechenaufwand zur Berechnung der Zielfunktionen sinkt. Allerdings bauen die Optimierungen der verschiedenen Ebenen aufeinander auf und müssen sequentiell durchgeführt werden. Die Anzahl der sequentiell durchzuführenden Optimierungen steigt linear mit den Ebenen an. Optimierungen auf der selben Ebene können parallel ausgeführt werden. Bei der MOBU-Methode werden auf jeder Ebene nur die Parameter und Ziele des jeweiligen Systems und die Zielvorgaben der direkt unterlagerten Systeme betrachtet. Die Ordnung des Optimierungsproblems steigt daher mit den Strukturebenen nur im Rahmen der Komplexitätszunahme der Regelungsstruktur an.

Fasst man diese Aussagen zusammen, so liegt bei der Top-Level-Methode der kritische Faktor in dem exponentiellen Anstieg der Ordnung des Optimierungsproblems. Die MOBU-Methode skaliert dagegen linear mit den Ebenen. Aus diesem Grund und wegen der starken Kapselung wird im weiteren Verlauf der Arbeit nur die MOBU-Methode betrachtet. Für eine Gegenüberstellung der Optimierungsergebnisse wird an dieser Stelle auf [MAK+08] verwiesen.

[6] Der Anstieg verhält sich linear zu der Anzahl der OCM. Aufgrund der Baumstruktur wird hier ein exponentieller Anstieg mit der Anzahl an Ebenen angenommen.

8 Hierarchisches Modell und Optimierung des Unterflur-Federungsprüfstandes

Das im vorangegangenen Kapitel 7 beschriebene Konzept des hierarchischen Bottom-Up-Modells und der darauf aufbauenden MOBU-Optimierung wird im Folgenden anhand des Prüfstandes eines neuartigen Unterflur-Feder-Neige-Systems vorgestellt. Ziel der Optimierung ist die Bestimmung paretooptimaler Reglereinstellungen unter Berücksichtigung des Federungskomforts und des dafür notwendigen Leistungsbedarfs.

Dieses „neue“ System besitzt den Vorteil, dass es flach aufgebaut ist und einen großen Nutzraum ermöglicht. Da der Fahrgastraum oder die Ladefläche einen ebenen Boden voraussetzen, sollte die Technik möglichst flach unter der Nutzlast, also Unterflur, angeordnet sein. Die flache Bauweise wird durch horizontal in Längsrichtung angeordnete Zylinder erreicht, die mit Hilfe einer Umlenkkinematik Kräfte in horizontaler und vertikaler Richtung stellen können. Demgegenüber schränkt das „alte“ System, welches im derzeitigen Versuchsfahrzeug verbaut ist, wegen der vertikal angeordneten Zylinder den vorhandenen Nutzraum stärker ein (vgl. Abbildung 8.1).

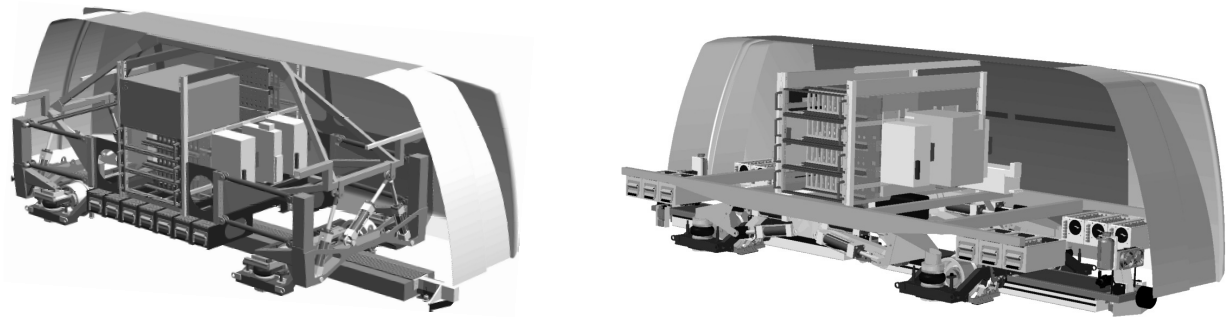


Abbildung 8.1: Gegenüberstellung des bisherigen und des neuen Unterflur-Feder-Neige-Systems im RailCab

8.1 Aufbau des Prüfstandes

Bei dem Unterflur-Feder-Neige-Prüfstand handelt sich um einen Halbfahrzeugprüfstand im Maßstab 1 : 2,5. Die Freiheitsgrade des Prüfstandes erlauben eine Federung und Dämpfung des Aufbaus in vertikaler Richtung z, quer zur Fahrtrichtung y und um die Drehachse φ längs zum Fahrzeug (Wanken). Die Drehung um die Hochachse (Gieren), das Drehen um

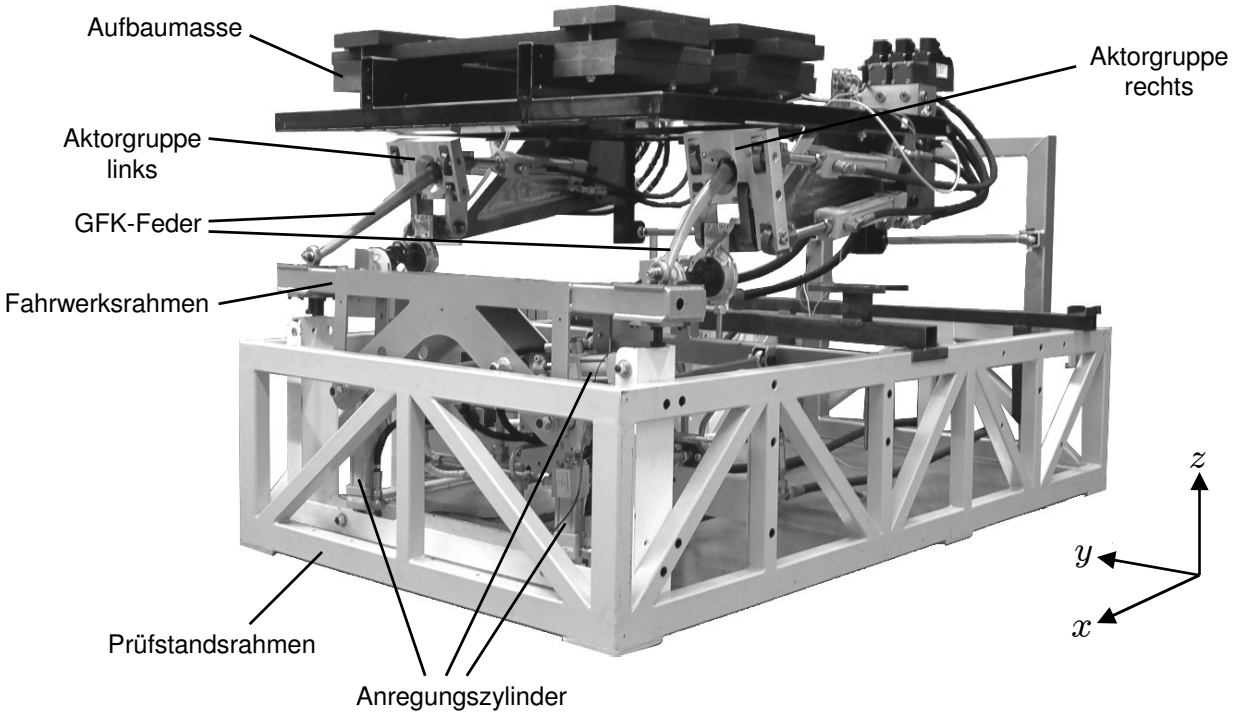


Abbildung 8.2: Halbfahrzeugprüfstand der Unterflur-Feder-Neige-Technik

die Querachse (Nicken) und die Bewegung in Längsrichtung sind eingeschränkt und werden nicht betrachtet. Die Feder-Neigetechnik besteht aus zwei spiegelsymmetrisch aufgebauten Aktormodulen, im Folgenden *Aktorgruppen* genannt. Diese Aktorgruppen binden jeweils drei horizontal angeordnete Hydraulikzylinder an, die über eine Umlenkkinematik einen Lenker in 3 Freiheitsgraden bewegen. Am Lenker ist eine GFK-Feder[1] montiert, über die der Aufbau an dem Fahrwerksrahmen abgestützt wird.

Die aktiven Eingriffe werden über eine Verstellung der beiden Umlenkkinematiken realisiert. Aus der Verstellung der Fußpunkte der GFK-Federn resultieren Federauslenkungen, die zu zusätzlichen Kräfte zwischen Aufbau und Fahrwerksrahmen führen.

Die Störanregung, die von den Gleisen auf das Fahrwerk wirkt, wird über drei Hydraulikzylinder simuliert. Diese ermöglichen eine Verstellung des Fahrwerkrahmens in vertikaler, in Quer- und in Wank-Richtung.

Der Prüfstand wurde von Schlautmann im Rahmen seiner Doktorarbeit entworfen und aufgebaut. Für detaillierte Informationen sei daher auf [Sch06] verwiesen. Die Erstellung eines Prüfstandmodells sowie der Entwurf und die Auslegung einer modularen Reglerstruktur ist in [Gei05] beschrieben.

[1] GFK: Glasfaser verstärkter Kunststoff

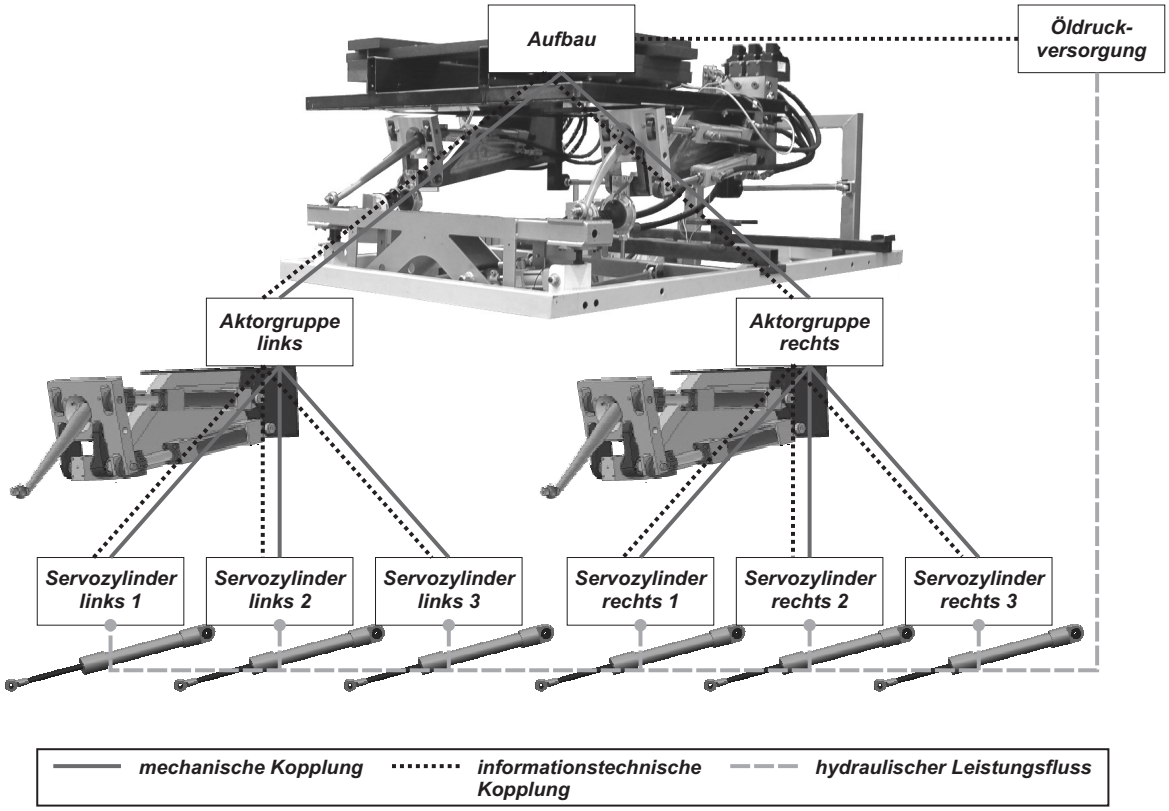


Abbildung 8.3: Hierarchischer Aufbau des Feder-Neige-Prüfstands

8.2 Hierarchisches Modell

Der Unterflur-Feder-Neige-Prüfstand ist wegen seines modularen Aufbaus ideal als Anwendungsbeispiel für den Aufbau eines hierarchischen Modells geeignet. Das Federungssystem lässt sich in insgesamt 10 Teilsysteme auf 3 Ebenen untergliedern. Die oberste Ebene besteht aus dem AMS *Aufbau* sowie dem Energieversorgungsmodul *Öldruckversorgung*. Die mittlere Ebene bilden zwei MFM, die beiden *Aktorgruppen*. Auf der untersten Ebene befinden sich insgesamt sechs gleichartige MFM, die *Servozylinder*. Diese Unterteilung ist in Abbildung 8.3 dargestellt.

Im Folgenden werden die nichtlinearen Basismodelle des Aufbaus, der Aktorgruppen und der Servozylinder beschrieben. Die Teilmodelle basieren auf dem in [Gei05] entwickelten Prüfstandsmodell. Wegen der Symmetrie der Aktorgruppen und wegen des identischen Aufbaus der Servozylinder wird nur eine Aktorgruppe und ein Servozylinder vorgestellt. Es wird insbesondere auf die Mechanik, die Aktorik und die Regler eingegangen.

8.2.1 Kopplung der mechanischen Teilmodelle

Die Teilmodelle des Unterflur-Feder-Neige-Prüfstand beschreiben unter anderem das Bewegungsverhalten mechanischer Mehrkörpersysteme (MKS). Eine Verkopplung im hierarchischen Modell erfordert eine modularisierte, mathematische Beschreibung des Mehrkörpersystems. Die Kopplung muss über einfache Eingangs- Ausgangsbeziehung abbildbar sein. Für die Kopplungen zweier mechanischer Teilsysteme müssen zwei Fälle unterschieden werden [Hah99]:

- **Dynamische Bindung:** Die dynamischen Bindung zweier Körper wird über Kräfte abgebildet. Dies ist der Fall, wenn die beiden Körper über Verbindungselemente gekoppelt sind, die eine Relativbewegung zulassen. Diese Art der Kopplung kann sowohl aktiver als auch passiver Natur sein. Bei aktiven Kopplungen werden über über Aktoren Kräfte auf das MKS eingeprägt. Beispiele für passive Kopplungen sind Feder- oder Dämpferelemente.

- **Kinematische Bindungen:** Eine kinematische Bindung liegt vor, wenn die beiden Teilsysteme über starre Verbindungen, wie z. B. Gelenke oder Schraubverbindungen, miteinander verbunden sind. Hieraus resultieren kinematische Zwangsbedinungen auf der Positionsebene[2].

Während die *dynamische Bindung* leicht über Kräfte abgebildet werden können, erfordert die *kinematische Bindung* spezielle Beschreibungsformen, die eine modulare Zerlegung des Systems erlauben. Einige Ansätze hierfür werden u. a. in [Bae86, Jun97, Hah99, Toe02, Koc05] beschrieben.

In diesem Anwendungsbeispiel wird die Kopplung mechanischer Teilsysteme über eine *Kraftkopplung* realisiert, welche dem Prinzip der *dynamischen Bindung* zuzuordnen ist.

Kraftkopplung

Bei einer *Kraftkopplung* wird das unterlagerte System als ein Aktor aufgefasst, welcher Kräfte und Momente auf das überlagerte Mehrkörpersystem aufprägt. Die Reaktion des Systems wird über Lage und Geschwindigkeit der Ankoppelpunkte auf die Eingänge des unterlagerten Modells geschaltet. Durch die Kraftkopplung ergibt sich eine klare und einfach zu modellierende Schnittstelle, die ohne zusätzliche Elemente zur Berücksichtigung der mechanischen Verkopplung auskommt.

Bei mechanischen Systemen ist in der Regel mindestens ein Ankoppelpunkt als starre Kopplung ausgelegt, z. B. über ein Gelenk oder einen Flansch. Die Kraftkopplung kann bei einfachen kinematischen Verhältnissen trotzdem anwendet werden. Dies führt jedoch zu einer Verletzung der Systemgrenzen. Körper, die eigentlich zum unterlagerten System gehören, die aber kinematisch über starre Gelenke an das überlagerte System angebunden sind, werden diesem zugeschlagen. Die Kinematik und Dynamik wird hierdurch nicht exakt nachgebildet und es kommt zu Abweichungen. Diese Vorgehensweise ist daher nur zulässig, wenn die Massen der unterlagerten Teilsysteme im Vergleich zum überlagerten System klein sind. Da die Masse der Aggregate im Allgemeinen mit höheren Strukturebenen zunimmt, können Abweichungen, für die ja die leichteren, unterlagerten Modelle verantwortlich sind, oft vernachlässigt werden. In Abbildung 8.4 ist diese Vorgehensweise am Beispiel eines Hydraulikzylinders dargestellt. Betroffen sind hier die Massen des Kolbens und des Zylinders, die dem Hydraulikzylinder zuzuordnen sind, aber starr mit den Körpern des überlagerten Systems verbunden sind. Die Massen und Trägheiten dieser Körper werden als zusätzliche Modellparameter an das überlagerte System übermittelt.

[2] Dies gilt im Falle eines holonomen mechanischen Systems. Bei einem nicht-holonomen System ergeben sich Zwangsbedinungen auf der Geschwindigkeitsebene [Jun97].

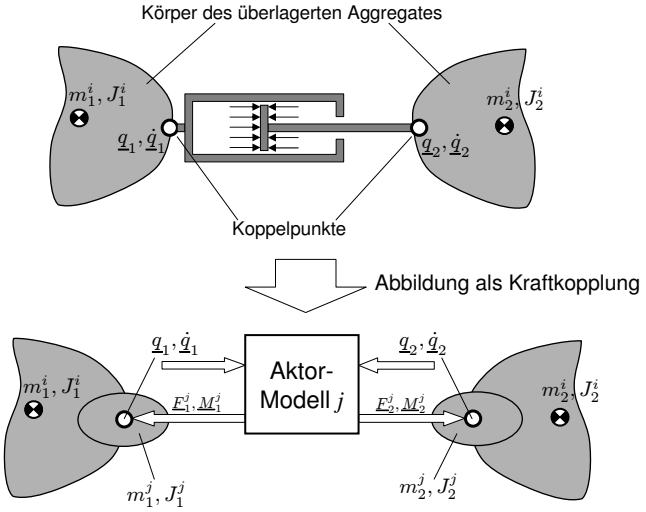


Abbildung 8.4: Anbindung eines Hydraulikzylinders an ein überlagertes System über eine Kraftkopplung

8.2.2 MFM: Servozylinder

Das Basismodell des *Servozylinders* beschreibt das Verhalten von drei Komponenten: einem 4/3-Wege-Hydraulikventil, einem Differentialzylinder und einem Positionsregler.

Der Servozylinder ist als eindimensionaler Kraftsteller modelliert. Die Massen der Kolbenstange und des Zylinders werden der überlagerten *Aktorgruppe* zugeschlagen. Abbildung 8.5 zeigt die Schnittstellen dieses Aktors.

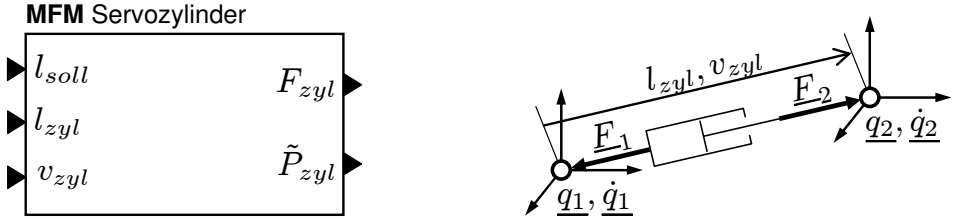


Abbildung 8.5: Schnittstellen und Einbindung des Servozylinders in ein MKS-System

Die Einbindung in ein dreidimensionales Mehrkörpersystem geschieht im überlagerten Modell. Die Umrechnung der Positionen $\underline{q}$ und Geschwindigkeiten $\underline{\dot{q}}$ auf die eindimensionalen Größen: Länge l_{zyl} und Geschwindigkeit v_{zyl} erfolgt dabei über die Beziehungen:

$$l_{zyl} = \left\| \underline{q}_2 - \underline{q}_1 \right\|_2 \tag{8.1}$$

$$v_{zyl} = \frac{\underline{q}_2 - \underline{q}_1}{l_{zyl}} \cdot \left(\underline{\dot{q}}_2 - \underline{\dot{q}}_1 \right) \tag{8.2}$$

und die Berechnung der Schnittstellenkräfte $\underline{F}_1$, $\underline{F}_2$ aus der Zylinderkraft F_{zyl} über:

$$\underline{F}_1 = -\frac{\underline{q}_2 - \underline{q}_1}{l_{zyl}} F_{zyl} \tag{8.3}$$

$$\underline{F}_2 = \frac{\underline{q}_2 - \underline{q}_1}{l_{zyl}} F_{zyl} \tag{8.4}$$

Differentialzylinder: Die Kraft an der Kolbenstange wird über:

$$F_{zyl} = p_A\, A_A - p_B\, A_B - F_d \tag{8.5}$$

bestimmt. Für die Reibkraft wird das einfache Reibmodell

$$F_d = d_{zyl}\, v_{zyl} \tag{8.6}$$

angesetzt.

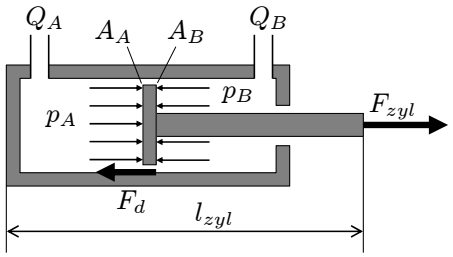


Abbildung 8.6: Modell des Differentialzylinders

Die Drücke in den Zylinderkammern p_A und p_B berechnen sich aus der Menge des zu- oder abströmenden Öls Q_A bzw. Q_B, dem Kompressionsmodul β_A bzw. β_B und dem Volumen V_A bzw. V_B. Die Auslenkungen des Zylinders sind bei diesem Federungssystem vergleichsweise gering. Die Volumina V_A und V_B können daher als konstant angenommen werden. Die Druckänderungen berechnen sich so zu:

$$\dot{p}_A = \frac{\beta_A}{V_A}\,(Q_A - A_A\, v_{zyl}) \tag{8.7}$$

$$\dot{p}_B = \frac{\beta_B}{V_B}\,(Q_B - A_B\, v_{zyl}) \tag{8.8}$$

Hydraulikventil: Das Hydraulikventil wird als ein 4/3-Wege-Ventil ohne Überdeckung der Steuerkanten abgebildet. Über Drosseleffekte werden die Durchflussmengen Q_A und Q_B zu den beiden Kammern des Zylinders reguliert. Über den Steuerschieber kann die Menge und Fließrichtung bestimmt werden (Abb. 8.7).

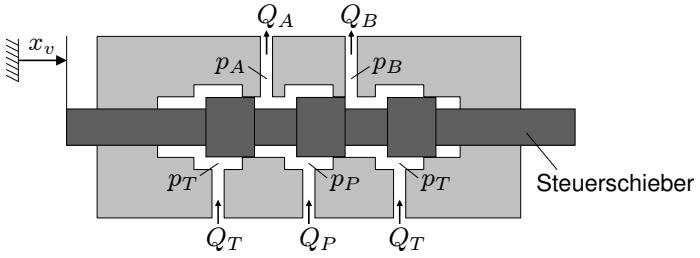


Abbildung 8.7: Modell des 4/3-Wege-Ventils

In Abhängigkeit zu der normierten Schieberstellung x_v und den Druckdifferenzen ergeben

sich die Volumenströme:

$$Q_A = \frac{Q_{Nenn}}{\sqrt{\Delta p_{Nenn}}}\, x_v \cdot \begin{cases} \sqrt{|p_P - p_A|} \cdot \mathrm{sgn}(p_P - p_A) & : \quad x_v > 0 \\ 0 & : \quad x_v = 0 \\ \sqrt{|p_A - p_T|} \cdot \mathrm{sgn}(p_A - p_T) & : \quad x_v < 0 \end{cases} \tag{8.9}$$

$$Q_B = \frac{Q_{Nenn}}{\sqrt{\Delta p_{Nenn}}}\, x_v \cdot \begin{cases} \sqrt{|p_T - p_B|} \cdot \mathrm{sgn}(p_T - p_B) & : \quad x_v > 0 \\ 0 & : \quad x_v = 0 \\ \sqrt{|p_B - p_P|} \cdot \mathrm{sgn}(p_B - p_P) & : \quad x_v < 0 \end{cases} \tag{8.10}$$

Der Steuerschieber wird über eine Feder und eine elektrische Spule aktuiert. Das dynamische Verhalten kann in der Regel über ein PT2-Glied hinreichend genau angegeben werden. Auf eine Änderung der Soll-Position $x_{v,soll}$ reagiert der Schieber mit der Übertragungsfunktion:

$$\frac{x_v(s)}{x_{v,soll}(s)} = \frac{1}{{T_v}^2\, s^2 + 2\, T_v\, \zeta_v\, s + 1} \tag{8.11}$$

mit der Zeitkonstante des Ventils T_v und der Dämpfung ζ_v. Die Positionen x_v und $x_{v,soll}$ sind dabei auf den Volumenstrom Q_{Nenn} beim Druck Δp_{Nenn} normiert.

Regler: Die Position des Hydraulikzylinders wird über einen PD-Regler eingeregelt. Mit der Regeldifferenz $\Delta l_{zyl} = l_{soll} - l_{zyl}$ wird der Reglerausgang $x_{v,soll}$ über:

$$\frac{x_{v,soll}(s)}{\Delta l_{zyl}(s)} = K_p + \frac{K_d\, s}{T_1\, s + 1} \tag{8.12}$$

berechnet. Die beiden Größen $K_{p,zyl}$ und $K_{d,zyl}$ stellen die einstellbaren Reglerparameter dar. Die Zeitkonstante des Differenzierers wird auf $T_1 = 0{,}001\,\mathrm{s}$ festgelegt.

Leistungsberechnung: Neben dem Bewegungsverhalten spielt die Leistungsaufnahme des Feder-Neige-Systems eine entscheidende Rolle. Die Leistung wird primär in den Aktoren auf der untersten Ebene eines Systems umgesetzt (vgl. Abbildung 8.3). Für die Berechnung ist hier die hydraulische Leistung an der Druckseite des Ventils relevant:

$$P_{zyl} = p_P \cdot Q_P \tag{8.13}$$

da diese Leistung von der Druckversorgung aufgebracht wird. Der Volumenstrom ergibt sich aus den Differenzdrücken und aus der Stellung des Ventilschiebers:

$$Q_P = \frac{Q_{Nenn}}{\sqrt{\Delta p_{Nenn}}}\, |x_v| \cdot \begin{cases} \sqrt{|p_P - p_A|} \cdot \mathrm{sgn}(p_P - p_A) & : \quad x_v > 0 \\ 0 & : \quad x_v = 0 \\ \sqrt{|p_P - p_B|} \cdot \mathrm{sgn}(p_P - p_B) & : \quad x_v < 0 \end{cases} \tag{8.14}$$

In Hinblick auf die Linearisierung und die Reduktion dieses Modells besteht hier die Schwierigkeit, dass der Volumenstrom nicht über ein LZI-Modell approximiert werden kann. Hierfür ist der folgende Effekt verantwortlich:

Der Volumenstrom Q_P bewegt sich bei normalen Betriebsbedingungen wegen der Umschaltung im Ventil sowohl bei $x_v < 0$ als auch $x_v > 0$ im positiven Bereich. Dieses Verhalten ist ähnlich dem einer Betragsfunktion.

Der Volumenstrom Q_P geht bei $x_v > 0$ zur Kammer A und bei $x_v < 0$ zur Kammer B. Interpretiert man den Volumenstrom zur Kammer B hin als negativen Volumenstrom, so kann Q_P im normalen Arbeitsbereich über:

$$\tilde{Q}_P = \max(Q_A, 0) - \max(Q_B, 0) \tag{8.15}$$

angenähert werden[3]. Die Approximation von $P_{zyl} \approx |\tilde{P}_{zyl}|$ erfolgt über:

$$\tilde{P}_{zyl} = p_P \cdot \tilde{Q}_P \tag{8.16}$$

Das Vorzeichen von $\tilde{P}_{zyl}$ überträgt die Information, welche Kammer mit der Druckseite verbunden ist.

8.2.3 MFM: Aktorgruppe

Das MFM *Aktorgruppe* besteht im Wesentlichen aus einem Aktorrahmen, einer GFK-Feder und einer Lenkerkinematik. Die Aktoren bilden drei unterlagerte MFM *Servozylinder*. Die Bewegung der Zylinder wird über die Lenkerkinematik in eine Federfußpunktverstellung umgelenkt. In Abbildung 8.8 ist ein CAD-Modell der Aktorgruppe mit den unterlagerten Hydraulikzylindern zu sehen.

Das Basismodell der Aktorgruppe beschreibt das dynamische Verhalten der Lenkerkinematik und den Krafteinfluss der GFK-Feder. Die Servozylinder stellen Kräfte zwischen den jeweiligen Koppelpunkten an der Lenkerplatte und am Aktorrahmen. Die Aktorgruppe ist als Kraftsteller zwischen dem Fahrwerksrahmen und dem Aufbau abgebildet. Die gesamte Masse der Aktorgruppe inklusive der Massen der Servozylinder muss hierzu dem Aufbau zugeschlagen werden.

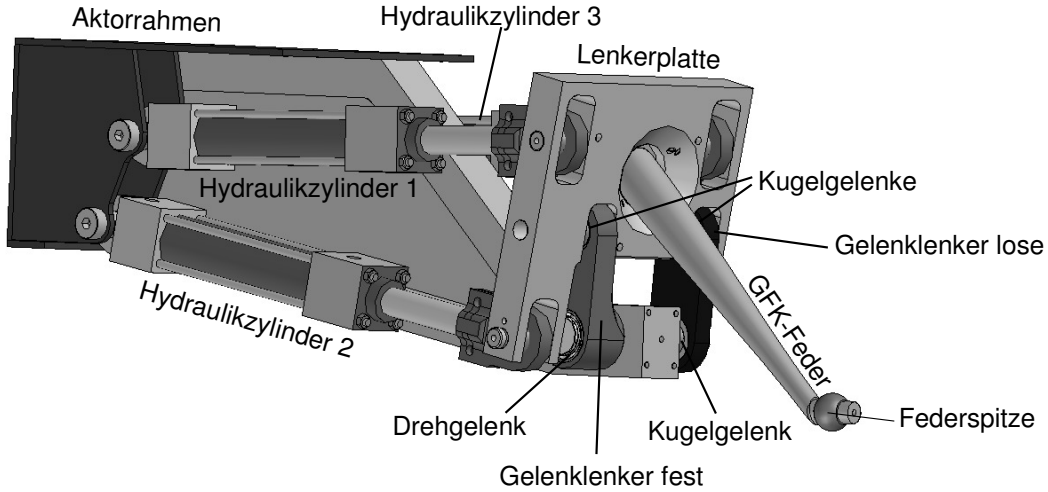


Abbildung 8.8: Aufbau der Aktorgruppe

[3] In [Löf08] wird ein alternativer Ansatz beschrieben. Der Volumenstrom wird hier über die Geschwindigkeit und den Querschnittsflächen des Kolbens abgeschätzt. Das asymmetrische Verhalten des Differentialzylinders findet hierdurch Berücksichtigung, demgegenüber wird jedoch die Kompressibilität des Öls vernachlässigt.

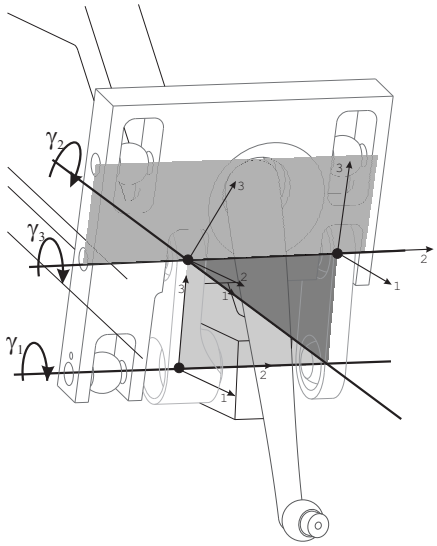


Abbildung 8.9: Lage der Achsen in der Lenkerkinematik

Lenkerkinematik: Die Lenkerkinematik besteht aus dem Lenker, der über einen „festen“ und einen „losen“ Gelenklenker mit dem Aktorrahmen verbunden ist. Diese Kinematik erlaubt eine Bewegung der Lenkerplatte in drei Freiheitsgraden, über die eine Verschiebung der Federspitze in x, y und z-Richtung möglich ist.

Die Kinematik weist eine parallele Struktur auf. Bei genauerer Betrachtung kann sie jedoch als eine offene kinematische Kette dargestellt werden. Das Ersatzsystem besteht aus drei Körpern, die über drei Gelenke verbunden sind. Die drei Achsen dieser Gelenke sind in Abbildung 8.9 eingezeichnet. Die Lenkerkinematik lässt sich so einfach als eine Kette von Gelenken und Starrkörpern mit Hilfe eines MKS-Tools abbilden.

Die Struktur des MKS-Modells ist in Abbildung 8.10 dargestellt. In das Modell gehen die Position und Orientierung des Aktorrahmens $\underline{q}_1$, die Position der Federspitze $\underline{q}_2$ und deren zeitliche Ableitungen $\dot{\underline{q}}_1$ und $\dot{\underline{q}}_2$.

GFK-Feder: Die GFK-Feder besteht aus unidirektionalem glasfaserverstärkten Kunststoff. Es handelt sich hier um einen dreidimensionalen Biegebalken mit anisotropen Werkstoffverhalten. Die Kraft-Weg-Beziehung ist analytisch nur schwer herzuleiten. In erster Näherung kann die Feder jedoch durch einen linearen Zusammenhang zwischen Auslenkung der Federspitze $\underline{q}_F$ und der Kraft $\underline{F}_F$ abgebildet werden:

$$\underline{F}_F = \underbrace{\begin{bmatrix} c_x & c_{xy} & c_{xz} \\ c_{xy} & c_y & c_{yz} \\ c_{xz} & c_{yz} & c_z \end{bmatrix}}_{\underline{\mathbf{C}}_F} \underline{q}_F + \underbrace{\begin{bmatrix} d_x & d_{xy} & d_{xz} \\ d_{xy} & d_y & d_{yz} \\ d_{xz} & d_{yz} & d_z \end{bmatrix}}_{\underline{\mathbf{D}}_F} \dot{\underline{q}}_F \tag{8.17}$$

Die Koppelfedersteifigkeiten c_{xy}, c_{xz} und c_{yz} resultieren aus der schiefen Einbaulage der GFK-Feder. Die in (8.17) verwendeten Größen werden im körperfesten Koordinatensystem der Lenkerplatte angegeben. Die Auslenkung $\underline{q}_F$ wird aus der gedachten Position der unausgelenkten und der tatsächlichen Position der Federspitze berechnet.

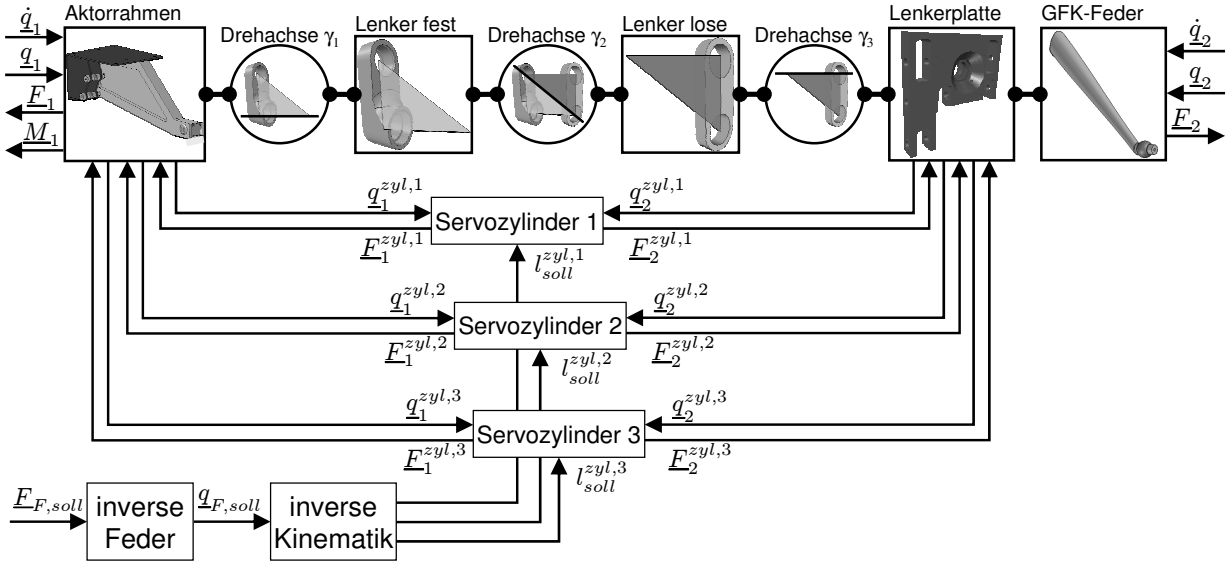


Abbildung 8.10: MKS-Modell der Aktorgruppe

Steuerung: Die Steuerungsaufgabe der Aktorgruppe besteht in der Fußpunktverstellung der GFK-Feder um zusätzliche Kräfte zwischen dem Aufbau und dem Fahrwerksrahmen zu stellen. Die Kraftvorgabe $\underline{F}_{F,soll}$ für die Aktorgruppe wird über die inverse Federsteifigkeit in die Federfußpunktverstellung $\underline{q}_{F,soll}$ umgerechnet:

$$\underline{q}_{F,soll} = \left(\underline{\mathbf{C}}_F\right)^{-1} \underline{F}_{F,soll} \tag{8.18}$$

Die Fußpunktverstellung beschreibt die relative Auslenkung der als unbelastet angenommenen Federspitze. Der Dämpfungsterm $\underline{\mathbf{D}}_F$ aus (8.17) wird hier vernachlässigt, da er im Grunde nur zu einer unnötigen Verlangsamung des Regelungsverhaltens führt.

Die Berechnung der Sollängen der drei Servozylinder $l_{soll}^{zyl,j}$ erfolgt über die Gleichungen der inversen Kinematik (Abb. 8.10). Diese Sollängen werden den unterlagerten Zylinderreglern als Sollwerte vorgegeben. Hierbei wird davon ausgegangen, dass die gewünschten Längen von den Servozylindern mit einer hinreichenden Regelgüte eingestellt werden. Die tatsächlichen Längen werden nicht zurückgeführt, es findet in der Aktorgruppe also keine Regelung sondern eine Steuerung statt.

8.2.4 AMS: Aufbau

Das Modell des AMS *Aufbau* beschreibt im Wesentlichen die Aufbaumasse, den Fahrwerksrahmen, Sensoren und den Aufbauregler. Die Bewegung der Aufbaumasse wird über zwei unterlagerten Aktorgruppen beeinflusst. Das AMS Aufbau ist ein autonomes System, die Regler sind daher in der Lage ohne Vorgaben von außen die Federungs- und Dämpfungseigenschaften einzustellen.

Aufbaumasse des Prüfstandes wird über drei Stützstangen in seiner Bewegung geführt. Diese Stützstangen realisieren eine Parallelführung und erlauben Bewegungen in der y-z-Ebene sowie Verdrehungen um die Längsachse φ. Die Bewegung in x-Richtung sind sehr gering und werden im Modell vernachlässigt.

Zur Abbildung der Anregungen, die über das Fahrwerk auf das Federungssystem wirken, kann der Fahrwerkrahmen in der y-z-Ebene bewegt und um φ verdreht werden.

Sensoren: Die Bewegung der Aufbaumasse wird mit zwei Sensorarten erfasst.

Die Absolutbeschleunigung wird mit Hilfe von vier Beschleunigungssensoren, die an verschiedenen Stellen des Aufbaus angebracht sind, gemessen. Aus diesen Signalen werden die Beschleunigungen in Hub-, Wank- und Querrichtung ermittelt und in dem Vektor $\underline{\ddot{x}}_a$ zusammengefasst.

Der relative Abstand zwischen dem Aufbau und dem Fahrwerk wird bei jeder Aktorgruppe über einen berührungslosen induktiven 3D-Sensor erfasst. Der Vektor der Relativposition $\underline{x}_{rel}$ enthält den Abstand in z, y und φ-Richtung.

Aufbauregler: Der Aufbauregler hat verschiedene Aufgaben zu erfüllen: Einstellen des Grundniveaus, Neigen des Aufbaus bei Kurvenfahrten und die aktive Federung und Dämpfung zur Erhöhung des Komforts. In dieser Arbeit wird nur auf den Komfortregler eingegangen, da die anderen Aspekte für die Optimierung eine untergeordnete Rolle spielen.

Der Komfortregler arbeitet nach dem Prinzip einer *Sky-Hook*-Dämpfung. Begriff der Sky-Hook-Dämpfung wurde von Karnopp geprägt [Kar78]. Er beschreibt anschaulich die Funktionsweise, die inertiale Aufbaugeschwindigkeit als Regelgröße zu verwenden (siehe auch [Str96, LG99, Hes06]).

Das Kraftgesetz für eine Bewegungsrichtung i lautet:

$$F_{a,i} = c_{rel,i}\, x_{rel,i} + d_{sky,i}\, \dot{x}_{a,i} \tag{8.19}$$

Da die inertiale Aufbaugeschwindigkeit nicht direkt gemessen werden kann, muss sie über die Beschleunigung durch Integration ermittelt werden. Um eine Drift des Integrators zu vermeiden wird das Beschleunigungssignal $\ddot{x}_{a,i}$ über einen Hochpassfilter um quasistationäre Anteile reduziert. Durch Zusammenfassen von Integrator und Hochpassfilter gelangt man zu dem Kraftgesetz:

$$F_{a,i}(s) = c_{rel,i}\, x_{rel,i}(s) + \frac{T_{HP}}{T_{HP}\, s + 1}\, d_{sky,i}\, \ddot{x}_{a,i}(s) \tag{8.20}$$

Die Reglerstruktur ist an das AKTAKON-Prinzip angelehnt [ADG89], welches sich in verschiedenen Anwendungen bei Kraftfahrzeugen bewährt hat [Rut98, Hes06, Sch09]. Die Bewegungsrichtungen werden als entkoppelt angesehen und das Kraftgesetz (8.20) separat für jede Bewegungsrichtung betrachtet. Die Hubbewegung ist dabei vollständig entkoppelt von den anderen Bewegungsrichtungen. Die Querbewegung und das Wanken weisen jedoch einen gewissen Grad der Verkopplung auf. Dennoch können mit einer Regelung in diesen Koordinaten gute Ergebnisse erzielt werden [Gei05]. Auf eine Modaltransformation wurde daher verzichtet. Abbildung 8.11 zeigt die Struktur der Regelung.

Über die Entwurfsparameter $\underline{c}_{rel}$ und $\underline{d}_{sky}$ können die Federsteifigkeit und Dämpfungsrate des Fahrwerks angepasst werden.

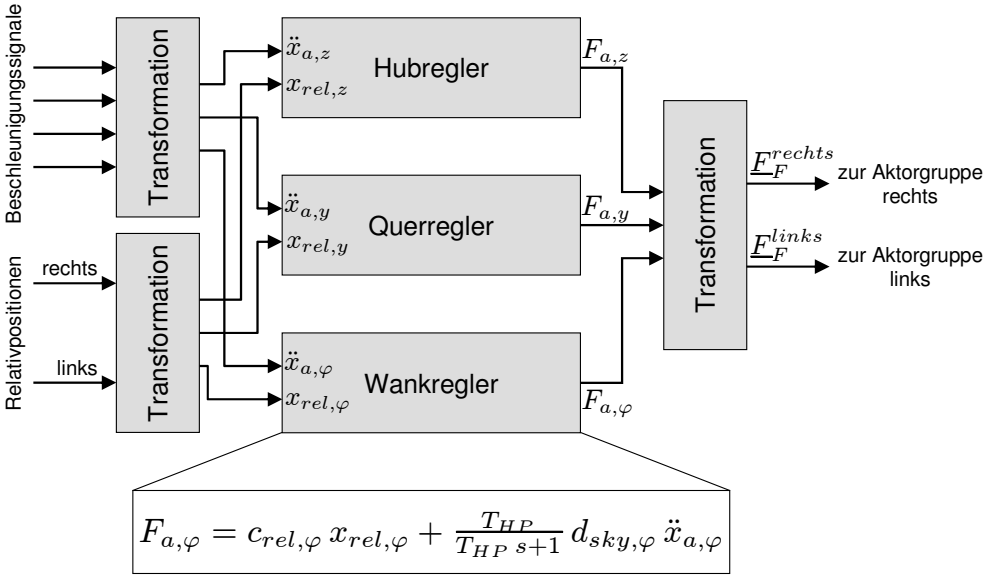


Abbildung 8.11: Struktur des Aufbaureglers

8.2.5 Öldruckversorgung

Die Hydraulik des Federungssystems und der Anregung wird über eine druckgeregelte Pumpe mit einem Druck von $p_P = 120\,\text{bar}$ versorgt. Auf der Tankseite herrscht Umgebungsdruck $p_T = 1\,\text{bar}$. Druckspeicher direkt von den Ansteuerventilen auf Tank- und Druckseite sorgen für einen möglichst gleichmäßigen Druck. Die Öldruckversorgung wird vereinfacht als Konstantdruckquelle angenommen und ihre Dynamik vernachlässigt. Schläuche und Rohrleitungen werden ebenfalls nicht berücksichtigt.

8.3 Reduktion der Basismodelle

Für den Modellaustausch im hierarchischen Modell werden die nichtlinearen Basismodelle linearisiert und in ihrer Ordnung reduziert. In Hinblick auf eine spätere Optimierung müssen insbesondere zwei Aspekte in den vereinfachten Modellen erhalten bleiben:

- Schwingungsverhalten
- Leistungsbedarf

Das Schwingungsverhalten bildet den Hauptaspekt, da sich hierüber die regelungstechnische Stabilität des Systems, das Zusammenwirken der verschiedenen Ebenen und die Regelgüte bzw. der Komfort ableiten lassen. Der Leistungsbedarf eines Teilsystems ist für die Bewertung des Aufwandes notwendig.

8.3.1 Modellreduktion der Servozylinder

Da die Massenträgheiten in dem Modell keine Berücksichtigung finden, beschreibt das Modell der Servozylinder im Wesentlichen den Positionsregler und das Verhalten der hydraulischen Komponenten.

Die folgenden Größen bilden die Schnittstellen und Zustände des Systems:

Parameter: $\underline{p}_{zyl} = \left[K_p, K_d\right]$

Eingangsgrößen: $\underline{u}_{zyl} = \left[l_{soll}, l_{zyl}, v_{zyl}\right]$

Ausgangsgrößen: $\underline{y}_{zyl} = \left[F_{zyl}, \tilde{P}_{zyl}\right]$

Zustandsgrößen: $\underline{x}_{zyl} = \left[x_v, \dot{x}_v, p_A, p_B, x_{T_1}\right]$

Tabelle 8.1 zeigt die Eigenwerte des Modells für die Parameter $K_p = 2000\,\frac{1}{\mathrm{m}}$ und $K_d = 0{,}2\,\frac{\mathrm{s}}{\mathrm{m}}$. Das Modell hat die Ordnung 5. Das konjungiert-komplexe Eigenwertpaar kann dem Steuerschieber zugeordnet werden. Die beiden Kammerdrücke p_A und p_B besitzen ein integrierendes Verhalten. Dies wird anhand der beiden Null-Spalten in der Systemmatrix $\underline{\mathbf{A}}_{zyl}$ ersichtlich, ebenso spiegelt sich dies in den beiden Null-Eigenwerten wieder. Im nichtlinearen Modell sind die beiden Kammerdrücke notwendig, um die Schub- und Zugkräfte des Zylinders zu realisieren, da aus physikalischen Gründen in den Zylinderkammern kein negativer Druck herrschen kann. Diese Zustände können im linearisierten Modell anschaulich als Differenzdrücke der beiden Kammern interpretiert werden, die hier sehr wohl auch negative Werte annehmen.

In der Ordnungsreduktion ist zu erwarten, dass die beiden redundanten Zustände p_A und p_B zu einem Zustand, also einem Differenzdruck, zusammengefasst werden. Darüber hinaus existiert bei dem Servozylinder kein Reduktionspotential. Sowohl die Dynamik des Steuerschiebers als auch der Differenzierer im Positionsregler sind auf den nächsthöheren Ebene von Bedeutung und können nicht entfernt werden.

Das reduzierte Modell lässt sich durch eine Minimalrealisierung erzeugen. Eine Minimalrealisierung bedeutet, dass alle nicht-beobachtbaren und/oder nicht-steuerbaren Zustände aus dem Modell entfernt werden.

Tabelle 8.1: Eigenwerte des Servozylindermodells

$Re\{\lambda\}$ in s^{-1}	$Im\{\lambda\}$ in s^{-1}	Frequenz in Hz
-1000,0	0,0	159,2
-263,9	269,2	60,0
-263,9	-269,2	60,0
0,0	0,0	0,0
0,0	0,0	0,0

Tabelle 8.2: Eigenwerte des minimal realisierten Servozylindermodells

$Re\{\lambda\}$ in s^{-1}	$Im\{\lambda\}$ in s^{-1}	Frequenz in Hz
-1000,0	0,0	159,2
-263,9	269,2	60,0
-263,9	-269,2	60,0
0,0	0,0	0,0

8.3.2 Modellreduktion der Aktorgruppen

Da die beiden Aktorgruppen spiegelsymmetrisch aufgebaut sind, wird im Folgenden nur die rechte Aktorgruppe betrachtet. Reduktion der Aktorgruppen verläuft über einen zweistufigen Ansatz, beim dem sequentiell zwei unterschiedliche Ordnungsreduktionsverfahren auf das Modell angewendet werden. Im Vorfeld wurden Kombinationen verschiedener Modellreduktionsverfahren (siehe A.1) getestet. Über den gesamten Parameterraum wurde mit dem folgenden Reduktionsansatz sehr gute Ergebnisse erzielt:

1. Balancierung des Modells und Abschneiden mit Hilfe der singulären Pertubation auf Ordnung 13

2. Modaltransformation und Abschneiden hoher Eigenwerte mit Hilfe der singulären Pertubation auf Ordnung 12

Das linearisierte Basismodell besitzt 9 Eingänge, 6 Ausgänge und 18 Zustände. Die Aktorgruppen besitzen keine Parameter, die im Zuge einer Optimierung justiert werden können. Die relevanten Größen des Modells sind:

Eingangsgrößen: $\underline{u}_{ag} = \left[\underline{F}_{F,soll}, \Delta\underline{q}_2, \Delta\underline{\dot{q}}_2\right]$

Ausgangsgrößen: $\underline{y}_{ag} = \left[\underline{F}_F, \underline{\tilde{P}}_{zyl}\right]$

Zustände: $\underline{x}_{ag} = \left[\underline{x}^1_{zyl}, \underline{x}^2_{zyl}, \underline{x}^3_{zyl}, \underline{\hat{x}}_{Lenker}\right]$

Die Ausgänge bilden neben der Federkraft $\underline{F}_F$ auch die 3 Leistungen der Servozylinder $\underline{\tilde{P}}_{zyl}$. Dieser Leistungsvektor kann im Zuge einer linearen Ordnungsreduktion nicht weiter zusammengefasst werden und muss bis zum AMS Aufbau mitgeführt werden.

Die folgenden Ergebnisse wurden mit identischen Einstellungen drei Servozylinder berechnet. Im ersten Reduktionsschritt wird das Modell durch die Kombination des balancierten Abschneidens mit der singulären Pertubation in der Ordnung reduziert. In den Tabellen 8.3 und 8.4 sind die Eigenwerte des ursprünglichen und des reduzierten Basismodells gegenübergestellt. Das Systemverhalten ist stabil, die Eigenwertlagen bleiben bei dieser Methode jedoch nicht erhalten. Das Reduktionsergebnis ist wegen des hohen Eigenwertes von $\lambda_1 = -12330\,\mathrm{s}^{-1}$ noch nicht zufriedenstellend, da die Simulation dieses Modells eine kleine Simulationsschrittweite erfordert, was zu einer langen Simulationszeit führt. Dieser Eigenwert ist um ca. Faktor 4,5 schneller als der nächst schnellste Eigenwert (siehe Abb. 8.12).

Da der hohe Eigenwert im ersten Reduktionsschritt nicht entfernt wurde, ist davon auszugehen, dass die Berechnungspfade über diesen Zustand einen wesentlichen Einfluss auf das Ergebnis ausüben. Der große Abstand zum nächst niedrigeren Eigenwert signalisiert jedoch, dass das Einschwingverhalten bezogen auf die übrigen Zustände als „unendlich“ schnell angenommen und über den Endwert in der Durchgriffsmatrix berücksichtigt werden kann.

Das reduzierte Modell wird in dem zweiten Reduktionsschritt auf Modalform transformiert. Alle Eigenwerte mit $Re\{\lambda_i\} < -5000\,\mathrm{s}^{-1}$ werden anschließend mittels singulärer Pertubation entfernt. Hieraus resultiert ein Zustandsraummodell der Ordnung 12 (siehe A.3).

Tabelle 8.3: Eigenwerte der rechten Aktorgruppe

$Re\{\lambda\}$ in s^{-1}	$Im\{\lambda\}$ in s^{-1}	Frequenz in Hz
– 14460,0	0,0	2301,0
–2746,0	0,0	437,0
–1253,0	2861,0	497,1
–1253,0	–2861,0	497,1
–1583,0	2107,0	419,4
–1583,0	–2107,0	419,4
–241,5	247,0	54,98
–241,5	–247,0	54,98
–242,3	244,3	54,76
–242,3	–244,3	54,76
–241,9	244,6	54,75
–241,9	–244,6	54,75
–44,56	0,0	7,092
–44,24	0,0	7,041
–44,19	0,0	7,033
0,0	0,0	0,0
0,0	0,0	0,0
0,0	0,0	0,0

Tabelle 8.4: Eigenwerte nach dem 1. Reduktionsschritt

$Re\{\lambda\}$ in s^{-1}	$Im\{\lambda\}$ in s^{-1}	Frequenz in Hz
– 12330,0	0,0	1962,0
–2760,0	0,0	439,3
–1630,0	1945,0	403,9
–1630,0	–1945,0	403,9
–1494,0	0,0	237,8
–206,6	224,0	48,50
–206,6	–224,0	48,50
–207,4	220,6	48,19
–207,4	–220,6	48,19
–174,9	125,9	34,30
–174,9	–125,9	34,30
–115,3	0,0	18,35
–113,9	0,0	18,13

Abbildung 8.13 zeigt einen Vergleich zwischen dem Basismodell, dem reduzierten Basismodell und einem monolithischen nichtlinearen Modell. Für die Simulation wurde der Aktorrahmen und die Federspitze starr an die Umgebung angebunden und derart verspannt, dass eine für den Betrieb des Federungssystems typische Grundlast vorherrscht. Angetragen sind die Federkräfte in z bzw. in y-Richtung und die kumulierten Leistungen der Hydraulikzylinder $\sum_{i=1}^{3} |\tilde{P}^{zyl,i}|$ bei einem Sprung von $F_{F,soll,z}$ und einem Sprung von $F_{F,soll,y}$. Die Sprungantworten des Basismodells und des reduzierten Modells sind nahezu deckungsgleich. Die Abweichungen der beiden Modelle zum monolithischen Modell, ist auf die Linearisierung der Servozylinder zurückzuführen.

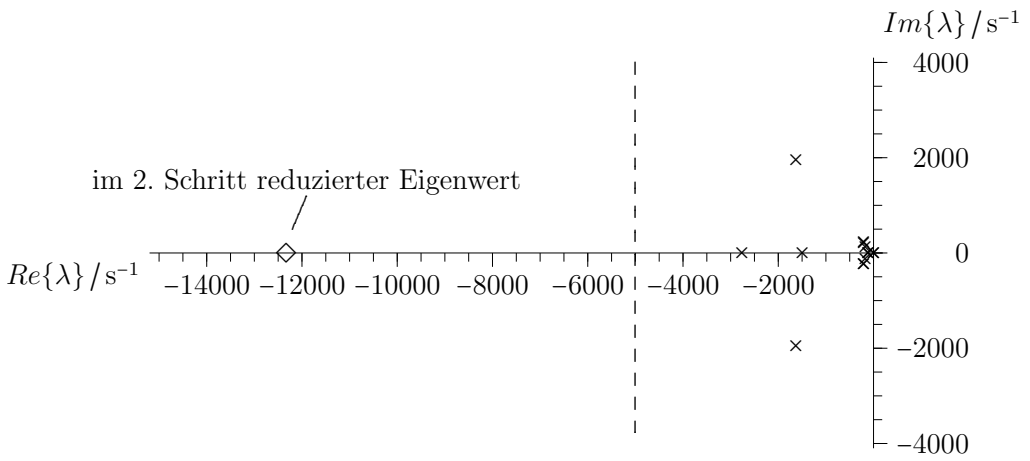


Abbildung 8.12: Eigenwerte in der komplexen Ebene nach dem 1. Reduktionsschritt

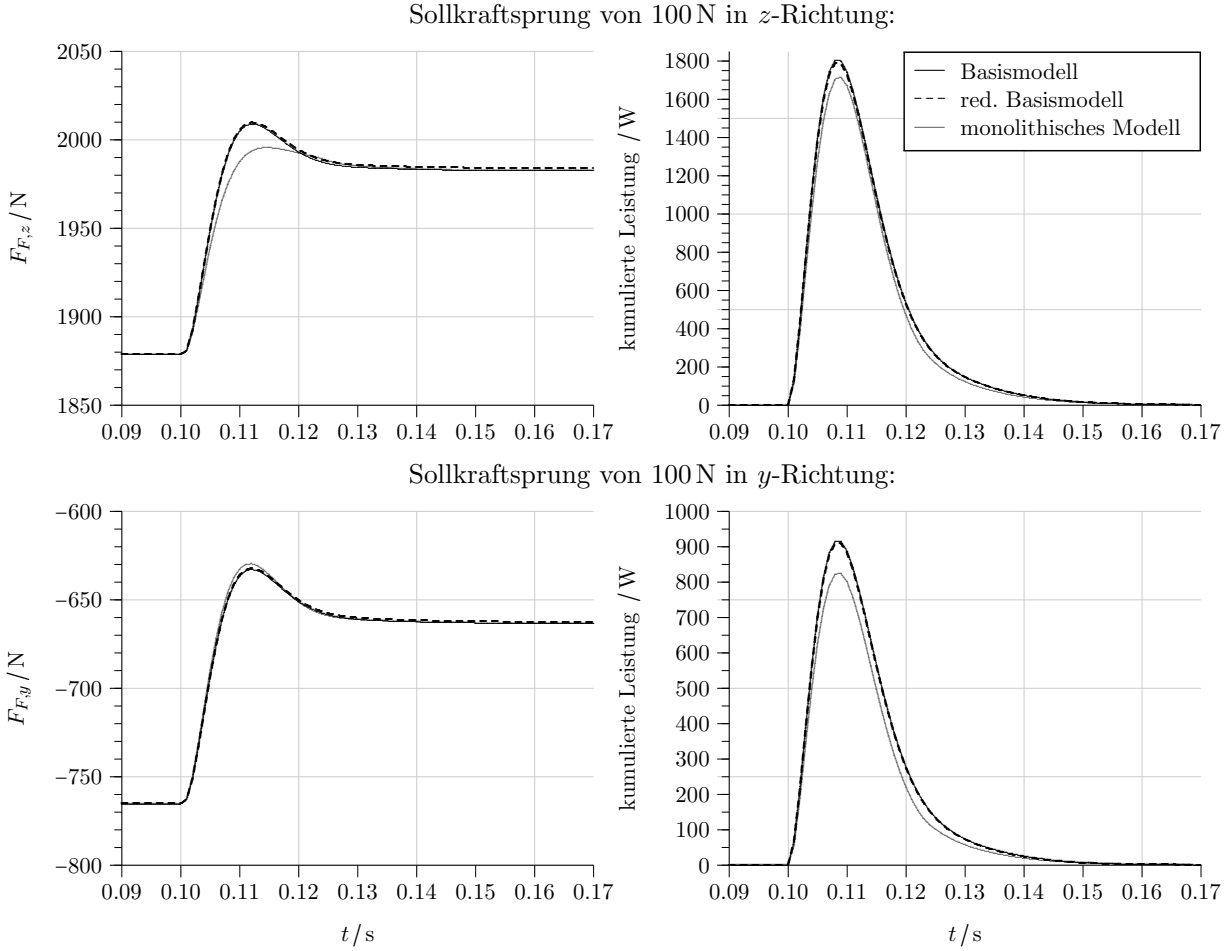


Abbildung 8.13: Vergleich des reduzierten Modell mit dem Basismodell der Aktorgruppe und einem monolithischen nichtlinearen Modell für einem Sprung in z und y-Richtung

8.3.3 Modellreduktion des Aufbaus

Im Grunde ist die Modellreduktion des Aufbaumodells für die hierarchische Optimierung nicht notwendig. Das Basismodell ist mit den reduzierten Modellen der Aktorgruppen vollständig ausgeprägt und kann für die Berechnung der Zielgrößen eingesetzt werden. Dennoch ist eine Reduktion sinnvoll, um anderen Prozessen im kognitiven Operator oder den überlagerten Systemen ein vereinfachtes Verhaltensmodell des Federungssystems zur Verfügung zu stellen, welches eine schnelle Simulation unter geringem Rechenaufwand ermöglicht. Ein Planungsverfahren beispielsweise kann mit Hilfe eines solche Modells verschiedenen Entscheidungsmöglichkeiten mit vertretbaren Rechenaufwand überprüfen.

Das linearisierte Modell des Aufbaus besitzt 3 Eingänge, 12 Ausgänge, 58 Zustände und 6 Parameter. Die folgenden Größen bilden die Schnittstelle des Modells:

Parameter: $\underline{p}_a = \left[\underline{c}_{rel}, \underline{d}_{sky}\right]$

Eingangsgrößen: $\underline{u}_a = \left[x_{anr,z}, x_{anr,y}, x_{anr,\varphi}\right]$

Ausgangsgrößen: $\underline{y}_a = \left[\ddot{x}_{a,z}, \ddot{x}_{a,y}, \ddot{x}_{a,\varphi}, x_{a,z}, x_{a,y}, x_{a,\varphi}, \underline{\tilde{P}}_{zyl}^{rechts}, \underline{\tilde{P}}_{zyl}^{links}\right]$

Die Größen $x_{anr,z}, x_{anr,y}, x_{anr,\varphi}$ bezeichnen die Störanregungen die über den Fahrwerkrahmen übertragen wird. Analog zu den Aktorgruppen wird das Modell in zwei Reduktionsschritten vereinfacht:

1. Balanciertes Abschneiden auf Ordnung 28
2. Modaltransformation und entfernen der Eigenwerte mit $Re\{\lambda_i\} < -500\,\mathrm{s}^{-1}$ mittels singulärer Pertubation auf Ordnung 24

Die folgenden Ergebnisse wurden mit den Reglerparametern des Aufbaus:

$$c_{rel,z} = -4000\,\tfrac{\mathrm{N}}{\mathrm{m}}\,,\quad c_{rel,y} = -4000\,\tfrac{\mathrm{N}}{\mathrm{m}}\,,\quad c_{rel,\varphi} = -2000\,\mathrm{N}$$

und

$$d_{sky,z} = \quad 1000\,\tfrac{\mathrm{Ns}}{\mathrm{m}},\ d_{sky,y} = \quad 1000\,\tfrac{\mathrm{Ns}}{\mathrm{m}},\ d_{sky,\varphi} = \quad 1000\,\mathrm{Ns}$$

erstellt. In den Tabellen 8.5 und 8.6 sind die Eigenwerte des Basismodells und der reduzierten Basismodells eingetragen. Im ersten Reduktionsschritt verbleiben insgesamt 4 Eigenwerte mit Realteilen $Re\{\lambda_i\} < -500\,\mathrm{s}^{-1}$ im Modell, die im 2. Schritt entfernt werden.

Wie in Abbildung 8.14 zu sehen, wird das Bewegungsverhalten vom hierarchischen Basismodell und dem reduzierten Modell des Aufbaus sehr gut wiedergegeben. Bei Anregungen in z und in φ-Richtung werden vergleichbare Ergebnisse erzielt. Die Kurven für einen Anregungssprung im Fahrwerk von 5 mm in y-Richtung sind nahezu deckungsgleich mit den Kurven aus dem nichtlinearen monolithischen Modell. Der kumulierte Leistungsbedarf $\sum_{i=1}^{3}|\tilde{P}_{zyl,i}^{rechts}| + \sum_{i=1}^{3}|\tilde{P}_{zyl,i}^{links}|$ wird im hierarchischen Modellierungsansatz etwas höher eingeschätzt, grundsätzlich stimmen die Verläufe aber ebenfalls überein.

Tabelle 8.5: Eigenwerte des Aufbaumodells

$Re\{\lambda\}$ in s^{-1}	$Im\{\lambda\}$ in s^{-1}	Frequenz in Hz
−2758,0	0,0	438,9
−2739,0	0,0	435,9
−1629,0	1944,0	403,7
−1629,0	−1944,0	403,7
−1645,0	1931,0	403,7
−1645,0	−1931,0	403,7
−1541,0	0,0	245,3
−1493,0	0,0	237,6
−500,5	0,0	79,66
−497,5	0,0	79,18
−497,2	0,0	79,13
−458,5	0,0	72,97
−208,6	220,5	48,31
−208,6	−220,5	48,31
−204,6	220,1	47,83
−204,6	−220,1	47,83
−206,0	217,6	47,69
−206,0	−217,6	47,69
−183,0	164,2	39,13
−183,0	−164,2	39,13
−170,2	135,9	34,66
−170,2	−135,9	34,66
−174,2	126,5	34,26
−174,2	−126,5	34,26
−188,5	0,0	30,00
−188,5	0,0	30,00
−188,5	0,0	30,00
−185,2	0,0	29,48
−118,6	0,0	18,88
−116,5	0,0	18,54
−115,3	0,0	18,35
−100,0	0,0	15,92
−100,0	0,0	15,92
−100,0	0,0	15,92
−100,0	0,0	15,92
−100,0	0,0	15,92
−100,0	0,0	15,92
−22,07	0,0	3,513
−3,434	17,50	2,838
−3,434	−17,50	2,838
−3,969	15,07	2,480
−3,969	−15,07	2,480
−10,40	0,0	1,655
−0,7671	1,131	0,2175
−0,7671	−1,131	0,2175
−0,8955	0,8957	0,2016
−0,8955	−0,8957	0,2016
−0,8942	0,8872	0,2005
−0,8942	−0,8872	0,2005
0,0	0,0	0,0
0,0	0,0	0,0

Tabelle 8.6: Eigenwerte des reduzierten Modells

$Re\{\lambda\}$ in s^{-1}	$Im\{\lambda\}$ in s^{-1}	Frequenz in Hz
−2868,0	0,0	456,5
−1600,0	1915,0	397,2
−1600,0	−1915,0	397,2
−2044,0	0,0	325,3
−272,0	84,79	45,34
−272,0	−84,79	45,34
−268,0	0,0	42,65
−158,7	178,1	37,97
−158,7	−178,1	37,97
−106,5	147,3	28,93
−106,5	−147,3	28,93
−131,7	34,24	21,66
−131,7	−34,24	21,66
−97,75	14,78	15,73
−97,75	−14,78	15,73
−90,20	20,86	14,73
−90,20	−20,86	14,73
−81,30	0,0	12,94
−17,11	7,261	2,958
−17,11	−7,261	2,958
−3,431	17,51	2,840
−3,431	−17,51	2,840
−3,346	17,51	2,837
−3,346	−17,51	2,837
−0,8617	1,020	0,2125
−0,8617	−1,020	0,2125
−1,043	0,0	0,1660
0,0	0,0	0,0

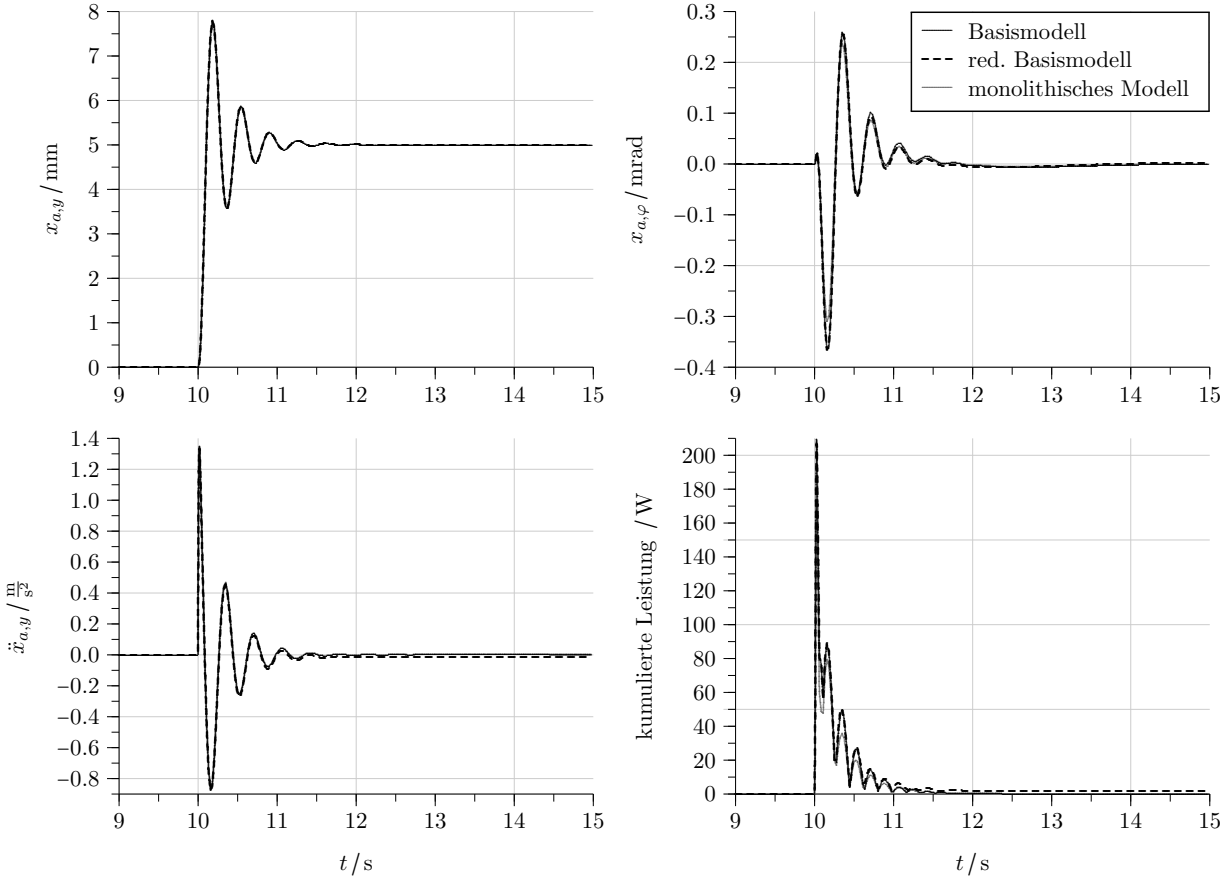


Abbildung 8.14: Vergleich des reduzierten Modell mit dem Basismodell des Aufbaus und einem monolithischen nichtlinearen Modell für einem Sprung des Fahrwerks in y-Richtung

8.3.4 Diskussion der Ergebnisse

In den oben vorgestellten Ergebnisse wird eine vergleichsweise moderate Reduktion des Modells durchgeführt. Es können durchaus noch niedrigere Ordnungen erreicht werden, die eine ähnlich gute Abbildung des Systemverhaltens liefern. In den Vorarbeiten [Löf08, Hua08] wurde bspw. mit modalen Ansätzen eine Reduktion der Aktorgruppe auf Ordnung 9 erreicht. Krüger zeigt in [KST10], dass mit modernen Krylov-Unterraummethoden (siehe z. B. [LS04]) sogar eine Reduktion auf Ordnung 6 bei guter Approximationsgüte möglich ist.

Die Verringerung der Modellordnung ist jedoch nur ein Aspekt. Mit niedrigen Ordnungen kann verstärkt das Problem auftreten, dass bei Variation der Parameter unterschiedliche Zustände der balancierten oder der modalen Darstellung entfernt werden. Bei kontinuierlicher Verschiebung der Parameter treten dann sprungartigen Änderungen im Modellverhalten auf, die eine Optimierung mit den in Kapitel 3 beschriebenen lokalen Methoden erschweren. Ein weniger stark reduziertes Modell verhält sich hier weniger problematisch, da die Sprünge mit der Ordnung geringer werden. Ein kontinuierlicher Verlauf kann bspw. mit parameterabhängigen Reduktionsverfahren erzeugt werden, wie sie von Krüger in [KST10] vorgeschlagen werden.

Für eine nachfolgende Optimierung ist neben der Approximationsgüte die Simulationsgeschwindigkeit von größter Bedeutung. Das Abschneiden hoher Eigenwerte hat hierauf

Tabelle 8.7: Beschleunigung der Simulation der verschiedenen Modelle für das Euler-Integrationsverfahren

Modell	Ordnung	max. Eigenwert in s^{-1}	max. Schrittweite in s	Speedup
Aufbau: reduziert	24	$-272{,}0 \pm 84{,}8\,i$	$5{,}5 \cdot 10^{-3}$	13,1
Aufbau: Basismodell	58	−2758,0	$5{,}0 \cdot 10^{-4}$	1,0
Aufbau: monolithisch	76	−14774,4	$1{,}35 \cdot 10^{-4}$	0,0172
Aktorgruppe: reduziert	12	−2760,0	$5{,}0 \cdot 10^{-4}$	22,2
Aktorgruppe: Basismodell	18	−14460,0	$1{,}38 \cdot 10^{-4}$	1,0
Aktorgruppe: monolithisch	21	−14440,3	$1{,}38 \cdot 10^{-4}$	0,940
Servozylinder: reduziert	4	−1000,0	$2{,}0 \cdot 10^{-3}$	1,15
Servozylinder: Basismodell	5	−1000,0	$2{,}0 \cdot 10^{-3}$	1,0

einen wesentlich größeren Einfluss als die Verringerung der Ordnung. So kann das reduzierte Basismodell der Aktorgruppe aufgrund der Simulationsschrittweite um ca. Faktor 4-5 schneller ausgewertet werden als das linearisierte Basismodell.

Tabelle 8.7 zeigt den Geschwindigkeitszuwachs der Simulation für die verschiedenen Modelle. Die monolithischen Modelle beschreiben das System auf allen Ebenen über nichtlineare Differentialgleichung, während die Basismodelle nur die nichtlinearen Effekte der jeweiligen Ebene abbilden. Die reduzierten Modelle werden prinzipbedingt über lineare Zustandraumdarstellungen beschrieben.

Die maximale Schrittweite bezeichnet die Integrationsschrittweite, bei der das Modell stabil simuliert werden kann. Sie ist stellvertretend für das Euler-Integrationsverfahren angegeben. Der Beschleunigungsfaktor (*Speedup*-Faktor) wurde empirisch anhand von Simulationen mit dieser Schrittweite ermittelt.

Die Simulationen wurden mit dem MKS-Simulations-Tool CAMeL-View 6.5 [Hah99, iXt01b] auf einem PC mit einem Intel Core 2 Duo T8300 Prozessor (2,4 GHz) durchgeführt. Auf diesem Rechner konnte das nichtlineare monolithische Modell des Aufbaus um Faktor 1,4 schneller als Echtzeit, das reduzierte Modell um Faktor 1070 schneller als Echtzeit berechnet werden.

Insgesamt können drei Effekte für diesen enormen Geschwindigkeitszuwachs verantwortlich gemacht werden:

- Linearisierung der Modellgleichungen
- Verringerte Anzahl an Rechenoperationen bei reduzierter Modellordnung
- Vergrößerung der Simulationsschrittweite

Bei der Aktorgruppe ist der große Unterschied in der Geschwindigkeit des Basismodells und des reduzierten Basismodells hauptsächlich auf die Linearisierung der komplexen Gleichungen der Lenkerkinematik zurückzuführen. Über die Simulationsschrittweite kann lediglich ein Faktor von ca. 3,6 erklärt werden. Bei den Aufbaumodellen wiederum resultiert der Zuwachs im Wesentlichen aus der großen Simulationsschrittweite.

8.4 Hierarchische Optimierung

Mit dem vorgestellten hierarchischen Bottom-Up-Modell des Federungsprüfstands ist die Grundlage geschaffen den hierarchischen Optimierungsansatz der MOBU-Methode zu untersuchen. Bei dieser Mehrzieloptimierung soll die Menge der paretooptimalen Reglerparameter berechnet werden. Die Ziele der obersten Ebene sind der Komfort und der dafür notwendige Leistungsbedarf. Die Optimierungen der einzelnen Ebenen werden aus Gründen der Vereinfachung unter den folgenden Einschränkungen durchgeführt:

- Pro Aggregat werden 2 Zielfunktionen betrachtet. Diese Ziele bewerten den regelungstechnischen Nutzen des Systems sowie den hierfür notwendigen energetischen Aufwand.
- Bei der Optimierung wird pro Aggregat nur eine Anregungsform betrachtet, die das jeweilige Aggregat in alle relevanten Richtungen auslenkt. Auf eine Variation der Anregungsform wird hier verzichtet.

Die Berechnung der Paretomengen erfolgt auf allen Ebenen mit der Goal-Attainment-Methode mit dem Bezugspunkt $\underline{P}$ im Maximum (siehe Seite 37). Dies bedeutet, dass zunächst die individuellen Minima der zwei Zielfunktionen bestimmt werden. Mit diesen Ergebnissen kann der Simplex $\underline{\mathbf{\Theta}}$ der NBI-Methode aufgespannt werden. Anschließend wird der Gewichtungsvektor $\underline{w}$ mit Hilfe des Simplex gleichmäßig verschoben und die Punkte der Paretomenge berechnet.

Die einzelnen skalaren Optimierungsprobleme werden mit einem SQP-Verfahren gelöst[4]. Die Algorithmische Differentiation wird im Vorwärtsmodus für die Berechnung der Gradienten eingesetzt.

8.4.1 Optimierung der Servozylinder

Das Optimierungsproblem der Servozylinder wird folgendermaßen definiert:

- **Umgebungsmodell:** Ein Servozylinder ist als Kraftsteller mit dem Kraftausgang F_{zyl} modelliert. Für die Optimierung muss der Servozylinder um ein Umgebungsmodell ergänzt werden, welches das rudimentäre Verhalten der überlagerten Aktorgruppe abbildet. Die Aktorgruppe verhält sich aus der Sicht der Servozylinder wie eine Masse – die Lenkerplatte – die über eine Feder – die GFK-Feder – abgestützt wird. Dieser Modellierungsansatz ist in Abbildung 8.15 dargestellt. Die Masse m_u wird in etwa auf die Masse von Lenkerplatte und Kolben gesetzt, in die Federsteifigkeit c_u fließen die Steifigkeit der GFK-Feder und die Hebelverhältnisse zu den Hydraulikzylindern ein.
- **Anregungsmodell:** Das System wird über einen Sprung in der Soll-Zylinderlänge l_{soll} von 10 mm angeregt.
- **Bewertungsmodell:** Aufgabe der Optimierung der Servozylinder ist die Berechnung der Paretomenge zwischen der Regelgüte und dem Leistungsbedarf des Aktors. Die

[4] Zum Einsatz kommt hier das SQP-Verfahren aus der NAG-Bibliothek (Routine: `nag_opt_nlp` [Nag02]).

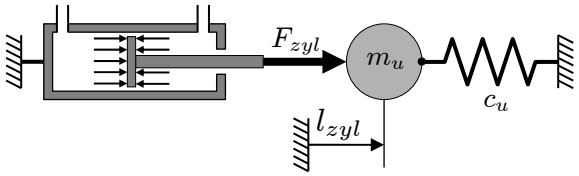


Abbildung 8.15: Umgebungsmodell der Servozylinder

Bewertungsfunktionen der Servozylinder sind:

$$\text{Regelgüte:} \qquad f_{zyl,1} = \frac{1}{T} \int_{t=0}^{T} |l_{zyl} - l_{soll}|\, t\, dt \tag{8.21}$$

$$\text{Leistung:} \qquad f_{zyl,2} = \frac{1}{T} \int_{t=0}^{T} |\tilde{P}_{zyl}|\, dt \tag{8.22}$$

Die Regelgüte wird über eine zeitgewichtete Betragsfehlerfläche bewertet. Der Vergleich verschiedener Fehlerflächen in [Löf08] ergab für diese Funktion den besten Kompromiss aus Schnelligkeit, Dämpfung und stationärer Genauigkeit.

- **Parameter:** Die Reglerparameter $K_p \in [50; 6000]\,\frac{1}{\mathrm{m}}$ und $K_d \in [0; 15]\,\frac{s}{\mathrm{m}}$ bilden die Parameter des Optimierungsproblems.

Abbildung 8.16 zeigt die Optimierungsergebnisse der Servozylinder. Eine Zuordnung der Bildpunkte (links) mit den Urbildpunkten (rechts) ist über die angetragene Größe α_{zyl} möglich. Der Reglerparameter K_d weist zwischen $\alpha_{zyl} = 0{,}92$ und $\alpha_{zyl} = 0{,}94$ einen Sprung auf. Dieser hat für die Optimierung der Aktorgruppe im folgenden Abschnitt keine Bedeutung, da in Paretomenge der Aktorgruppe Punkte bis maximal $\alpha_{zyl} \approx 0{,}7$ vorkommen.

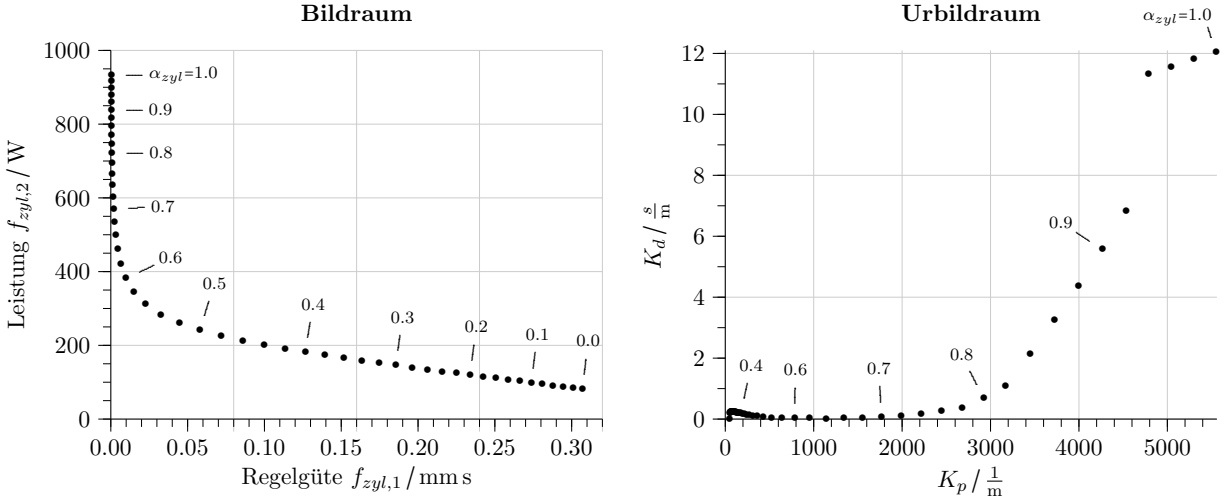


Abbildung 8.16: Paretomenge der Servozylinder

8.4.2 Optimierung der Aktorgruppen

- **Umgebungsmodell:** Für das Umgebungsmodell der Aktorgruppe wird eine sehr große Aufbaumasse angenommen. Der Aktorrahmen wird daher starr an der Umgebung angebunden (siehe Abb. 8.17). Die Federspitze wird am ortsfesten Fahrwerksrahmen in z und y-Richtung gelagert. Bewegungen in x-Richtung werden zugelassen, so dass in dieser Richtung keine Kräfte aufgebaut werden. Die Aktorgruppe wird über die beiden Lager derart verspannt, dass sich eine typische Grundbelastung der GFK-Feder ergibt. Dieses Umgebungsmodell enthält keine Zustände. Dies ist für den rein gesteuerten Betrieb der Aktorgruppe bedeutend. Zustände, z. B. zur realistischeren Abbildung der Aufbaumasse, können eine Drift aufweisen, die eine Rückführung notwendig macht.

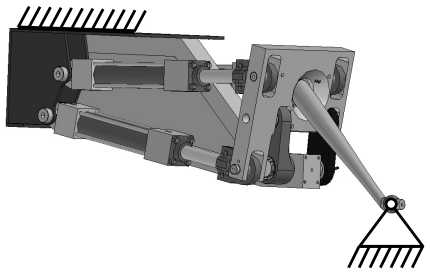


Abbildung 8.17: Umgebungsmodell der Aktorgruppe

- **Anregungsmodell:** Aufgabe der Aktorgruppe ist das Stellen einer Soll-Kraft $\underline{F}_{F,soll}$. Die Anregung der Soll-Kraft geschieht über ein Rauschsignal, welches über ein PT1-Glied mit der Zeitkonstante $T_{anr,ag} = 0{,}1\,s$ gefiltert wird.

- **Bewertungsmodell:** Die Bewertung der Aktorgruppen erfolgt über die beiden folgenden Funktionen:

$$\text{Regelgüte:} \qquad f_{ag,1} = \frac{1}{T} \int_{t=0}^{T} \left\| \underline{F}_F - \underline{F}_{F,soll} \right\|_2^2 \, dt \tag{8.23}$$

$$\text{Leistung:} \qquad f_{ag,2} = \frac{1}{T} \int_{t=0}^{T} \sum_{i=1}^{3} |\tilde{P}_{zyl,i}| \, dt \tag{8.24}$$

  Die Regelgüte wird über eine quadratische Fehlerfläche bestimmt. Der Fehler zwischen Soll- und Ist-Kraft ist in alle Richtungen gleichermaßen gewichtet.

- **Parameter:** Die Optimierung verläuft über die Einstellung der drei Servozylinder:

$$\underline{p}_{ag} = \left[\alpha^{zyl,1}, \alpha^{zyl,2}, \alpha^{zyl,3}\right]^{\mathrm{T}} \tag{8.25}$$
$$\text{mit} \quad \alpha^{zyl,i} \in [0;1], \quad i = 1, \ldots, 3$$

Im Diagramm 8.18 ist links die Paretofront und rechts die Einstellungen der drei unterlagerten Servozylinder zu sehen. Bei den energiegünstigen Paretopunkten im Bereich $\alpha_{ag} = 0$ bis 0,26 werden die beiden äußeren Servozylinder 1 und 2 mit minimaler Leistungsaufnahme betrieben. Diese beiden Zylinder besitzen hier ein niedriges Reaktionsvermögen. Das Stellen dynamischer Kräfte erfolgt im Wesentlichen über den innen liegenden Zylinder 3.

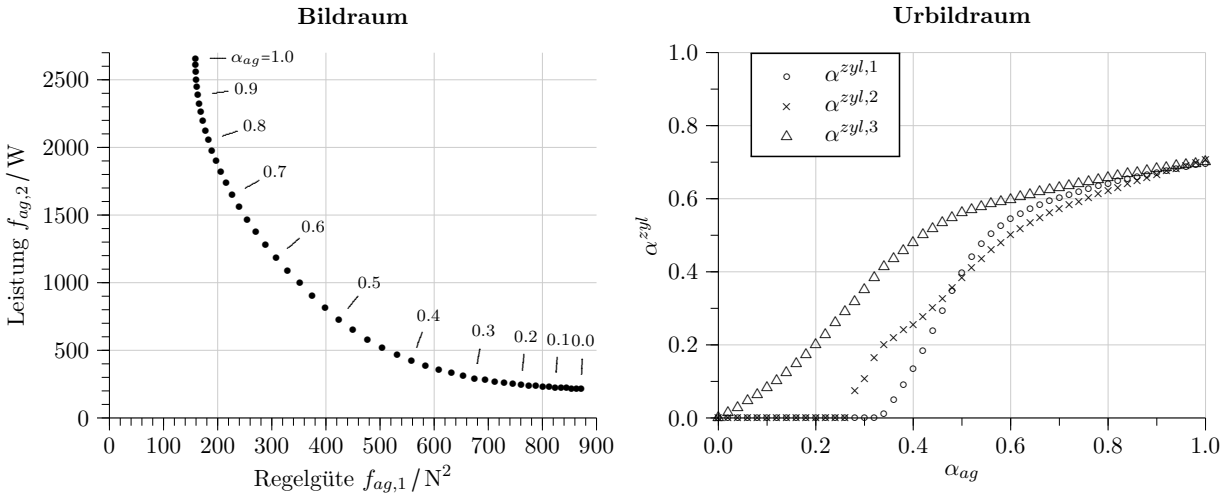


Abbildung 8.18: Paretomenge der rechten Aktorgruppe

8.4.3 Optimierung des Aufbaus

- **Bewertungsmodell:** Die Zielgrößen der Optimierung des Aufbaus stellen der Komfort für die Insassen sowie die Leistungsaufnahme des Federungssystems dar. Der Komfort wird über die frequenzgewichtete Aufbaubeschleunigung abgeschätzt:

$$\ddot{k}_{a,i}(s) = \frac{0{,}08754\,s}{0{,}0006333\,s^2 + 0{,}08754\,s + 1} \cdot \ddot{x}_{a,i}(s) \quad \text{mit} \quad i \in \{x, y, \varphi\} \tag{8.26}$$

  Diese Filterfunktion bildet die Wahrnehmung von Menschen auf die unterschiedlichen Frequenzen ab (vgl. [VDI02]). Damit ergeben sich die Bewertungsfunktionen des Aufbaus:

$$\text{Komfort:} \qquad f_{a,1} = \left(\frac{1}{T} \int_{t=0}^{T} \left\| \underline{\ddot{k}}_a \right\|_2^{\,2} dt \right)^{\frac{1}{2}} \tag{8.27}$$

$$\text{Leistung:} \qquad f_{a,2} = \frac{1}{T} \int_{t=0}^{T} \left[\sum_{i=1}^{3} |\tilde{P}_{zyl,i}^{rechts}| + \sum_{i=1}^{3} |\tilde{P}_{zyl,i}^{links}| \right] dt \tag{8.28}$$

- **Parameter:** Die Optimierungsparameter werden von den Parametern des Aufbaureglers und den Einstellungen der Aktorgruppen gebildet:

$$\underline{p}_a = [c_{rel,z}, c_{rel,y}, c_{rel,\varphi}, d_{sky,z}, d_{sky,y}, d_{sky,\varphi}, \alpha_{ag}]^{\mathrm{T}} \tag{8.29}$$

  mit

$$c_{rel,i} \in [-50000; 0]\,\tfrac{\mathrm{N}}{\mathrm{m}}, \qquad d_{sky,i} \in [0; 5000]\,\tfrac{\mathrm{N}}{\mathrm{m}} \quad \text{mit} \quad i \in \{x, y\}$$
$$c_{rel,\varphi} \in [-50000; 0]\,\tfrac{\mathrm{N}}{\mathrm{rad}}, \qquad d_{sky,\varphi} \in [0; 5000]\,\tfrac{\mathrm{Ns}}{\mathrm{rad}}$$

  Wegen der Symmetrie erhalten die Aktorgruppen identische Vorgaben: $\alpha_{ag} = \alpha_{ag}^{links} = \alpha_{ag}^{rechts}$.

- **Anregungsmodell:** Die Anregung des Fahrwerks in z, y und φ-Richtung erfolgt über je ein bandbegrenztes gefiltertes Rauschsignal. Die Filterung geschieht über ein PT1-Glied.

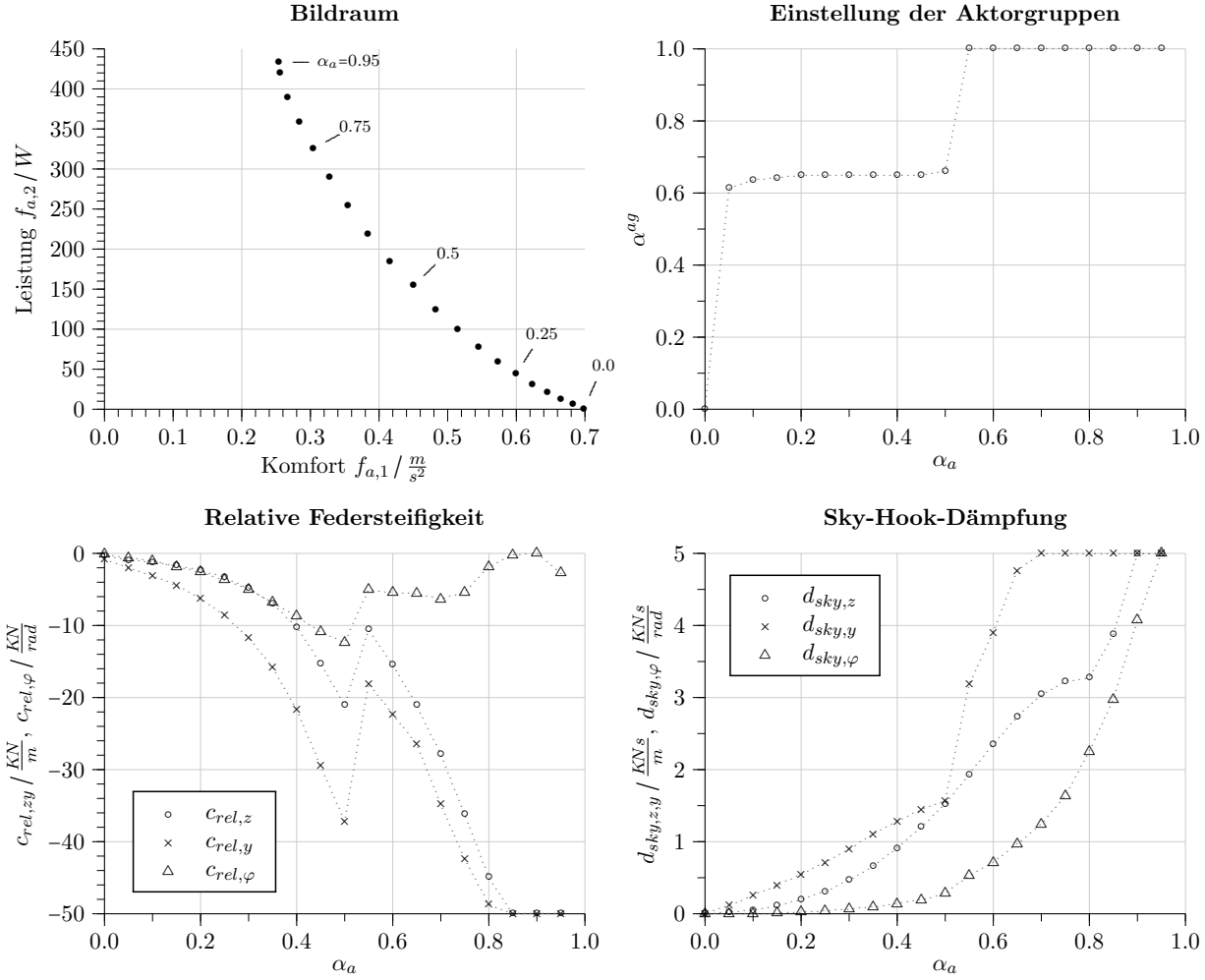


Abbildung 8.19: Paretomenge des Aufbaus

Die Paretomenge der Aufbauebene ist in 8.19 angetragen. Die guten Komfortwerte werden bei hohen Werten für α_a erreicht. Der Paretopunkt bei $\alpha_a = 1{,}0$ wurde entfernt, da er von dem Punkt bei $\alpha_a = 0{,}95$ dominiert wird. Wie im Abschnitt 3.5.1 beschrieben, werden zunächst die individuellen Minima der einzelnen Zielfunktionen bestimmt. Das hierfür eingesetzte Verfahren der unbeschränkten Optimierung erreicht nicht die nötige Genauigkeit. Es kann jedoch davon ausgegangen werden, dass der Punkt mit $\alpha_a = 0{,}95$ das Minimum der Zielgröße $f_{ag,1}$ darstellt.

Bei guten Komfortwerten zeigt sich, dass die Federsteifigkeiten $c_{rel,z}$ und $c_{rel,y}$ sowie alle Dämpfungsraten c_{sky} an die Grenzen des zulässigen Parameterraums stoßen. Eine weitere Verbesserung des Komforts ist aufgrund dieser Grenzen nicht möglich.

Die Aktorgruppen werden nur in den drei Bereichen $\alpha_{ag} = 0$, um $\alpha_{ag} \approx 0{,}65$ und bei $\alpha_{ag} = 1$ betrieben. Der Wechsel zwischen diesen Bereichen verläuft nahezu sprungartig. Bei $\alpha_a = 0{,}5$ wirkt sich dies stark auf die Federsteifigkeiten und die Sky-Hook-Dämpfung aus. Die Parameter verändern sich hier ebenfalls sprungartig, um den Verlauf der Aktorgruppeneinstellung auszugleichen.

Einfluss der Aktorgruppeneinstellung

Der Einfluss der Aktorgruppeneinstellung α_{ag} auf die Paretofront des Aufbaus wurde mit Hilfe einer Parameterstudie näher untersucht. Es kann gezeigt werden, dass der Einfluss im Bereich niedrigen Leistungsbedarfs vergleichsweise gering ist. Abbildung 8.20 zeigt Paretofronten, die mit festen Werten für α_{ag} bestimmt wurden. Die Änderung in den Verstärkungen der Servozylinder kann in weiten Bereichen durch eine Anpassung des Aufbaureglers ausgeglichen werden. Der Wert von α_{ag} bestimmt im Wesentlichen die Güte des erreichbaren Komforts. So können mit niedrigen Einstellungen die besten Komfortwerte der Paretofront in 8.19 nicht erreicht werden. Dies ist u. a. auf die Begrenzungen der Aufbauparameter zurückzuführen, die bei niedrigen α_{ag} schneller erreicht werden. Die Paretofronten unterscheiden sich im Bereich niedrigen Leistungsbedarfs nur geringfügig. Dies bedeutet, dass mit der Einstellung $\alpha_{ag} = 1$ durchgehend gute Ergebnisse erzielt werden; eine Anpassung der Aktorgruppen ist bei diesem System nicht erforderlich.

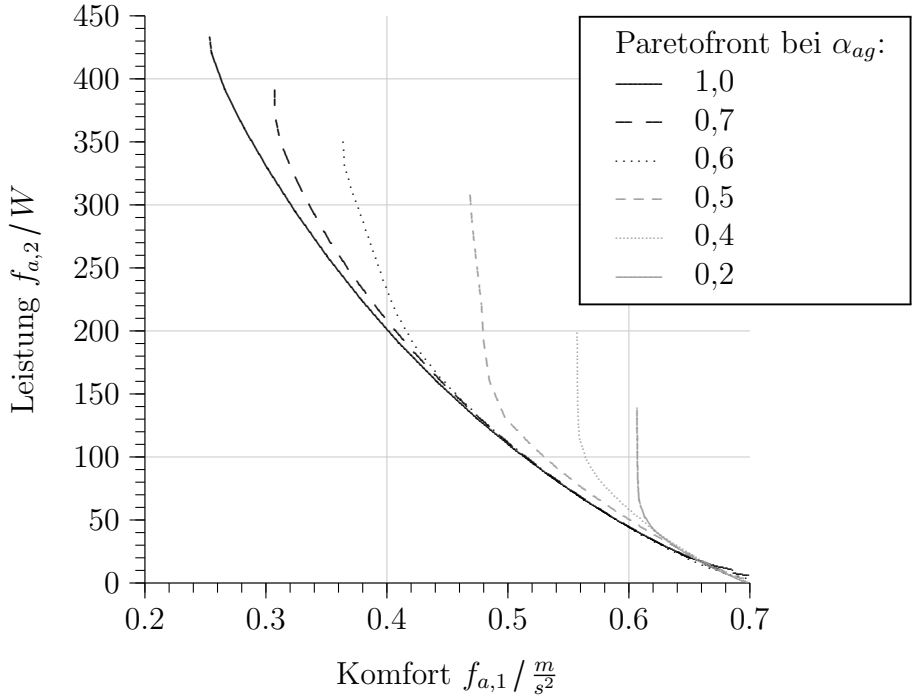


Abbildung 8.20: Paretofronten mit festen α_{ag}

9 Selbstoptimierung gestörter mechatronischer Systeme

Eine Problematik bei der Nutzung der berechneten, paretooptimalen Systemeinstellungen ergibt sich aus dem Unterschied zwischen dem simulierten Systemverhalten und dem Verhalten des realen Systems. Während des Betriebs ist das System vielfältigen Störungen ausgesetzt, die vorab nicht oder nicht exakt bekannt sind und die nicht selten einen stochastischen Charakter aufweisen. Die Störungen auf das System stellen also einen Unsicherheitsfaktor für die Auslegung dar. Bei einer modellbasierten Optimierung muss der Entwickler notgedrungen vereinfachende Annahmen über die auftretenden Störungen machen. Oft kommen synthetische Signale wie Sprunganregungen oder stochastische Anregungen wie bandbegrenztes Rauschen zum Einsatz.

In Abbildung 9.1 ist ein Vergleich zwischen einem virtuellen System (Modell) und dem realen System am Beispiel eines aktiv gefederten Viertelfahrzeugs dargestellt. Das reale System wird hier in Gedanken als eine mathematische Funktion angenommen. Aus Sicht eines Optimierungsverfahrens sind die Anregungen, wie auch das gesamte reale oder virtuelle System, Teil der zu minimierenden Zielfunktionen. Die Zielfunktionswerte werden in einem Bewertungsblock aus den Ausgängen des Systems ermittelt. Während der Regler und die Bewertungsfunktionen identisch sind, weichen wegen Modellungenauigkeiten das Streckenverhalten und auch die Anregung voneinander ab. Allein schon wegen der Unterschiede zwischen dem synthetischen Anregungssignal und der realen Anregung, stimmen die Zielfunktionen $\underline{f}(\underline{p})$ des virtuellen Systems mit den Zielfunktionen $\underline{\Phi}_{ist}(\underline{p})$ des realen Systems nicht überein.

Aufgrund der Diskrepanz zwischen Realität und Simulation ergeben sich Abweichungen zwischen den gewünschten, den berechneten und den real erreichbaren Zielgrößen, die eine Korrektur des gewählten Paretopunktes erforderlich machen. Für diese Korrektur muss die

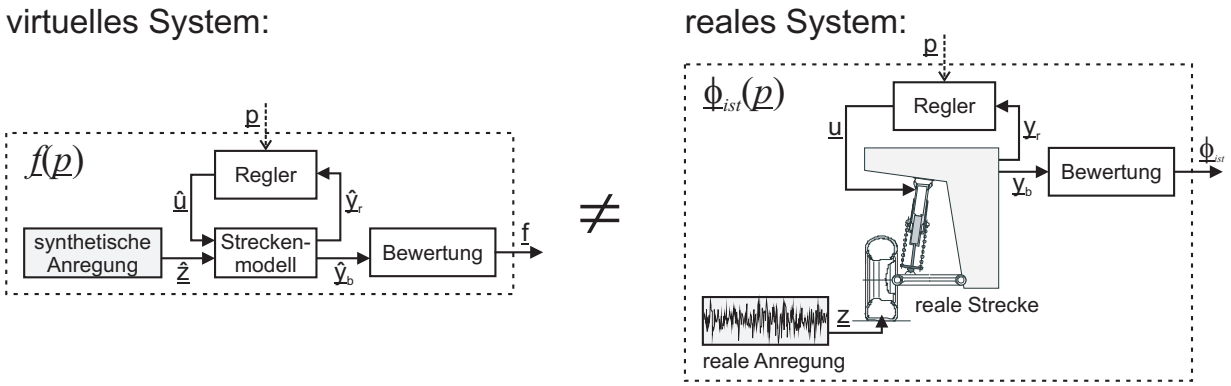


Abbildung 9.1: Zielfunktionen des realen Systems und des virtuellen Systems weichen im Allgemeinen voneinander ab.

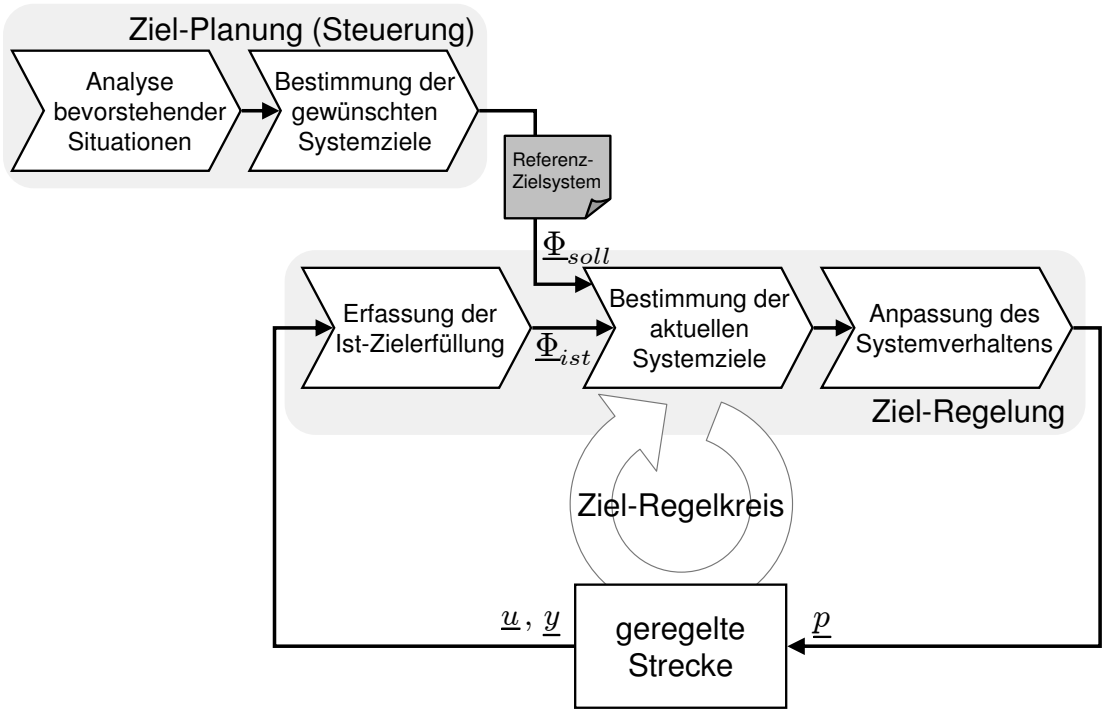


Abbildung 9.2: Aufteilung des Selbstoptimierungsprozesses in einen regelnden und einen steuernden Teil.

Zielerfüllung $\underline{\Phi}_{ist}(\underline{p}_k)$ mit der aktuellen Systemkonfiguration $\underline{p}_k$ des Systems herangezogen werden. Die Zielerfüllung wird, wie in Abbildung 9.1 rechts dargestellt, durch Anwendung von Bewertungsfunktionen auf die Messgrößen des Systems im laufenden Betrieb berechnet. Durch einen Vergleich zwischen den vorgegebenen gewünschten Zielen $\underline{\Phi}_{soll}$ und der real erreichten Zielerfüllung $\underline{\Phi}_{ist}(\underline{p}_k)$ lässt sich eine neue Systemkonfiguration $\underline{p}_{k+1}$ ermitteln, welche die Differenz zwischen gewünschten Zielen und realer Zielerfüllung verringert.

Prinzipiell handelt es sich bei dieser Aufgabe um ein Regelungsproblem. Anstelle eines gemessenen oder beobachteten Systemausgangs $\underline{y}$ wird die Regelgröße jedoch durch die aktuelle Zielerfüllung des Systems $\underline{\Phi}_{ist}$ gebildet. Die Zielregelung arbeitet somit auf einer hohen Abstraktionsstufe. Sie setzt selbst keine Funktion um, sondern regelt vielmehr die Qualität bzw. Güte einer Funktionalität.

Analog zu einem klassischen Regelungskreis kann eine solche Zielregelung in die zwei Bestandteile Regelung und Steuerung unterteilt werden. Abbildung 9.2 zeigt diese Unterteilung des Selbstoptimierungsprozesses in einen steuernden Teil, der *Ziel-Planung*, und einen regelnden Teil, der *Ziel-Regelung*.

Die *Ziel-Planung* dient der Generierung von Referenzzielen $\underline{\Phi}_{soll}$. Diese Referenzziele beschreiben den gewünschten Verlauf der Zielerfüllung für einen gewissen Zeithorizont in die Zukunft. Bei einer solchen vorausschauenden Planung können Informationen über bekannte zukünftige Ereignisse berücksichtigt werden. Ebenso können externe Ziele von benachbarten oder übergeordneten Systemen in die Berechnung der Referenzziele eingebunden werden. Die Ziel-Planung wird in dieser Arbeit nicht weiter betrachtet. Beispiele für Planungssysteme und -aufgaben im Kontext selbstoptimierender Systeme finden sich u. a. in [MAK+08, Klö09, AEH+11].

Die Aufgabe der *Ziel-Regelung* ist das Einregeln der Referenzziele im realen System

sowie die schnelle Korrektur von bestehenden Zielabweichungen. Ein exaktes Einstellen der gewünschten Ziele ist aufgrund der unterschiedlichen Zielfunktionen von Modell und Realität meist nicht möglich, jedoch kann die Abweichung deutlich verringert werden. Eine Anpassung der Zielerfüllung des Systems kann zum einen über eine Verschiebung der aktuellen Parameter entlang einer festen Paretomenge, zum anderen durch Auswahl einer neuen, besser auf die aktuelle Situation angepassten Paretomenge erfolgen. Wie ein klassischer Regler reagiert die *Ziel-Regelung* auf Abweichungen der Zielerfüllungen.

Während bei der *Ziel-Planung* bekannte zukünftige Ereignisse in die Planung einfließen, arbeitet die *Ziel-Regelung* auf Basis von Informationen aus der direkten Vergangenheit.

9.1 Paretomengen gestörter Systeme

Eine gezielte Anpassung des eingestellten Paretopunktes auf ein geändertes Anregungsverhalten erfordert Kenntnisse darüber, wie sich die unterschiedlichen Störanregungen auf die Paretomengen des mechatronischen Systems auswirken. Dieses Wissen liefert wertvolle Hinweise über die Art und Weise wie eine Zielanpassung im Rahmen eines selbstoptimierenden Reglers durchgeführt werden kann.

Im Folgenden werden die Veränderungen der Paretomengen im Bild- und Urbildraum bei unterschiedlichen Störanregungen betrachtet. Die Untersuchungen zielen darauf ab, Gesetzmäßigkeiten zu finden, in wie weit sich die Paretomengen durch eine Variation der Störanregung verändern. Der Einfachheit halber werden nur Systeme mit einem Störeingang betrachtet, da es bei den Gesetzmäßigkeiten weniger um eine mathematische präzise Formulierung als vielmehr um tendenzielle Aussagen über die Veränderungen geht.

9.1.1 LZI-Systeme mit einem Störeingang

Abbildung 9.3 zeigt das Optimierungsmodell eines linearen Systems mit einer Störanregung w und den n Zielgrößen $\underline{f} = [f_1, \cdots, f_n]^{\mathrm{T}}$. Das Schwingungsverhalten der einzelnen Teilsysteme ist über lineare Übertragungsfunktionen beschrieben.

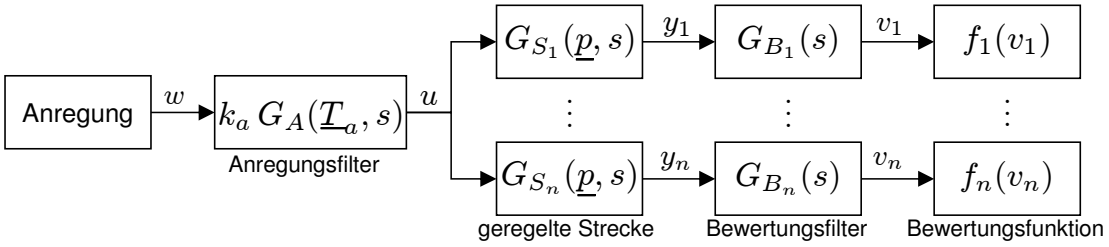


Abbildung 9.3: Optimierungsmodell mit einer Störanregung und zwei Zielgrößen.

Bei der Variation des Anregungssignals können die Amplitude und Frequenzcharakteristik des Signals unabhängig voneinander betrachtet werden. Die Amplitude der Anregung wird im Anregungsfilter über die Verstärkung k_a, der Frequenzbereich über Zeitkonstanten $\underline{T}_a$ angegeben. Die Funktionen $G_{S_1}(\underline{p}, s)$ bis $G_{S_n}(\underline{p}, s)$ beschreiben das Übertragungsverhalten des geregelten Systems und hängen von den Optimierungsparametern $\underline{p}$ ab.

Das lineare Übertragungsverhalten von der Anregung w bis zu den Ausgängen v_i ergibt sich zu:

$$\frac{v_i}{w} = G_i(s) = k_a \cdot G_A(\underline{T}_a, s) \cdot G_{S_i}(\underline{p}, s) \cdot G_{B_i}(s) \tag{9.1}$$

Variation der Amplitude

Die Ausgänge v_i der Übertragungsfunktionen (9.1) verhalten sich proportional zu k_a , d. h. es gilt:

$$v_i = k_a \cdot \bar{v}_i \tag{9.2}$$

mit den auf $k_a = 1$ normierten Ausgängen $\bar{v}_i$.

Die Berechnung der Zielgrößen erfolgt über die nichtlinearen Bewertungsfunktionen f_i. Wie sich eine Änderung von k_a auf diese Bewertungsfunktionen auswirkt, hängt von der Beschaffenheit der jeweiligen Funktion ab. Geht man von einer allgemeinen Fehlerfläche (4.4) aus, so lässt sich die Zielgrößenänderung folgendermaßen beschreiben:

$$f_i = \frac{1}{T} \int_0^T |k_a \cdot \bar{v}_i|^n \, t^m \, dt = \frac{|k_a|^n}{T} \int_0^T |\bar{v}_i|^n \, t^m \, dt = |k_a|^n \cdot \bar{f}_i \tag{9.3}$$

Der Term $|k_a|^n$ lässt sich als zeitunabhängiger Faktor vor das Integral ziehen. Die Funktion f_i verhält sich somit ordnungserhaltend bezüglich der Anregungsverstärkung, d. h. die „Größer-Kleiner“-Beziehung zwischen den Zielgrößen zweier Parametersätze ändert sich aufgrund der Verstärkung nicht. Betrachten wir hierfür die normierte Zielfunktion $\bar{f}_i$ mit beiden Parametersätzen $\underline{p}_1$ und $\underline{p}_2$, so nimmt der Faktor $|k_a|^n$ keinerlei Einfluss auf die Beziehung:

$$|k_a|^n \cdot \bar{f}_i(\underline{p}_1) < |k_a|^n \cdot \bar{f}_i(\underline{p}_2) \tag{9.4}$$

Da die Definition des Paretooptimums (vgl. Kapitel 3.3.1) auf genau diesen Ordnungsprinzipien aufsetzt, besitzt k_a keinen Einfluss auf die Lage der Paretopunkte im Urbildraum. Die Veränderung der Amplitude des Anregungssignals hat also bei Zielfunktionen, die sich ordnungserhaltend bzgl. der Amplitude des betrachteten Systemausgangs verhalten, keinerlei Auswirkung auf die Lage der optimalen Parameter. Die Paretomenge des Systems ist robust gegenüber Änderungen der Anregungsamplitude.

Neben den hier betrachteten Fehlerflächen existieren weitere ordnungserhaltende Bewertungsfunktionen, wie z. B. das Maximalwertkriterium (4.5). Die Triggerfunktion (4.6) oder das Schwellwertkriterium (4.7) fallen jedoch nicht hierunter, da sie über feste Schwellwerte definiert werden, die nicht mit der Anregungsamplitude skalieren.

Viele Optimierungsprobleme definieren harte Grenzen in Form von Nebenbedingungen, die es einzuhalten gilt. Grenzen, die nicht im gleichen Maße wie die Nebenbedingung mit der Anregungsamplitude skalieren, führen, sofern sie aktiv sind, zu einer Verzerrung der Paretomenge. Begrenzungen der Parameter sind hiervon bspw. nicht betroffen. Diese werden bereits im Urbildraum definiert und hängen daher nicht von den Anregungsverhältnissen ab. Ähnliches gilt für Nebenbedingungen die unabhängig vom Anregungssignal berechnet werden können, wie z. B. Nebenbedingungen auf der Basis von Eigenwerten.

Variation im Frequenzbereich

Die Auswirkung einer Veränderung der Anregung im Frequenzbereich auf die Lage der Paretopunkte ist von dem jeweiligen System abhängig. In den Übertragungsfunktionen

(9.1) greifen die Zeitkonstanten $\underline{T}_a$ in das Schwingungsverhalten des Gesamtsystems ein. Die Auswirkungen können je nach Übertragungsfunktionen sehr unterschiedlich ausfallen. Eine Veränderung von $\underline{T}_a$ führt daher in den meisten Fällen zu einer anderen Paretomenge. Aussagen über die Auswirkungen bei Frequenzänderungen können nur zu konkreten Anwendungen gemacht werden.

9.1.2 Anwendungsbeispiel: Viertelfahrzeug

Die Veränderungen von Paretomengen bei variierenden Anregungsspektren werden anhand eines einfachen Beispiels untersucht. Darüber hinaus können an diesem Beispiel auch die Aussagen für variierende Anregungsamplituden verifiziert werden.

Um die veränderten Lagen der Paretopunkte im Bild- und Urbildraum zu bestimmen, muss eine Berechnung der Paretomenge wiederholt für jede Anregungsvariante erfolgen. Anhand der Zielgrößen lässt sich der Einfluss der Anregungen auf das optimale Systemverhalten ablesen. Die Betrachtung des Urbildraums ermöglicht eine Aussage darüber, welche Bandbreite an unterschiedlichen Parametersätzen sich durch die veränderlichen Störungen ergeben.

Als Anwendungsbeispiel dient ein einfaches Viertelfahrzeugmodell eines aktiv gefederten Fahrwerks (siehe auch [MT06]). Das Federungssystem dient der aktiven Beeinflussung der Aufbaudynamik. Das vornehmliche Ziel besteht in der Verbesserung des Komforts für die Insassen. In Abbildung 9.4 ist auf der linken Seite der prinzipielle Aufbau des aktiven Federungssystems für ein einzelnes Federbein zu sehen. Das System besteht aus der Aufbaumasse und der Radmasse, die über einen Querlenker miteinander verbunden sind. Zwischen Querlenker und Aufbaumasse befindet sich das Federbein, bestehend aus einem passiven Feder-Dämpfer-Paar, sowie einem Hydraulikzylinder. Der Hydraulikzylinder kann über eine Fußpunktverstellung der Feder Kräfte zwischen Aufbau und Rad stellen. Hierzu wird er über ein Servoventil angesteuert.

Modell: Auf der rechten Seite der Abbildung 9.4 ist das lineare Ersatzmodell des Viertelfahrzeugs dargestellt. Die Parameter des Modells wurden folgendermaßen gewählt:
$m_B = 430\,\mathrm{Kg}$, $c_B = 20000\,\frac{\mathrm{N}}{\mathrm{m}}$, $d_B = 2000\,\frac{\mathrm{Ns}}{\mathrm{m}}$, $m_W = 30\,\mathrm{Kg}$, $c_W = 200000\,\frac{\mathrm{N}}{\mathrm{m}}$

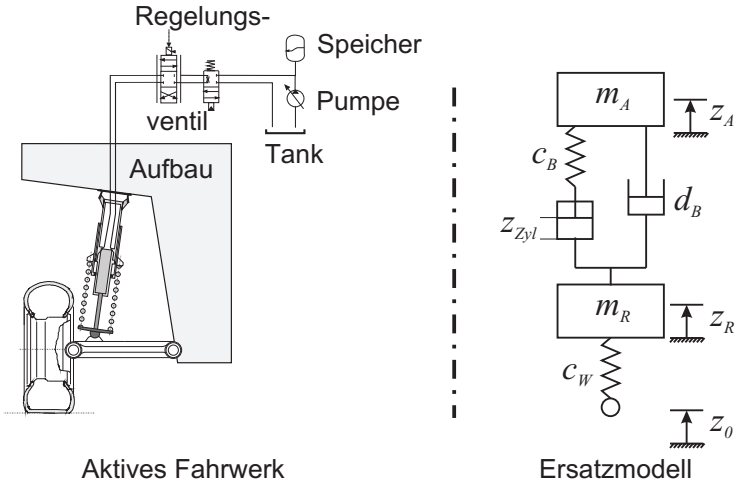


Abbildung 9.4: Ersatzmodell eines aktiv gefederten Viertelfahrzeugs

Die Bewegungsgleichung des mechanischen Teilsystems ergibt sich zu:

$$\begin{bmatrix} \dot{z}_B \\ \ddot{z}_B \\ \dot{z}_W \\ \ddot{z}_W \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -46{,}5 & -4{,}65 & 46{,}5 & 4{,}65 \\ 0 & 0 & 0 & 1 \\ 667 & 66{,}7 & -7333 & -66{,}7 \end{bmatrix} \begin{bmatrix} z_B \\ \dot{z}_B \\ z_W \\ \dot{z}_W \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 46{,}5 \\ 0 & 0 \\ 6670 & -667 \end{bmatrix} \begin{bmatrix} z_0 \\ z_{zyl} \end{bmatrix} \tag{9.5}$$

Die Dynamik des Hydraulikzylinders ist vereinfacht über das PT_2-Glied:

$$z_{zyl} = \frac{u_{zyl}}{{T_{zyl}}^2 s^2 + 2\, d_{zyl}\, T_{zyl}\, s + 1} \tag{9.6}$$

mit $T_{zyl} = 0{,}01\,\mathrm{s}$ und $d_{zyl} = 0{,}7$ abgebildet.

Regler: Die Stellgröße u_{zyl} setzt sich aus zwei Bestandteilen zusammen. Zum einen wird über aktive Bewegungen des Hydraulikzylinders dem Federbein eine andere Federsteifigkeit aufgeprägt. Zum anderen wird eine Bedämpfung der Absolutbewegung des Aufbaus über eine Sky-Hook-Dämpfung erreicht. Das resultierende Regelungsgesetz ergibt sich zu:

$$u_{zyl} = \frac{c_{rel}}{c_B}(z_B - z_W) + \frac{d_{sky}}{c_B}\dot{z}_B \tag{9.7}$$

Die beiden Größen c_{rel} und d_{sky} bilden die Entwurfsparameter der Anwendung.

Zielkriterien: Als Ziele des Systems werden die Verbesserung des Komforts für die Insassen und der hierfür notwendige Leistungsbedarf der Aktorik berücksichtigt. Dementsprechend ergeben sich zwei Zielkriterien:

$$f_{komfort} = \left(\frac{1}{T} \int_0^T G_k(\ddot{z}_B)^2\, dt \right)^{\frac{1}{2}} \tag{9.8}$$

$$f_{leistung} = \frac{1}{T} \int_0^T |\dot{z}_{zyl}|\, dt \tag{9.9}$$

Die Zielgröße $f_{komfort}$ bewertet die Aufbaubeschleunigung des Fahrzeugs und stellt ein Maß für den Komfort der Insassen dar. Der lineare Filter $G_k(\ddot{x}_B)$ bildet die gemessene Aufbaubeschleunigung auf das subjektive Empfinden der Insassen ab. Durch die Quadratwurzel in (9.8) wird erreicht, dass die beiden Zielfunktionen linear mit der Anregungsamplitude skalieren.

Anregung: Die vertikale Anregung des Rades z_0 durch die Straße wird über ein stochastisches Signal abgebildet. Als Ausgangssignal wird *weißes Rauschen* verwendet. Das PT_1-Glied (9.10) dient als Tiefpassfilter zur Reduktion der hohen Frequenzanteile.

$$G_A(s) = \frac{\hat{z}_0 \sqrt{2\, T_{z_0}}}{T_{z_0}\, s + 1} \tag{9.10}$$

Die Charakteristik dieses Anregungssignals lässt sich über die beiden Parameter $\hat{z}_0$ und T_{z_0} verändern. Über $\hat{z}_0$ wird die Standardabweichung des Signals angeben; mit der Zeitkonstante $T_{z_0} = \frac{1}{2\pi f_{z_0}}$ kann die Eckfrequenz f_{z_0} des Tiefpassfilters eingestellt werden. Mit diesen

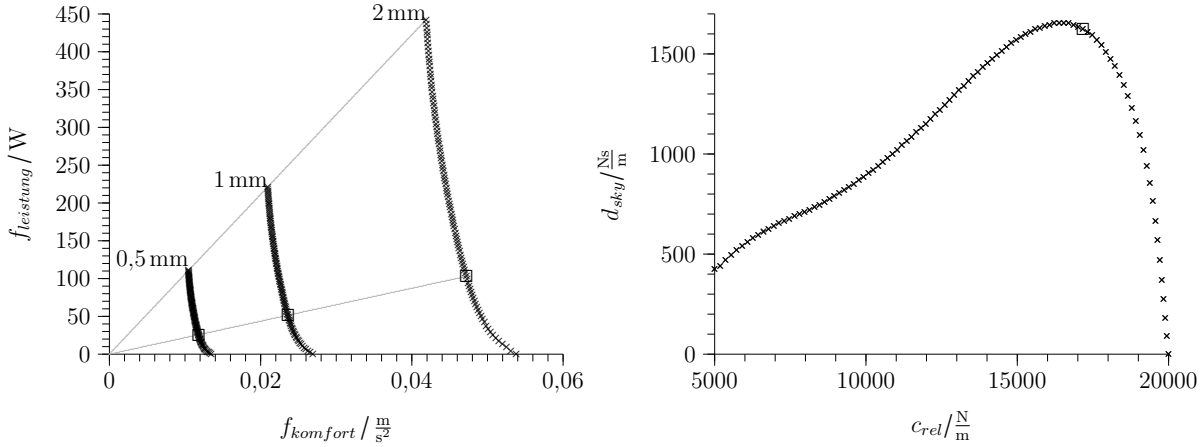


Abbildung 9.5: Paretomenge im Bild- und Urbildraum eines aktiven Federungssystem bei unterschiedlichen Anregungsamplituden

Parametern lässt sich die Amplitude und die Frequenzcharakteristik des Anregungssignals variieren.

Das Modell befindet sich zu Beginn der Simulationen in einer stationären Ruhelage. Einschwingeffekte werden hierdurch vermieden. Dies ist insofern wichtig, da das Einschwingen des Systems als eine Art von Störung angesehen werden kann. Das Einschwingen hängt nicht von $\hat{z}_0$ oder f_{z_0} ab. Besonders bei niedrigen Anregungen gewinnen Einschwingeffekte gegenüber der Straßenanregung an Gewicht und führen zu einer verzerrten Paretofront.

Variation der Amplitude: Abbildung 9.5 zeigt die Optimierungsergebnisse im Bildraum (rechts) und Urbildraum (links) für die Eckfrequenz $f_{z_0} = 0.5\,\text{Hz}$ und die Standardabweichungen $\hat{z}_0 = 0{,}5\,\text{mm}$, $1{,}0\,\text{mm}$ und $2{,}0\,\text{mm}$.

Die unterschiedlichen Anregungsamplituden führen wie erwartet zu deckungsgleichen Urbildmengen. Die Paretofronten sind zueinander ähnlich. Um dies zu verdeutlichen, wurde beispielhaft ein Punkt der Urbildmenge und die zugehörigen Zielgrößen mit einem Kästchen markiert. Diese drei Punkte lassen sich durch eine Gerade verbinden, die durch den Ursprung des Bildraumes geht. Die Abstände der Punkte zum Ursprung skalieren linear mit $\hat{z}_0$.

Variation der Frequenzcharakteristik: Abbildung 9.6 zeigt die Paretomengen für unterschiedliche Eckfrequenzen. Die Standardabweichung des Anregungssignals beträgt für alle Optimierungen $\hat{z}_0 = 1{,}0\,\text{mm}$. Die Energie im Anregungssignal steigt zu den höheren Frequenzen hin an. Dies resultiert in einem höheren Leistungsbedarf und in einem schlechteren Komfort.

Die Formen der Paretofronten weisen eine vergleichbare Form auf; sie sind jedoch nicht vollkommen ähnlich zueinander. Dennoch ergeben sich gewisse Skalierungseigenschaften. Die Punkte mit minimalem $f_{komfort}$ oder minimalem $f_{leistung}$ liegen in etwa auf einer Geraden, die jeweils durch den Ursprung geht. Die Paretofronten besitzen somit das gleiche maximale und minimale Zielgrößenverhältnis $f_{komfort}/f_{leistung}$. Über das Zielgrößenverhältnis können Punkte der verschiedenen Paretomengen einander zugeordnet werden. Beispielhaft wurden die Punkte eines bestimmten Zielgrößenverhältnisses mit einer Raute markiert.

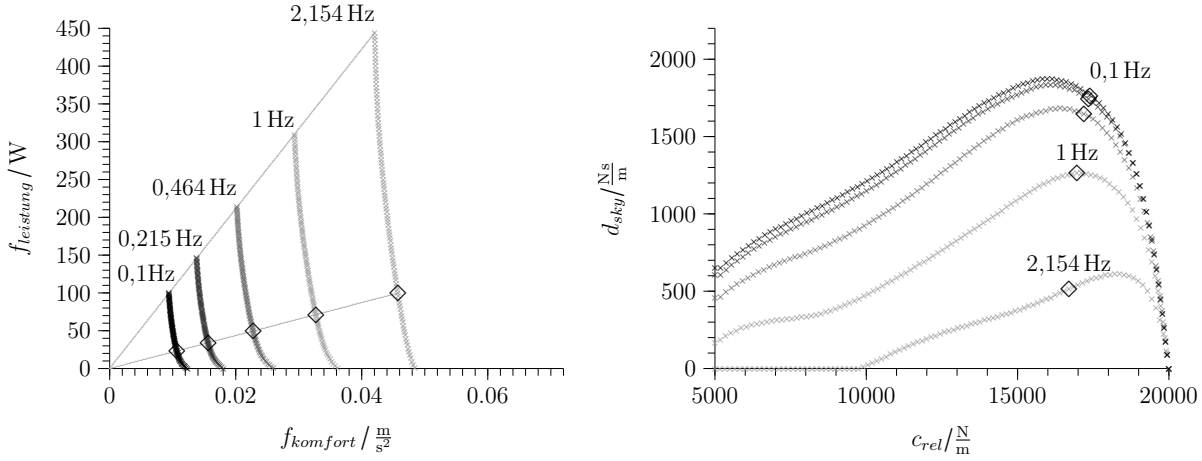


Abbildung 9.6: Paretomenge im Bild- und Urbildraum des aktiven Federungssystem bei Anregungen mit unterschiedlicher Frequenzcharakteristik

9.1.3 Robustheit von Paretomengen

Die Robustheit von Paretomengen gegenüber veränderlichen Störungen ist für den Betrieb des selbstoptimierenden Systems von besonderem Interesse. Im Betrieb des Systems wird im Selbstoptimierungsprozess eine Entscheidung für einen bestimmten Paretopunkt getroffen. Da die tatsächlich auf das System einwirkenden Störungen vorab nicht vollständig bekannt sind, ist es wichtig, dass der Paretopunkt auch unter anderen als den angenommenen Störungen ein gutes Systemverhalten erreicht. Ein solcher Punkt verhält sich robust gegenüber Änderungen der Anregungssituation.

> Unter robusten Paretopunkten werden hier Punkte des Parameterraums verstanden, die bei Variation der Systemanregung weiterhin nahezu optimale Zielgrößenwerte liefern, d. h. die Zielgrößen liegen in der Nähe der tatsächlichen Paretofront der jeweiligen Anregung. Der Abstand zwischen den Zielgrößenwerten und der tatsächlichen Paretofront liefert ein Maß für die Robustheit der betrachteten Paretopunkte.

Ein selbstoptimierendes System kann auf Basis robuster Paretomengen auch bei abweichenden Anregungssituationen näherungsweise optimal betrieben werden. Ein Wechsel zwischen verschiedenen Paretomengen zur Berücksichtigung unterschiedlicher Anregungssituationen kann hier entfallen.

Die Untersuchung der Robustheit von Paretomengen verläuft in zwei Schritten (siehe Abb. 9.7). Im ersten Schritt werden die tatsächlichen Paretomengen für verschiedene Anregungssituationen bestimmt. Aus den resultierenden Urbildmengen wird eine Referenzmenge gewählt. Im zweiten Berechnungsschritt werden Bildmengen für diese Referenzmenge durch Simulationen des Modells berechnet. Dabei wird die gleiche Menge an Anregungssituationen wie im Schritt eins verwendet. Die Robustheit der Referenz-Paretomenge lässt sich durch Vergleich dieser Bildmengen mit den tatsächlichen Paretofronten bestimmen.

Im Diagramm 9.8 ist die Robustheit des Viertelfahrzeugmodells aus 9.1.2 bei verschiedenen Einstellungen des Anregungsfilters dargestellt. Als Referenz wurde hier die Paretomenge bei f_{z_0} = 1,0 Hz gewählt. Die Symbole zeigen die Zielgrößenwerte, die sich bei einer Simulation mit abweichenden Eckfrequenzen ergeben. Die grauen Kurven stellen im

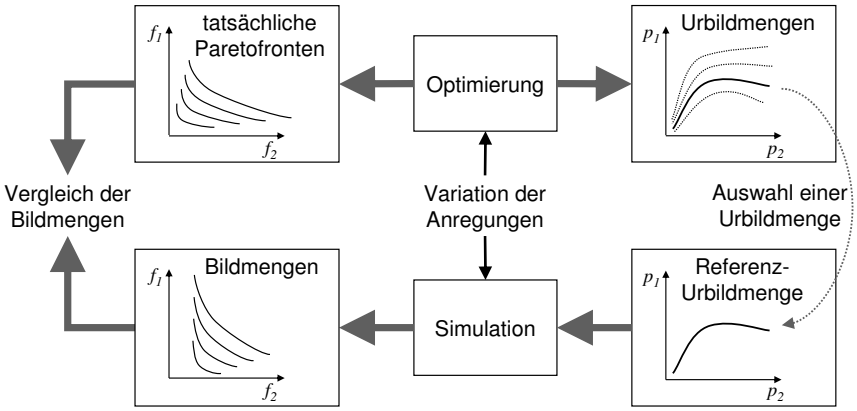


Abbildung 9.7: Vorgehensweise bei der Bestimmung der Robustheit einer Paretomenge gegenüber veränderlichen Anregungen.

Vergleich die Paretofronten der jeweiligen Anregung dar. Die Abweichungen zwischen Paretofronten und den Zielgrößenwerten der Referenzparameter sind sehr gering; die Paretomengen verhalten sich demnach äußerst robust gegenüber Variationen der Anregung.

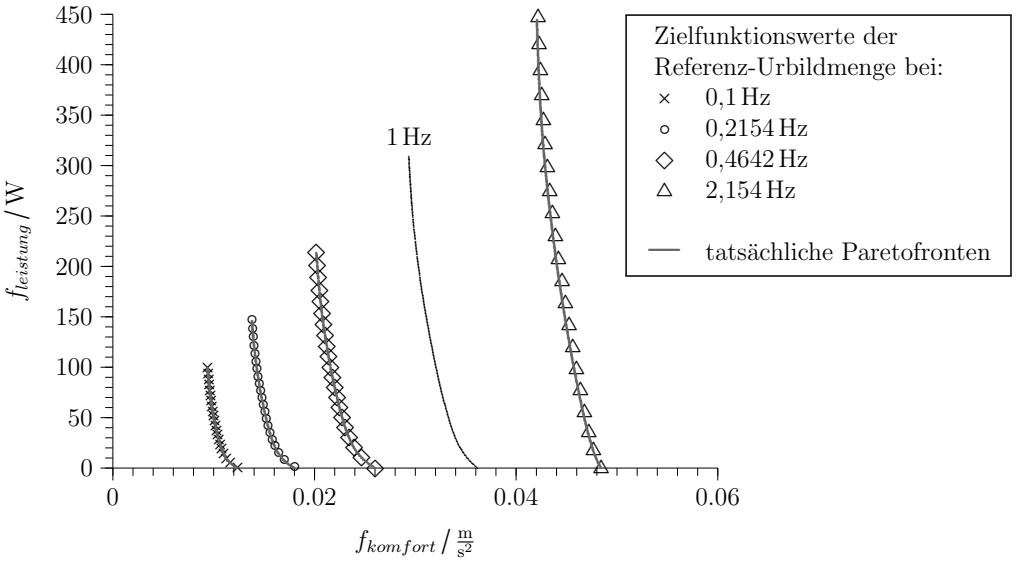


Abbildung 9.8: Vergleich der Zielgrößen der Referenzparetomenge für $f_{z_0} = 1{,}0\,\mathrm{Hz}$ bei unterschiedlichen Anregungsfiltern mit den tatsächlichen Paretofronten

9.1.4 Skalierung von Paretofronten

Die in den vorangegangenen Abschnitten beobachtete Ähnlichkeit der Paretofronten eröffnet bei solchen Systemen die Möglichkeit, die Paretofronten für andere Anregungen abzuschätzen. Dies ist insofern bedeutend, da zum einen Berechnungen lediglich für einzelne Anregungsarten durchgeführt werden müssen. Zum anderen lassen sich aus solchen Gesetzmäßigkeiten Regeln für eine Zielanpassung entwickeln. Diese Abschätzung ist besonders für Variationen der Anregungsamplitude, aber mit gewissen Einschränkungen auch für verschiedene Anregungsspektren möglich.

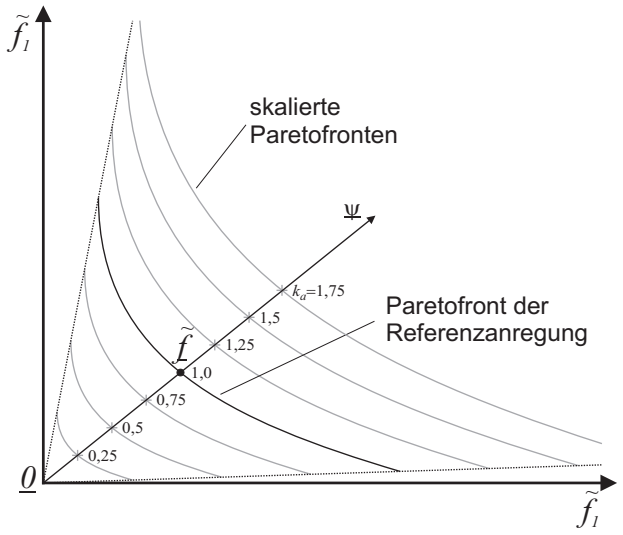


Abbildung 9.9: Skalierung der Paretofront für unterschiedliche Anregungsamplituden.

Durch die Abhängigkeit der Fehlerfläche (9.3) von der Anregungsverstärkung k_a lassen sich bei linearen Systemen die Paretofronten für verschiedene Verstärkungsfaktoren durch einfache Skalierungen abschätzen. Voraussetzung hierfür ist, dass sich die Zielfunktionen proportional bezüglich der Anregungsverstärkung k_a verhalten. Die Fehlerfläche in (9.3) kann über die Beziehung:

$$\tilde{f}_i = |k_a| \cdot \sqrt[n]{\tilde{f}_i} \tag{9.11}$$

in eine zur Anregungsamplitude proportionale Zielfunktion $\tilde{f}_i$ umgerechnet werden.

Die Lage der Paretopunkte im Urbildraum wird hierdurch nicht beeinträchtigt. Die Paretofronten von $\tilde{f}_i$ sind anschließend für unterschiedliche Amplituden ähnlich zueinander und lassen sich im Bildraum durch eine lineare Skalierung ineinander umrechnen. Abbildung 9.9 zeigt beispielhaft die Skalierung einer Paretofront mit den zugehörigen Skalierungsfaktoren.

Alle Punkte im Bildraum, die sich über die verschiedenen Anregungsverstärkungen aus einem Punkt der Urbildmenge ableiten lassen, liegen auf einer Geraden, die durch den Ursprung des Bildraumes geht. Der Abstand der Bildpunkte zum Ursprung ist dabei proportional zur jeweiligen Anregungsverstärkung k_a. Die Punkte auf der Geraden weisen ein konstantes Zielgrößenverhältnis $\underline{\Psi}$ auf:

$$\underline{\Psi} = \frac{\underline{\tilde{f}}}{\left\| \underline{\tilde{f}} \right\|_2} \tag{9.12}$$

Über dieses Zielgrößenverhältnis ist unabhängig von der Anregungsverstärkung eine eindeutige Zuordnung zwischen Punkten der verschiedenen Paretofronten und dem zugehörigen Paretopunkt im Urbildraum möglich.

9.1.5 Anwendungsbeispiel: Niederflur-Federungsprüfstand

Beide Eigenschaften, die Skalierungseigenschaften und die Robustheit wurden an dem hierarchischen Modell des Niederflur-Federungsprüfstand überprüft. Das Diagramm 9.10 zeigt die Ergebnisse dieser Berechnungen für variierende Anregungsverstärkungen. Die

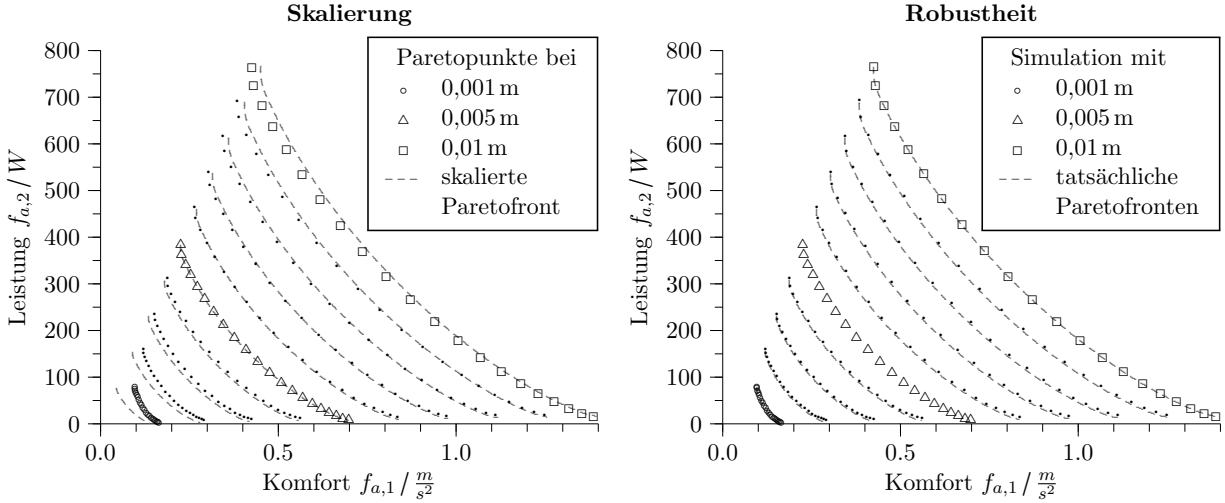


Abbildung 9.10: Eigenschaften der Paretomengen des Federungsprüfstandes bei unterschiedlichen Anregungsverstärkungen: Skalierte Paretofronten (links), Robustheit der Referenz-Urbildmenge mit einer Standardabweichung von $0{,}005\,\mathrm{mm}$ (rechts)

Verstärkungen gelten für die Anregungsrichtungen in z und y-Richtung für das linke und rechte Rad gleichermaßen.

Im linken Diagramm ist der Vergleich zwischen den tatsächlichen Paretopunkten und den skalierten Bildmengen (gestrichelte Linie) dargestellt. Bei niedrigen Verstärkungen können vor allem bei den komfortoptimalen Punkte größere Abweichungen ausgemacht werden. Diese spielen für die spätere Anwendung jedoch nur eine untergeordnete Rolle, da bei diesen Anregungen sowohl die Werte des Komfort als auch des Energiebedarf absolut gesehen vernachlässigbar sind.

Das rechte Diagramm zeigt die Abweichung der Bildpunkte der Referenz-Urbildmenge von den tatsächlichen Paretopunkten der jeweiligen Anregung. Als Referenzmenge wurde die Menge mit einer Standardabweichung des Anregungssignals von $0{,}005\,\mathrm{m}$ gewählt. Die Simulationen mit anderen Standardabweichungen ergeben Bildpunkte, die nahezu auf der tatsächlichen Paretofront der jeweiligen Anregung liegen. Dies lässt auf eine hohe Robustheit bezüglich Amplitudenänderungen schließen. Die größten Abweichungen liegen bei ca. 2%.

9.2 Ziel-Regelung

Im Folgenden wird eine Ziel-Regelung vorgestellt, die auf der Basis einer vorab, für eine feste Anregung berechneten Paretomenge arbeitet. Der Wechsel zwischen verschiedenen Paretomengen zur Berücksichtigung unterschiedlicher Anregungssituation wird hier nicht betrachtet. Aufgabe der Ziel-Regelung ist die Angleichung der Zielerfüllung $\underline{\Phi}_{ist}$ eines sich im Betrieb befindlichen realen Systems an die gewünschten Ziele $\underline{\Phi}_{soll}$, die von der Ziel-Planung oder einer überlagerten Ziel-Regelung vorgegeben werden. In Abbildung 9.11 ist der prinzipielle Aufbau einer Ziel-Regelung, die auf Basis von paretooptimalen Systemeinstellungen arbeitet, dargestellt.

Die Ziel-Regelung besteht – analog zu einem klassischen Regelkreis – aus einer Sensorkomponente, hier der *Ist-Ziel-Erfassung*, dem *Soll-Ist-Vergleich* sowie einem Regelungsal-

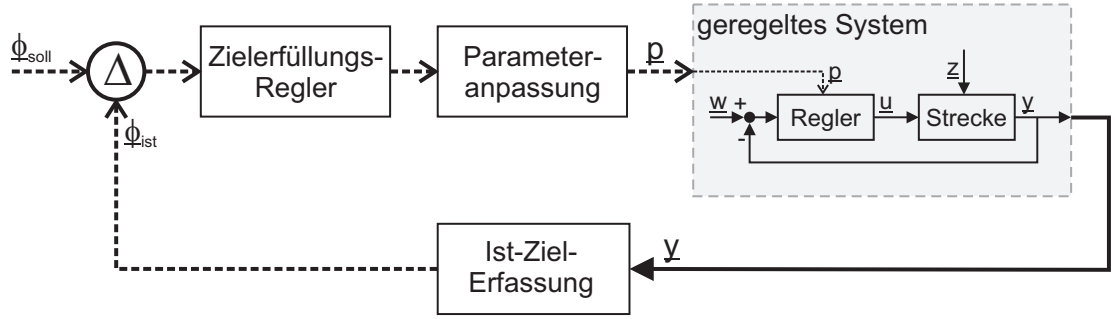


Abbildung 9.11: Aufbau eines Ziel-Reglers zur Anpassung der tatsächlich erreichten Zielerfüllung.

gorithmus, dem *Zielerfüllungsregler*. Anstelle von gemessenen Sensorsignalen arbeitet diese Regelung auf Basis der aktuellen Zielerfüllung $\underline{\Phi}_{ist}$ des geregelten Systems. Die Auswertung des Zielregelkreises geschieht dabei zyklisch in diskreten Zeitschritten. Grundsätzlich können hierbei feste Zeitschritte h, aber auch variable Zeitschritte h_k verwendet werden. Variable Schrittweiten erlauben die einfache Implementierung unter weichen Echtzeitbedingungen bspw. im kognitiven Operator des OCM.

Bestimmung der aktuellen Zielerfüllung: Für die Realisierung der Ziel-Regelung ist es notwendig, den aktuellen Grad der Zielerfüllung im Betrieb zu erfassen. Dies geschieht in der *Ist-Ziel-Erfassung* durch „mitrechnen“ der Bewertungskriterien anhand gemessener oder geschätzter Systemgrößen. Die Erfassung dieser Informationen ist dem ersten Schritt der Selbstoptimierung, der *Analyse der Ist-Situation*, zugeordnet.

Für die Definition der Bewertungskriterien muss beachtet werden, dass diese unter realen Bedingungen ausgewertet werden müssen. Zudem stimmen die im Betrieb erfassten Zielgrößen nicht mit den in der Optimierung verwendeten Zielfunktionswerten überein (siehe Seite 139). Es muss daher zwischen der realen Zielerfüllung $\underline{\Phi}_{ist}$ und den virtuellen Optimierungszielen $\underline{f}$ differenziert werden.

Soll-Ist-Vergleich: Aufgabe des *Soll-Ist-Vergleichs* ist es, einen Vergleich zwischen gemessener Zielerfüllung $\underline{\Phi}_{ist}$ und gewünschten Zielen $\underline{\Phi}_{soll}$ durchzuführen. Das Ziel der Regelung besteht in der Angleichung von $\underline{\Phi}_{ist}$ an $\underline{\Phi}_{soll}$. Da die tatsächlichen Paretomengen der aktuellen Betriebssituation zur Bestimmung der Zielvorgabe $\underline{\Phi}_{soll}$ nicht zur Verfügung stehen, kann eine exakte Übereinstimmung im Allgemeinen nicht erreicht werden. Entweder stellt $\underline{\Phi}_{soll}$ eine utopische Forderung dar oder die aktuelle Situation ist für das System günstiger als angenommen, z. B. wenn nur geringe Störungen vorliegen, so dass eine wesentlich bessere Zielerfüllung $\underline{\Phi}_{ist}$ realisierbar ist. Die Aufgabe der Regelung kann daher nur sein, die Abweichung von $\underline{\Phi}_{ist}$ und $\underline{\Phi}_{soll}$ bezüglich einer Relativgröße, z. B. des Zielgrößenverhältnisses $\underline{\Psi}$ aus Gl. (9.12), zu minimieren.

Das Ergebnis der Vergleichsfunktion wird als Differenz sinnvollerweise in den Optimierungszielen $\underline{f}$ an den *Zielerfüllungsregler* übermittelt. Im *Soll-Ist-Vergleich* muss daher eine Abbildung der Zielerfüllungen $\underline{\Phi}_{ist}$ und $\underline{\Phi}_{soll}$ auf die Optimierungsziele $\underline{f}$ vorgenommen werden. Diese Umrechnung geschieht anhand zuvor berechneter Paretomengen. Aus diesem Grund ist der *Soll-Ist-Vergleich* dem zweiten Schritt der Selbstoptimierung, der *Bestimmung der Systemziele* zuzuordnen.

Zielerfüllungsregler: Die Anpassung des Zielsystems findet im *Zielerfüllungsregler* statt. Auf Basis der Soll-Ist-Differenz ermittelt der Zielerfüllungsregler eine Zielkorrektur, welche den Abstand zwischen gemessener und gewünschter Zielerfüllung verringert. Die Berechnung der Zielkorrektur ist dem zweiten Schritt der Selbstoptimierung, der *Bestimmung der Systemziele*, zugeordnet.

Parameteranpassung: In der *Parameteranpassung* wird der gewünschte Paretopunkt im unterlagerten Regler eingestellt. Die neuen Systemparameter $\underline{p}$ können recht einfach aus der Urbildmenge ausgelesen werden.

Der *Soll-Ist-Vergleich* definiert die Art und Weise, wie die Differenz bzw. der Abstand zwischen $\underline{\Phi}_{ist}$ und $\underline{\Phi}_{soll}$ ermittelt wird. Er besitzt eine große Bedeutung für den gesamten Regelkreis, da hierüber das tatsächliche Regelungsziel des Zielerfüllungsreglers vorgegeben wird. Grundsätzlich sind zahlreiche verschiedene Definitionen für diesen Abstand möglich. Häufig werden in der Mathematik Normen für solche Problemstellungen eingesetzt. In den nachfolgenden Abschnitten 9.2.1 und 9.2.2 werden die folgenden beiden Varianten beschrieben:

- Regelung auf Zielgrößenrelationen
- Regelung auf absolute Zielgrößenwerte

Die erste Variante zeichnet sich durch eine einfache Formulierung des *Soll-Ist-Vergleichs* und durch ein sehr stabiles Regelungsverhalten aus. Die zweite Variante ist in Hinblick auf Anwendungen besonders nützlich.

Voraussetzungen

Diese Regelungen nutzen die im Abschnitt 9.1.4 festgestellten Skalierungseigenschaften von Paretofronten, um die Angleichung von Φ_{ist} an $\underline{\Phi}_{soll}$ zu realisieren. Bei der Umsetzung eines solchen Zielerfüllungs-Regelkreises gilt es einige Voraussetzungen zu beachten:

- Es wird im Folgenden davon ausgegangen, dass die Paretomengen im Bild- und Urbildraum zusammenhängend sind. Mechanismen um Unstetigkeiten zu behandeln, werden hier nicht betrachtet.
- Die Zielfunktionen $\underline{\Phi}(\underline{p})$ des realen Systems müssen sich bei Parameteränderungen ähnlich zu den Zielfunktionen $\underline{f}(\underline{p})$ der modellbasierten Optimierung verhalten. Für die Ziel-Regelung wird die Veränderung eines Ziels Φ_i auf eine Änderung der Zielfunktion f_i abgebildet. Wird durch Auswahl eines neuen Paretopunktes die Zielgröße f_i der Optimierung verbessert, so muss dies in einer Verbesserung des Systemverhaltens bzgl. des Ziels Φ_i resultieren. D. h. bei der Auswahl eines neuen Parametersatzes $\underline{p}_{k+1}$ mit der Eigenschaft
$$f_i(\underline{p}_{k+1}) < f_i(\underline{p}_k) \tag{9.13}$$
muss, unter der Annahme einer unveränderlichen Anregungsform $\underline{z}_k$, auch die Beziehung
$$\Phi_i(\underline{p}_{k+1}, \underline{z}_k) < \Phi_i(\underline{p}_k, \underline{z}_k) \tag{9.14}$$
gelten. In der Regel wird dies durch die Verwendung identischer Bewertungsfunktionen bei der Optimierung sowie im Betrieb des Systems erreicht.

- In vielen Fällen ist es günstig, bereits bei der Optimierung möglichst realitätsnahe Anregungen einzusetzen, da hierdurch eine bessere Übereinstimmung der Optimierungsziele $\underline{f}$ und der Zielerfüllung $\underline{\Phi}$ erreicht wird. Mechatronische Systeme unterliegen oft Anregungen, die einen stochastischen Charakter aufweisen. Im Gegensatz hierzu können für eine modellbasierte Optimierung synthetische Anregungen, wie z. B. Sprung- oder Rampenantworten, verwendet werden, die in der Realität so nicht vorkommen.

- Bei der Definition der Bewertungskriterien sollte auf Funktionen zurückgegriffen werden, die gute Skalierungseigenschaften bzgl. der Anregungsamplitude besitzen. Ebenso sollte auf Nebenbedingungen verzichtet werden, die von den Anregungsverhältnissen abhängen, da diese zu einer „unnatürlichen“, anregungsabhängigen Verzerrung der Paretomengen führen können. Vielmehr wird an dieser Stelle vorgeschlagen, Nebenbedingungen nachträglich durch Selektion geeigneter Paretopunkte zu erfüllen.

- Zielfunktionen, wie bspw. die zeitgewichteten Fehlerflächen (siehe Tabelle 4.1), die ein eindeutiges Startereignis voraussetzen und verschiedene Zeitpunkte unterschiedlich stark bewerten, sind als Bewertungskriterien im Betrieb nicht geeignet.

  Integrale Zielkriterien ohne Zeitgewichtung wie Betrags- oder quadratische Fehlerflächen eignen sich dagegen gut. Aufgrund ihres mittelnden Charakters werden Messrauschen und Ausreißer im gemessenen Signal abgeschwächt. Durch eine variable Vorgabe der Integralgrenzen können sie leicht an veränderliche Schrittweiten h_k angepasst werden:

$$\Phi_{i,k} = \left(\frac{1}{h_k} \int_{t_k}^{t_k+h_k} |e_i(t)|^n \, dt \right)^{\frac{1}{n}} \tag{9.15}$$

- Im Sinne einer Kaskadenregelung muss die Dynamik des äußeren Zielregelkreises geringer sein, als die Dynamik des klassischen, inneren Regelkreises (vgl. [Föl94]). Die Dynamik des Zielregelkreises kann im Wesentlichen durch die Einstellung des Zielreglers und durch die Größe der Zeitschritte h_k beeinflusst werden. Durch die Wahl eines genügend großen h_k, werden die beiden Regelkreise zeitlich voneinander entkoppelt; das innere geregelte System erhält hinreichend Zeit zum Einschwingen auf neue Parameter $\underline{p}_{k+1}$. Im Gegensatz hierzu führt ein „schwach“ eingestellter Zielregler zu eher geringen Änderungen an den Parametern, so dass Einschwingeffekte mit sinkender Dynamik geringer ausfallen.

9.2.1 Regelung auf Zielrelationen

Die Regelung auf eine Zielrelation gründet sich auf der Anforderung, einen bestimmten Kompromiss zwischen verschiedenen Zielen im Betrieb unabhängig von der aktuellen Anregung im System einzustellen. Ein konkreter Kompromiss lässt sich über das Verhältnis $\underline{\Psi}$ aus Gleichung (9.12) definieren als:

$$\underline{\Psi}_{ist} = \frac{\underline{\Phi}_{ist}}{\|\underline{\Phi}_{ist}\|_2} \tag{9.16}$$

Diese Zielrelation dient der hier vorgestellten Regelung als Regelgröße. Bei Problemen mit zwei Zielgrößen kann alternativ auch der Quotient:

$$\Xi_{ist} = \frac{\Phi_{1,ist}}{\Phi_{2,ist}} \tag{9.17}$$

genutzt werden. Dieser beschreibt den gleichen Vektor im Bildraum wie das Verhältnis (9.16). Dieser Vektor ist in Abbildung 9.12 als Gerade eingezeichnet, da lediglich die Richtungsinformation von Interesse ist.

Einer Regelung auf diese Relation kommt zugute, dass bei robusten Paretofronten das Verhältnis $\underline{\Psi}_{ist}$ relativ unempfindlich auf Änderungen der Anregungsverhältnisse reagiert. Im Falle linearer Systeme reagieren Zähler und Nenner von (9.16) bzw. (9.17) vergleichbar auf Änderungen der Störung. Da der Quotient so unabhängig von der Anregungssituation in etwa dem eingestellten Zielverhältnis $\underline{\Psi}_{stell}$ entspricht, gibt es für den Regler im Grunde nicht viel zu tun. Die Aufgabe dieses Regelungsansatzes ist daher Abweichungen vom idealen Verhalten, bedingt durch Unterschiede zwischen Modell und realem System, nichtlinearem Streckenverhalten und Änderungen in der Anregungscharakteristik, zu kompensieren.

Der Berechnungsansatz zur Bestimmung eines neuen Paretopunktes setzt im Bildraum an. Abbildung 9.12 zeigt diesen Ansatz für ein Problem mit zwei Zielfunktionen. Gegeben sind die Zielvorgaben $\underline{\Phi}_{soll}$ und die gemessene Zielerfüllung $\underline{\Phi}_{ist}$. Durch Skalierung

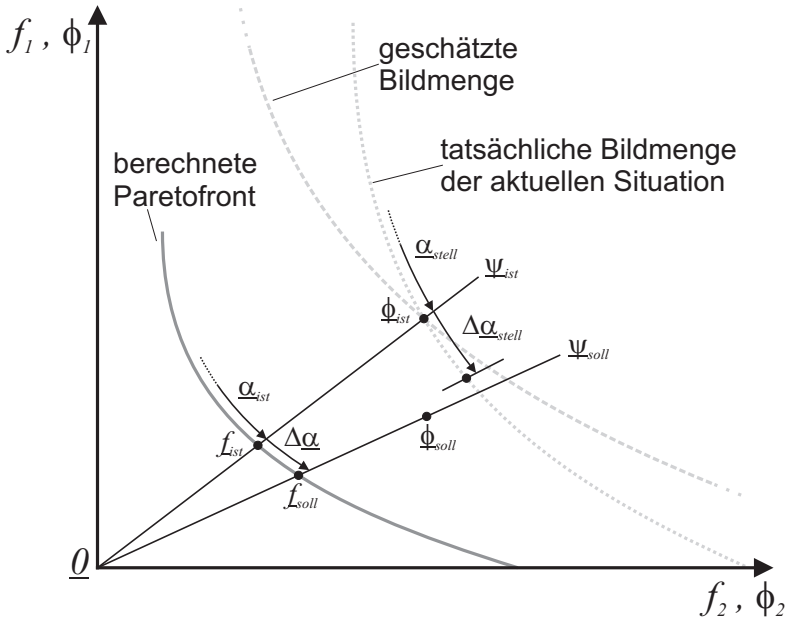


Abbildung 9.12: Bestimmung der Zielkorrektur im Bildraum

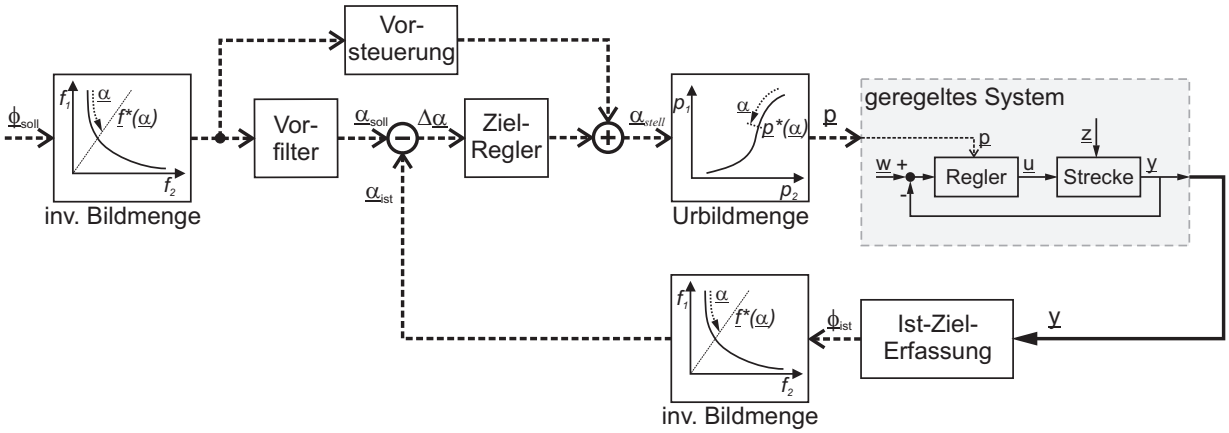


Abbildung 9.13: Regelkreis für die Zielrelation

der berechneten Paretofront, derart dass sie durch den Punkt $\underline{\Phi}_{ist}$ geht, erhält man eine Schätzung der Bildmenge der aktuellen Betriebssituation. Diese Schätzung stimmt i. A. nicht mit der tatsächlichen Paretofront des realen Systems überein. Es wird jedoch die Annahme getroffen, dass sie in der Nähe des Optimums liegt. Die tatsächliche Bildmenge der aktuellen Situation enthält die Bildpunkte, die durch Einstellen von Urbildpunkten der berechneten Paretomenge erreicht werden können.

Die Idee des Ansatzes besteht darin, die gemessene und gewünschte Zielerfüllung über die beiden Zielrelationen $\underline{\Psi}_{ist}$ und $\underline{\Psi}_{soll}$ auf die berechnete Paretomenge abzubilden. Hierdurch erhält man $\underline{f}_{ist}$ und $\underline{f}_{soll}$. Im Allgemeinen liegt $\underline{\Phi}_{soll}$ abseits der tatsächlichen Bildmenge von $\underline{\Phi}_{ist}$. Ein direkter Vergleich der beiden Zielerfüllungen gestaltet sich daher schwierig. Durch die Abbildung auf die berechnete Paretomenge wird dieser ermöglicht.

Die Regelung arbeitet intern auf Basis einer Referenzierungsgröße $\underline{\alpha}$. Mit dieser Referenzierungsgröße lassen sich einzelne Punkte der Paretomenge adressieren. Den beiden Zielrelationen $\underline{\Psi}_{ist}$ und $\underline{\Psi}_{soll}$ sind die Referenzierungsgrößen $\underline{\alpha}_{ist}$ und $\underline{\alpha}_{soll}$ zugeordnet. Über deren Differenz $\Delta\underline{\alpha}$ lässt sich in einem Regelungsgesetz die Korrektur $\Delta\underline{\alpha}_{stell}$ berechnen und hierüber ein neuer Paretopunkt $\underline{\alpha}_{stell}$ einstellen.

Im Diagramm 9.13 ist die Struktur dieses Regelungskreises dargestellt. Die Abbildung der gemessenen und gewünschten Zielerfüllung auf $\underline{\alpha}_{ist}$ und $\underline{\alpha}_{soll}$ wird in den Blöcken *inverse Bildmenge*[1] vorgenommen. Die Referenzierung von $\underline{\alpha}_{stell}$ auf den neuen Parametersatz $\underline{p}$ findet im Block *Urbildmenge* statt.

Zielregler und Vorsteuerung: Die Aufgaben des Zielreglers und der Vorsteuerung verhalten sich analog zu denen klassischer Regelungen. Die Vorsteuerung ist für das Führungsverhalten verantwortlich. Die gewünschte Zielrelation kann über die Urbildmenge direkt auf das System aufgeschaltet werden. Der Zielregler kompensiert die Abweichungen zwischen dem gewünschten und dem tatsächlich erreichten Zielverhältnis. Die zu kompensierenden Effekte resultieren im Wesentlichen aus Abweichungen zwischen Optimierungsmodell und realem System- und Anregungsverhalten sowie aus idealisierten Annahmen. Da diese Fehler einen beständigen Einfluss auf die Zielerfüllung des Systems ausüben, ist

[1] Mit $\underline{\alpha}$ werden Punkte der Paretomenge referenziert; somit wird die Abbildung $\underline{\alpha} \rightarrow \underline{f}$ im Zusammenhang mit der Bildmenge als die direkte Abbildung angesehen. Die inverse Bildmenge beschreibt die Abbildung $\underline{f} \rightarrow \underline{\alpha}$ bzw. hier $\underline{\Phi} \rightarrow \underline{\alpha}$.

mit stationären Abweichungen zu rechnen. Der Einsatz eines Reglers mit integrierendem Anteil ist daher sinnvoll.

Referenzierungsgröße: Für ein gleichmäßiges Regelungsverhalten, besonders in den Extrembereichen der Paretomenge, ist es wichtig, dass $\underline{\alpha}$ eine möglichst homogene Verteilung der Punkte auf der Paretofront beschreibt. Dieser Sachverhalt ist vergleichbar zu der in Kapitel 3.5 beschriebenen Forderung nach einer möglichst gleichmäßigen Verteilung der berechneten Punkte im Bildraum.

Die NBI-Methode aus Kapitel 3.5.1 liefert einen guten Ansatz zur Definition von $\underline{\alpha}$. Mit dem Simplex $\underline{\boldsymbol{\Theta}}$ aus (3.40) erhält man die Beziehung:

$$\underline{\boldsymbol{\Theta}}\,\underline{\beta} = \gamma\,\underline{\Psi} \tag{9.18}$$

$$\text{mit} \quad \sum_i \beta_i = 1\,,\ \beta_i \geq 0 \tag{9.19}$$

Die Länge γ beschreibt den Abstand vom Ursprung zum Punkt $\underline{\boldsymbol{\Theta}}\,\underline{\beta}$ auf dem Simplex und $\underline{\Psi}$ gibt die Zielrelation dieses Punktes an.

Der Vektor $\underline{\beta}$ wird in (9.18) für die Referenzierung des Paretopunktes genutzt. Wegen der Beziehung 9.19 ist der Vektor $\underline{\beta}$ um ein Element überbestimmt. Im Regelkreis wird daher anstelle von $\underline{\beta} \in \mathbb{R}^m$ der reduzierte Vektor $\underline{\alpha} \in \mathbb{R}^{m-1}$ eingesetzt, mit:

$$\underline{\alpha} = \left[\beta_1, \ldots, \beta_{m-1}\right]^{\mathrm{T}} \quad \text{und} \quad 0 \leq \left\|\underline{\alpha}\right\|_1 \leq 1 \tag{9.20}$$

Das Element β_m wird in der Regelung implizit über die Betragssummennorm $\left\|\underline{\alpha}\right\|_1$ mitgeführt.

In der *inversen Bildmenge* wird die Referenzierungsgröße $\underline{\alpha}$ aus einer gegebenen Zielerfüllung $\underline{\Phi}$ berechnet. Die Überführung von $\underline{\Phi}$ nach $\underline{\alpha}$ erfolgt über die Zielrelation $\underline{\Psi}$. Einem $\underline{\beta}$ (und somit einem $\underline{\alpha}$) lässt sich mit $\gamma = \left\|\underline{\boldsymbol{\Theta}}\,\underline{\beta}\right\|_2$ immer eine Zielrelation $\underline{\Psi}$ zuordnen. Die Transformation der Zielerfüllung $\underline{\Phi}$ in die Referenzierungsgröße $\underline{\alpha}$ erhält man durch Umstellen von (9.18) nach $\underline{\beta}$:

$$\underline{\beta} = \gamma\,\underline{\boldsymbol{\Theta}}^{-1}\underline{\Psi} = \gamma\,\underline{\boldsymbol{\Theta}}^{-1}\frac{\underline{\Phi}}{\left\|\underline{\Phi}\right\|_2} \tag{9.21}$$

Durch Berücksichtigung der Bedingung (9.19) gelangt man zur Berechnungsvorschrift:

$$\underline{\beta} = \frac{\underline{\boldsymbol{\Theta}}^{-1}\,\underline{\Phi}}{\left\|\underline{\boldsymbol{\Theta}}^{-1}\,\underline{\Phi}\right\|_1} \tag{9.22}$$

Über die Beziehung (9.20) lässt sich aus (9.22) das gesuchte $\underline{\alpha}$ ermitteln.

Dynamik des Regelkreises: Für die Auslegung des Reglers und der Vorsteuerung kann auf die Methoden der digitalen Regelungstechnik zurückgegriffen werden. Das dynamische Verhalten des äußeren Regelkreises kann in erster Näherung über den Modellierungsansatz in Diagramm 9.14 abgebildet werden. Das Diagramm zeigt einen zeitdiskreten Regelkreis mit einem Regler $G_R(z)$, einer Vorsteuerung ${G_S}^{-1}(z)$ und einem Vorfilter $G_V(z) = \frac{1}{z}$.

Bei diesem Modellierungsansatz wurden die folgenden Annahmen getroffen:

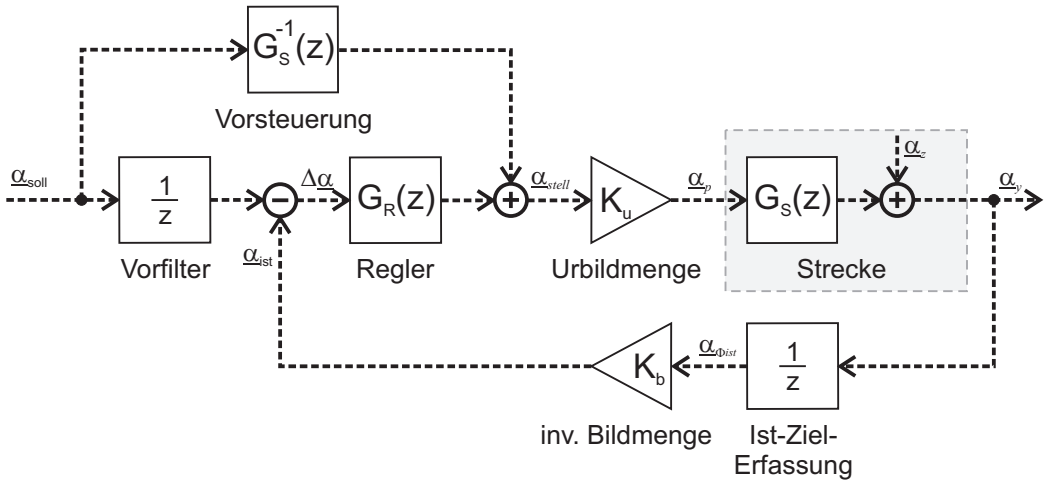


Abbildung 9.14: Ersatzmodell des Regelkreises auf ein konstantes Zielverhältnis

- Der Signalpfad von $\underline{\alpha}_{stell}$ nach $\underline{\alpha}_{ist}$ wird stark abstrahiert. Der Einfluss der Paretomengen und die Dynamik des unterlagerten geregelten Systems wird auf das Übertragungsverhalten reduziert.

- Die Übertragungspfade der einzelnen Komponenten des Vektors $\underline{\alpha}$ werden unabhängig voneinander betrachtet. Diese Annahme stützt sich auf die entkoppelnde Wirkung der Transformationen in den Blöcken *Urbildmenge* und *inverse Bildmenge*.

- Im idealen Fall gleichen sich die Transformation von $\underline{\alpha}_{stell}$ über die *Urbildmenge* auf den inneren Regelkreis und die Rücktransformation nach $\underline{\alpha}_{ist}$ über die *inverse Bildmenge* gegenseitig weitgehend aus. Für diesen Pfad kann in erster Näherung eine Gesamtverstärkung von 1 mit $K_u = 1$ und $K_b = 1$ angesetzt werden. Im ungestörten Fall mit $\underline{\alpha}_z = 0$ strebt die „Messgröße“ $\underline{\alpha}_{ist}$ für $t \to \infty$ gegen die Vorgabe $\underline{\alpha}_{stell}$.

- Die Reaktion des unterlagerten geregelten Systems auf Änderungen von $\underline{\alpha}_{stell}$ wird durch die Übertragungsfunktion $G_S(z)$ abgebildet. Hierdurch werden Überblend- und Einschwingeffekte berücksichtigt, die beim Aufschalten eines neuen Parametersatzes auftreten können. In vielen Fällen reagiert das System prompt auf Parameteränderungen. In diesen Fällen kann $G_S(z) = 1$ gesetzt werden.

- Der Einfluss der jeweiligen Betriebssituation auf das System fließt über die Störung $\underline{\alpha}_z$ ein.

- Aufgrund des Integrals in (9.15) werden die Zielerfüllungen $\underline{\Phi}_{ist}$ in der *Ist-Ziel-Erfassung* zeitversetzt ausgegeben. Dies wird durch ein Halteglied $\frac{1}{z}$ abgebildet.

Die Vorsteuerung ist so ausgelegt, dass sie die Dynamik der geregelten Strecke kompensiert. Dies wird über die inverse Übertragungsfunktion der Strecke ${G_S}^{-1}(z)$ erreicht. Wegen des Halteglieds der *Ist-Ziel-Erfassung* wird eine Reaktion der Strecke jedoch erst um einen Takt verzögert erwartet. Durch den Vorfilter wird das Soll-Signal zeitlich auf diesen Erwartungswert „synchronisiert“. Bei ungestörter Strecke gilt für die Differenz $\Delta\underline{\alpha} = 0$, und der Regler beeinflusst das Ergebnis der Vorsteuerung nicht. Der Übertragungspfad von $\underline{\alpha}_{soll}$ nach $\underline{\alpha}_{ist}$ reduziert sich durch diese Maßnahme auf das Halteglied der *Ist-Ziel-Erfassung*.

Das Führungsverhalten des Regelkreises ergibt sich zu:

$$\frac{\underline{\alpha}_y(z)}{\underline{\alpha}_{soll}(z)} = K_u \tag{9.23}$$

Das Störverhalten kann mit dem Übertragungsverhalten von der Störung $\underline{\alpha}_z$ bis zur Systemantwort $\underline{\alpha}_y$ angegeben werden:

$$\frac{\underline{\alpha}_y(z)}{\underline{\alpha}_z(z)} = \left(1 + \frac{1}{z} K_b G_R(z) K_u G_S(z)\right)^{-1} \tag{9.24}$$

9.2.2 Regelung auf absolute Zielvorgaben

Diese Art der Regelung hat zum Ziel, den Wert einer vorgegebenen Ziels Φ_{soll} im System einzuregeln. Durch eine solche Regelung lässt sich der Erfüllungsgrad eines bestimmten Ziels auf einen konkreten Wert einstellen. Dementsprechend werden die übrigen Zielerfüllungen in Abhängigkeit von der aktuellen Betriebssituation und im Sinne eines paretooptimalen Systems vergrößert oder verkleinert. Eine solche Vorgabe kann auf vielfältige Weise eingesetzt werden. Beispielsweise kann einem Aggregat von einem überlagerten Energiemanagement ein bestimmter Energieverbrauch vorgegeben werden. Dieser Energieverbrauch wird vom Aggregat eingehalten. Die übrigen Ziele werden unter Berücksichtigung dieser Beschränkung bestmöglich erreicht.

Wie der „Regler auf Zielrelationen“ setzt dieser Regler im Bildraum an. Die beiden Regelkreise unterscheiden sich im Wesentlichen in der Berechnung der Sollgröße $\underline{\alpha}_{soll}$. Der Zielregler und die Transformation auf $\underline{p}$ über die Urbildmenge arbeitet intern ebenfalls auf Basis der Referenzierungsgröße $\underline{\alpha}$. Es kann hier auf dieselbe Berechnungsvorschrift zurückgegriffen werden.

Abbildung 9.15 zeigt eine grafische Interpretation des Verfahrens. Bekannt sind die Vor-

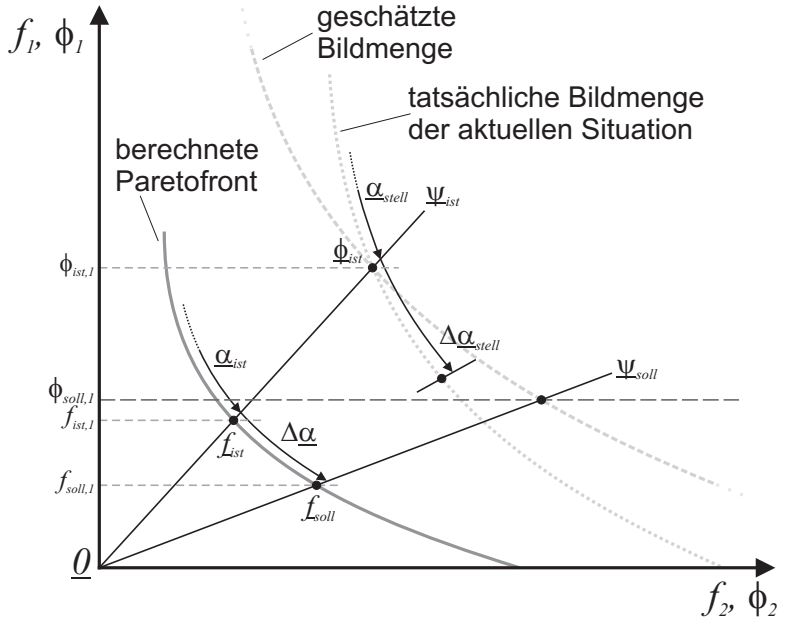


Abbildung 9.15: Bestimmung der Zielkorrektur im Bildraum

gabe $\Phi_{soll,1}$ sowie die reale Zielerfüllung $\underline{\Phi}_{ist}$. Es wird hier davon ausgegangen, dass $\Phi_{soll,1}$ den Sollwert des Regelkreises bildet. Über $\underline{\Phi}_{ist}$ lässt sich die Referenzierungsgröße $\underline{\alpha}_{ist}$ und der zugehörige Paretopunkt $\underline{f}_{ist}$ bestimmen.

Das Verfahren nutzt eine Schätzung der Bildmenge der aktuellen Situation zur Bestimmung der gewünschten Referenzierungsgröße $\underline{\alpha}_{soll}$. Diese Schätzung erhält man durch Skalieren der berechneten Paretofront. Es kann daher die Ähnlichkeitsbeziehung:

$$\frac{\Phi_{soll,1}}{f_{soll,1}} = \frac{\Phi_{ist,1}}{f_{ist,1}} = \gamma_{ist} \tag{9.25}$$

ausgenutzt werden, wobei γ_{ist} den Skalierungsfaktor zwischen der berechneten Paretofront und dem tatsächlichen erreichten Bildpunkt bezeichnet. Durch Umstellen von (9.25) erhält man die Berechnungsvorschrift:

$$f_{soll,1} = \frac{\Phi_{soll,1}}{\gamma_{ist}} \tag{9.26}$$

zur Bestimmung von $f_{soll,1}$.

Bemerkenswert an diesem Regelkreis ist die Aufteilung in die Regelgrößen $\underline{\alpha}_{ist}$ und γ_{ist}. Beide Größen können aus Messgrößen ermittelt werden. Die Größe $\underline{\alpha}_{ist}$ gibt das Verhältnis der Zielgrößen an. Sie kann über eine Verschiebung des Paretopunktes beeinflusst werden. Demgegenüber ergibt sich γ_{ist} im Falle robuster Paretofronten nur aus der Höhe der am System angreifenden Störung; γ_{ist} kann nicht beeinflusst werden.

Der Zielregler sorgt für das Einstellen der geforderten Zielerfüllung $\underline{\alpha}_{soll}$ und weist bezüglich $\underline{\alpha}_{ist}$ das gleiche Störverhalten wie der im vorangegangene Abschnitt beschriebene Zielrelationsregler auf. Über γ_{ist} wird die Sollvorgabe $\underline{\alpha}_{soll}$ ermittelt.

Auf eine strikte Aufgabenteilung zwischen Vorsteuerung und Regler, um das gewünschten Führungs- und Störverhalten einzustellen, wird hier verzichtet, da ein wesentlicher Teil der Störung über γ_{ist} bereits in $\underline{\alpha}_{soll}$ einfließt. Die „Nervosität“ des Reglers und somit auch die Empfindlichkeit auf Änderungen von γ_{ist} kann bei dem in Abbildung 9.16 dargestellten Regelkreis über die Einstellung des Zielreglers beeinflusst werden.

Bei einem Problem mit zwei Zielgrößen kann mit $f_{soll,1}$ über die Bildmenge direkt das gesuchte, skalare α_{soll} ermittelt werden. Bei mehr als zwei Zielen existieren jedoch weitere Freiheitsgrade. Die Berechnungsvorschrift (9.26) kann dementsprechend auf weitere Ziele solange angewendet werden, bis $\underline{\alpha}_{soll}$ eindeutig bestimmt ist oder ein Sollwert $\Phi_{ist,i}$ nicht mehr erreicht werden kann. Tritt letzteres auf, so wurde der Rand der Paretofront erreicht.

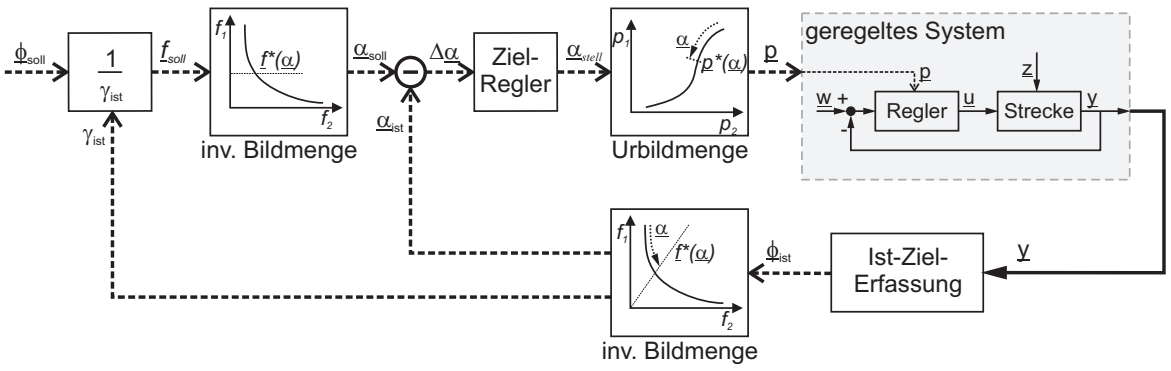


Abbildung 9.16: Regelkreis zum Einregeln absoluter Zielvorgaben

Die Ränder der Paretofront stellen Stellwertbegrenzungen dar, die dabei über geeignete Mechanismen, vergleichbar den *Anti-Windup*-Maßnahmen, zu berücksichtigen sind.

9.3 Ziel-Regelung des Niederflur-Federungsprüfstandes

Die Funktionalität der beiden Ziel-Regelkreise wurde anhand des Niederflur-Feder-Neige-Prüfstandes überprüft. Die Überprüfung erfolgt am nichtlinearen monolithischen Modell des Prüfstandes. In diesem Modell sind insbesondere die Lenkerkinematik und die hydraulischen Ventile und Zylinder des Systems detailliert abgebildet. Die Paretomengen werden von der MOBU-Optimierung des hierarchischen Modells in Kapitel 8.4 übernommen. Die Güte des Systems wird über die beiden Zielgrößen, dem Komfort und der hierfür notwendige Leistungsaufnahme, beschrieben.

$$\underline{f} = [f_{komfort}, f_{leistung}]^{\mathrm{T}} \quad \text{und} \quad \underline{\Phi} = [\Phi_{komfort}, \Phi_{leistung}]^{\mathrm{T}} \tag{9.27}$$

Für die Berechnung der realen Zielerfüllungen kommen die gleichen Bewertungsfunktionen wie bei der Optimierung zum Einsatz. Den Parametervektor $\underline{p}$ bilden die 6 Parameter des Aufbaureglers sowie die Einstellung der unterlagerten Aktorgruppen.

Ein wesentlicher Aspekt dieser Untersuchungen ist das Verhalten der Regelkreise bei unterschiedlichen Anregungscharakteristiken. Zur Beschreibung der Anregungen auf das System wird auf zwei unterschiedliche Anregungsmodelle zurückgegriffen.

1. Das erste Modell beruht auf einer klassischen Vorgehensweise. Die Anregung durch die Schiene in den drei Raumrichtungen z, y, und in der Verdrehung φ werden durch ein bandbegrenztes, gefiltertes Rauschsignal abgebildet. Die Filterung des Rauschsignals geschieht über das PT_1-Glied:

   $$G_{PT_1}(s) = \frac{\sigma\sqrt{2\,T_1}}{T_1\,s + 1} \tag{9.28}$$

   Über die Standardabweichung σ und die Zeitkonstante T_1 kann das Signal variiert werden. Die verwendete Paretomenge basiert auf einer Optimierung mit diesem Anregungsmodell.

2. Der zweite Modellierungsansatz bildet Gleislagefehler durch einen kubischen Spline ab. Der Spline beschreibt die Biegung der Schiene; die Bahnschwellen bilden die Stützpunkte des Splines. Die Anregung erfolgt über eine zufällige Verschiebung der Stützpunkte in den Koordinaten z, y und φ.

   Der Abstand der Bahnschwellen wird im Modell mit 1 m angenommen. In den folgenden Simulationen wird von einer Fahrgeschwindigkeit von $10\,\frac{\mathrm{m}}{\mathrm{s}}$ ausgegangen; die Bahnschwellen werden daher mit einer Frequenz von 10 Hz passiert.

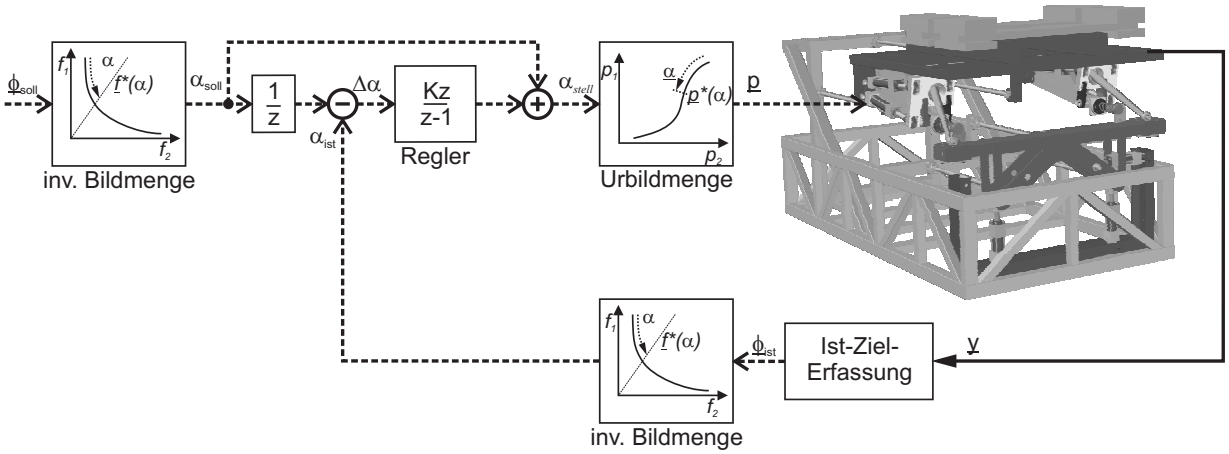


Abbildung 9.17: Regelkreis für das Einhalten eines konstanten Zielverhältnisses beim Feder-Neige-System

9.3.1 Regelung auf Zielrelationen

Für die Untersuchung des Zielrelationsreglers wurde der Regelkreis in Abbildung 9.17 umgesetzt.

Die folgenden Einstellungen wurden am Regelkreis vorgenommen:

- Eine Aktualisierung der Reglerparameter findet alle $h = 2{,}0\,\mathrm{s}$ statt.
- Die Einstelldynamik auf neue Parametersätze kann bei diesem System vernachlässigt werden. Hieraus folgt $G_S(z) = 1$.
- Es wird ein I-Regler mit der Übertragungsfunktion $G_R(z) = \frac{Kz}{z-1}$ verwendet. Der Verstärkungsfaktor wird auf $K = 0{,}7$ gesetzt.
- Die Referenzierungsgrößen beziehen sich auf den Komfort. Es wird demnach auf $\alpha = \beta_{komfort}$ geregelt.

Das Störübertragungsverhalten des Regelkreises aus Gl. (9.24) lässt sich unter der Annahme $K_u = 1$ und $K_b = 1$ mit

$$\frac{\alpha_y(z)}{\alpha_z(z)} = \frac{1}{1 + \frac{1}{z}\,\frac{Kz}{z-1}} = \frac{z-1}{z+K-1} \tag{9.29}$$

angeben. Die Übertragungsfunktion (9.29) besitzt eine reelle Polstelle bei $1 - K$. Stabile Einstellungen für die Verstärkung des I-Reglers liegen demnach zwischen $0 < K < 2$. Hierbei führen Werte über 1 zu einem oszillierenden Verhalten von (9.29); Werte nahe der 1 erreichen ein schnelles, „nervöses“ Reglerverhalten; Einstellungen mit niedrigen Verstärkungen sind dagegen stärker gedämpft, kompensieren Abweichungen in den Zielerfüllungen jedoch auch langsamer.

Führungsverhalten

Das Führungsverhaltens des Regelkreises wird mit Hilfe einer Variation des Soll-Signals $\underline{\alpha}_{soll}$ untersucht. Um möglichst geringe Variationen im Störsignal zu erhalten, werden die Parameter des Anregungsmodells während der Simulation nicht verändert. Bei der Simulation wird auf ein anderes Anregungsmodell als bei der Optimierung zurückgegriffen. Es wird hier der Spline-Ansatz genutzt. Es ist damit Aufgabe des Reglers, diese abweichende Anregungscharakteristik auszugleichen.

Abbildung 9.18 zeigt das Führungsverhalten des Regelkreises. Das Führungssignal α_{soll} wird treppenförmig zwischen den Minimal- und Maximalwerten 0 und 1 verstellt. Das zugehörige $\underline{\Phi}_{soll}$ wird anschließend über die Bildmenge ermittelt, so dass hier $\underline{\Phi}_{soll} = \underline{f}_{soll}$ gilt. Der Regelkreis ist in der Lage die Regeldifferenz zwischen α_{soll} und α_{ist} klein zu halten. Die abweichende Anregungscharakteristik verursacht eine Differenz zwischen Stellsignal α_{stell} und der Systemantwort α_{ist}, die bewirkt, dass das System für $\alpha_{soll} > 0{,}7$ nicht in der Lage ist, die hohe Forderung an den Komfort zu erfüllen. Dies wird anhand des Stellsignals α_{stell} deutlich, welches sich im Zeitraum bis 40 s und ab 220 s an der oberen Grenze bewegt. Dies entspricht der komfortoptimalen Einstellung.

Die Zielrelation wird über den gewünschten Ξ_{soll} und den erreichten Quotienten Ξ_{ist} dargestellt. Die Zielrelation reagiert trotz des starken Rauschens in der realen Zielerfüllung sehr unempfindlich auf die stochastische Anregung. Aufgrund des I-Reglers werden die Sollwerte im Mittel stationär genau eingehalten.

Störverhalten

Das Störverhalten des Regelkreises wird mit Hilfe eines „Sprungs“ in der Anregungscharakteristik simuliert. Zu Beginn der Simulation ist das Spline-Anregungsmodell aktiv. Der Sprung wird durch eine Umschaltung zum Zeitpunkt $t = 22$ s auf das gefilterte Rauschsignal realisiert. Das Führungssignal wird anfangs auf den Wert $\alpha_{soll} = 0{,}5$ gestellt und ist anschließend konstant.

Abbildung 9.19 zeigt die Ergebnisse der Simulation. Der Sprung bei $t = 22$ s führt zu einer Abweichung im Ist-Verlaufs α_{ist} bzw. im Quotienten, die bereits nach 2 Korrekturschritten durch α_{stell} ausgeglichen ist. Gut zu erkennen sind auch die unterschiedlichen Niveaus von α_{stell}, welche bei verschiedenen Anregungsmodellen erforderlich sind, um den Soll-Wert zu halten.

Die Umschaltung des Anregungsmodells führt zur Absenkung von $\Phi_{komfort,ist}$ und $\Phi_{leistung,ist}$. Während die Leistungsaufnahme bei $t = 24$ s um ca. $\frac{1}{3}$ absinkt, wird $\Phi_{komfort,ist}$ um den Faktor 6 kleiner. Dies wird auch in der Änderung des Quotienten Ξ_{ist} ersichtlich.

Insgesamt wird deutlich, dass die Regelgröße, die Zielrelation, gegenüber Störungen sehr robust ist und erst durch die gravierende Änderung der Anregungscharakteristik beeinflusst wird. In diesen Fällen ist der Zielrelationsregler geeignet, die stationäre Abweichung zu kompensieren und das gewünschte Verhältnis zwischen den Zielen wieder herzustellen.

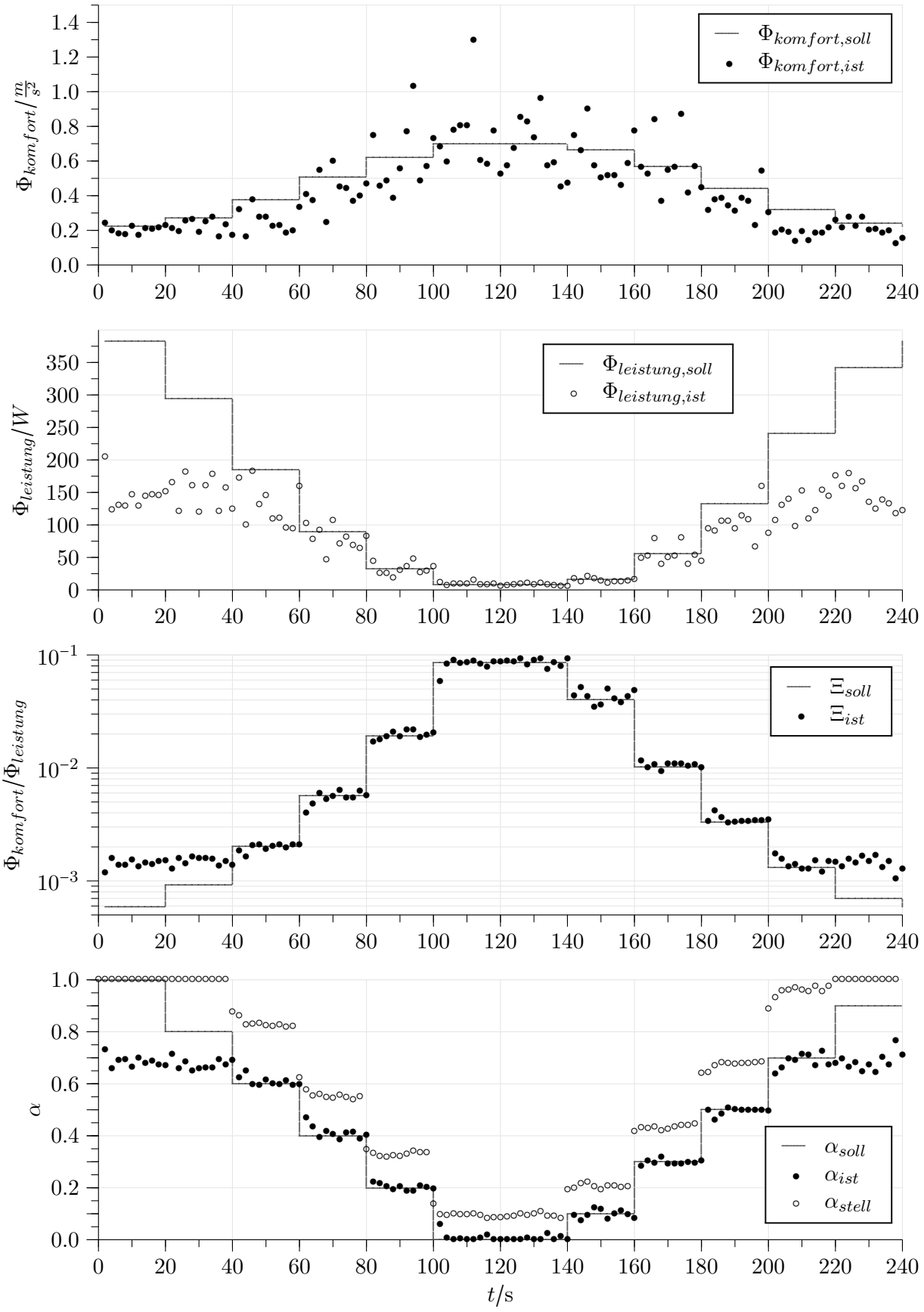


Abbildung 9.18: Reaktion des Zielrelationsregelkreises auf Änderungen der Soll-Ziele

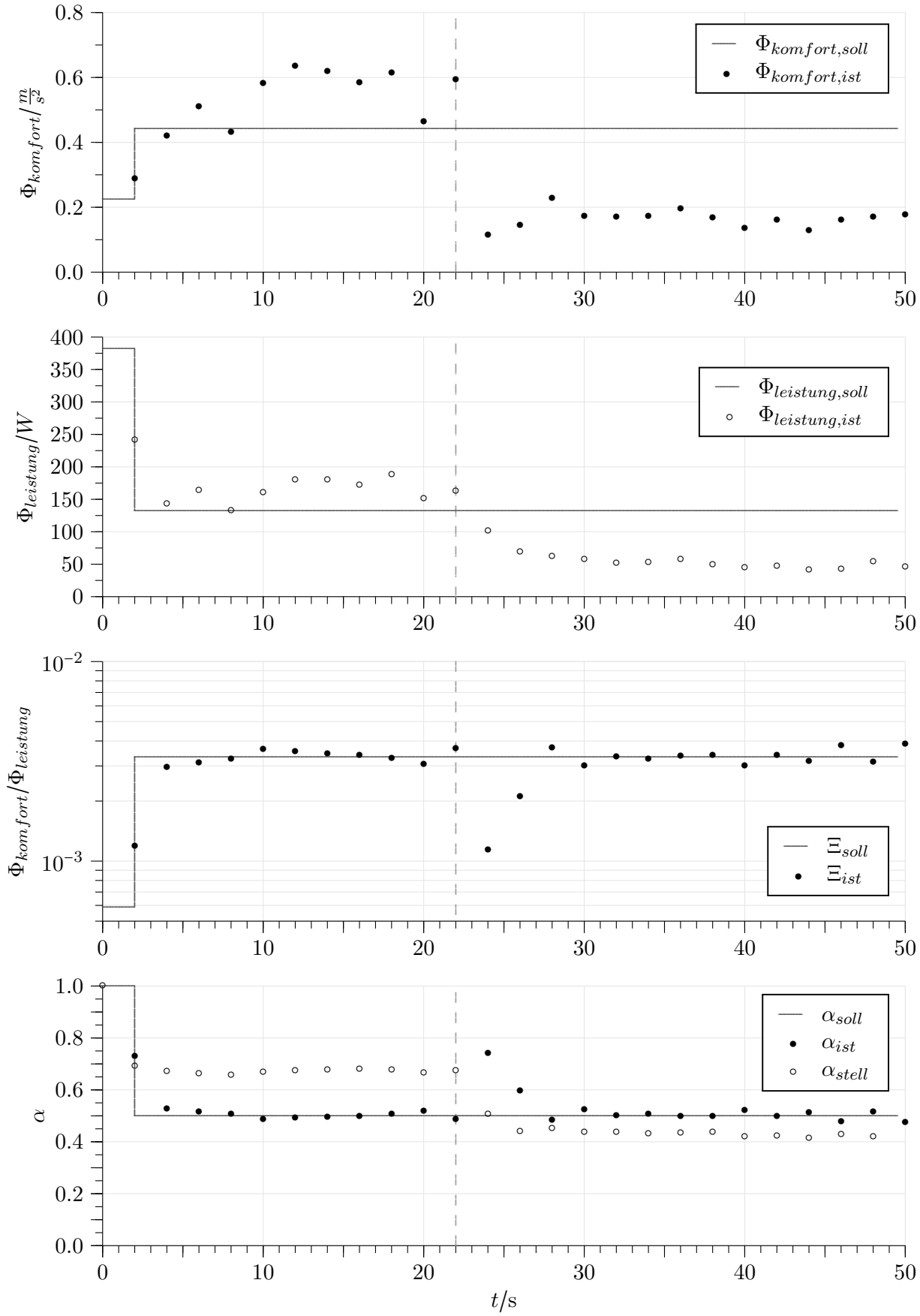


Abbildung 9.19: Reaktion des Zielrelationsregelkreises auf eine sprungartige Änderung der Anregungscharakteristik

9.3.2 Regelung auf absolute Zielvorgaben

Bei der Regelung auf absolute Zielvorgaben wird auf eine Umschaltung des Störverhalten verzichtet, da der Regelkreis wesentlich empfindlicher auf stochastische Anregungen reagiert. Beide Anregungsmodelle beeinflussen die Absolutwerte der realen Zielerfüllungen aufgrund ihres stochastischen Verhaltens auch ohne Änderung der Anregungsparameter bereits ausreichend.

Der Regelkreis wurde unter den folgenden Randbedingungen simuliert:

- Eine Aktualisierung der Reglerparameter findet alle $h = 5{,}0\,\mathrm{s}$ statt.
- Die Einstelldynamik auf neue Parametersätze kann bei diesem System vernachlässigt werden. Hieraus folgt $G_S(z) = 1$.
- Es wird ein I-Regler mit der Übertragungsfunktion $G_R(z) = \frac{Kz}{z-1}$ verwendet. Der Verstärkungsfaktor wird auf $K = 1{,}0$ gesetzt.
- Die Sollvorgabe ist der Leistungsbedarf des Systems, d. h. es wird auf $\Phi_{soll} = \Phi_{leistung,soll}$ geregelt. Als Regelgröße wird $\alpha = \beta_{komfort}$ verwendet.
- Für die Anregung wird das gefilterte Rauschsignal verwendet. Die Filtercharakteristik wird während der Simulation nicht verändert.

Abbildung 9.20 zeigt den Verlauf der Simulation. Die Führungsgröße $\Phi_{leistung,soll}$ wird in Stufen auf die Werte 200 W, 100 W, 20 W und 80 W verstellt. Das Systems folgt der geforderten Leistungsaufnahme prompt. Dies ist auf die Berechnungsvorschrift zur Bestimmung des neuen Paretopunktes zurückzuführen. Die hohe Reglerverstärkung sorgt dafür, dass dieser Punkt im nächsten Schritt direkt zum Einsatz kommt. Die mittleren Leistungsaufnahmen liegen mit 195,7 W, 107,5 W, 26,6 W und 75,6 W in der Nähe der geforderten Werte.

Der Skalierungsfaktor γ_{ist} dient zur Abschätzung der aktuell gültigen Paretofront. Eine Abhängigkeit von den Soll-Vorgaben ist nicht auszumachen. Im Großen und Ganzen spiegelt er die Höhe der gerade durchlaufenen Anregung wieder und ist weitgehend unabhängig vom eingestellten Paretopunkt.

Die Größe α_{soll} erfährt durch den Einfluss von γ_{ist} eine ständige Änderung, wie in Abbildung 9.20 unten zu sehen. Der Soll-Ist-Vergleich und die Stellgröße besitzen bei diesem Reglertyp keinen ähnlich anschaulichen Verlauf, wie beim Zielrelationsregler.

Das Stellsignal α_{stell} verbleibt in der Nähe von α_{ist}. Dies ist ein Indikator dafür, dass die gerade auf das System einwirkende Anregung eine ähnliche Charakteristik wie die bei der Optimierung verwendete Anregung besitzt, was bei dieser Simulation ja auch der Fall ist.

Die Regelung erweist sich als stabil und ist geeignet, eine Zielvorgabe im System einzustellen und im Mittel zu halten. Bei ungünstigeren Verhältnissen kann eine Stabilisierung über die Einstellung des Reglers oder über zusätzliche Filtermaßnahmen, bspw. der Größe γ_{ist}, erfolgen. Hervorzuheben ist das Führungsverhalten des Regelkreises. Bedingt durch die Abschätzung der Paretofront und der darauf aufbauenden Berechnungsvorschrift liegen die korrigierten Paretopunkte nach einem Führungssprung sofort in der Nähe der Zielvorgabe.

Die Größe γ_{ist} liefert ein Maß für die Höhe der gerade wirkenden Anregung. Bei diesem Regler besteht daher die Möglichkeit, den Skalierungsfaktor γ_{ist} und evtl. auch die Abweichung von α_{ist} zu α_{stell} für eine einfache Klassifikation der gerade wirkenden Anregung zu nutzen. Entsprechende Untersuchungen wurden hier jedoch nicht durchgeführt.

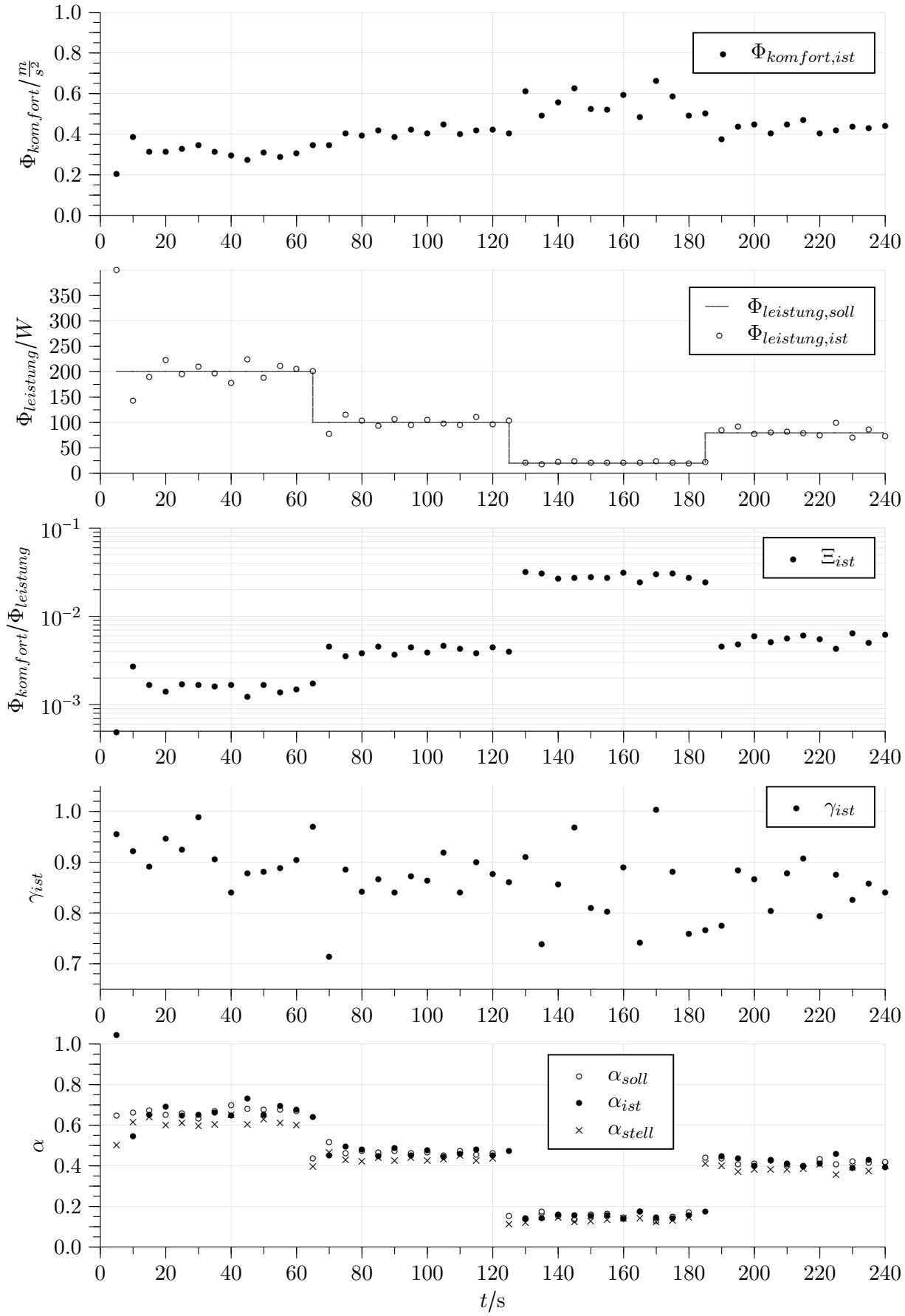


Abbildung 9.20: Ziel-Regelkreis auf absolute Zielvorgaben bei Änderungen des Soll-Ziels

9.4 Gradientenbasierte Zielregelung

Die in den vorangegangenen Abschnitten 9.2 und 9.3 vorgestellten Ziel-Regelkreise erreichen eine Anpassung des gewählten Paretopunktes anhand fester Regelungsgesetze. Aufgrund der starren Formulierung müssen einige Voraussetzungen für die Anwendung dieser Konzepte erfüllt sein (siehe Seite 151f). Dies gilt insbesondere für die Definition der Zielfunktionen, die einigen Einschränkungen unterliegt. So müssen die Zielfunktionen $\underline{f}(\underline{p})$ bei der Optimierung ein ähnliches Verhalten wie die Zielfunktionen $\underline{\Phi}_{ist}(\underline{p})$ im realen Betrieb aufweisen.

In diesem Abschnitt wird das Konzept einer gradientenbasierten Zielregelung vorgestellt, welche auf Betrachtungsweisen der Algorithmischen Differentiation (Kapitel 5) und Methoden der mathematischen Optimierung (Kapitel 3) aufbaut. Abbildung 9.21 zeigt die Regelungsstruktur. Den Kern bildet ein erweiterter Zustandsbeobachter, ein sog. Gradientenbeobachter, welcher in [MT06] beschrieben ist. Die Aufgabe dieses Gradientenbeobachters ist die Abschätzung des Gradienten $\partial\underline{\Phi}_{ist}/\partial\underline{p}^{\mathrm{T}}$. Auf Basis dieser Gradienten wird mit Hilfe eines speziellen Gradientenverfahrens eine Anpassung des Paretopunktes berechnet.

Über die Anwendung der Kettenregel lässt sich der Gradient der Zielerfüllung $\underline{\Phi}_{ist}$ nach dem eingestellten Paretopunkt $\underline{\alpha}$ bestimmen:

$$\nabla_{\underline{\alpha}}\underline{\Phi}_{ist} = \frac{\partial\underline{\Phi}_{ist}}{\partial\underline{p}^{\mathrm{T}}} \cdot \frac{\partial\underline{p}}{\partial\underline{\alpha}^{\mathrm{T}}} \tag{9.30}$$

Die Ableitung $\partial\underline{p}/\partial\underline{\alpha}^{\mathrm{T}}$ kann hierbei leicht über eine Interpolation der Urbildmenge ermittelt werden.

Das Regelungskonzept beruht auf der Beobachtung von Gradienten. Mit diesen Gradienten lässt sich die Richtung bestimmen, in die der aktuell gestellte Paretopunkt verschoben wird. Hierbei werden nur die Informationen der Urbildmenge genutzt. Es stellt daher wesentlich geringere Anforderungen an die Definition der Zielfunktionen:

- Eine übereinstimmende Definition der Zielkriterien $\underline{f}$ des Optimierungsmodells und der Kriterien $\underline{\Phi}_{ist}$ des realen Systems ist hier nicht zwingend erforderlich.
- Auf die Skalierungseigenschaften aus Abschnitt 9.1.4 kann verzichtet werden.
- Die gradientenbasierte Zielregelung muss nicht auf die Paretomenge einer einzelnen Anregungs- bzw. Betriebssituation beschränkt werden. Über eine parameter-

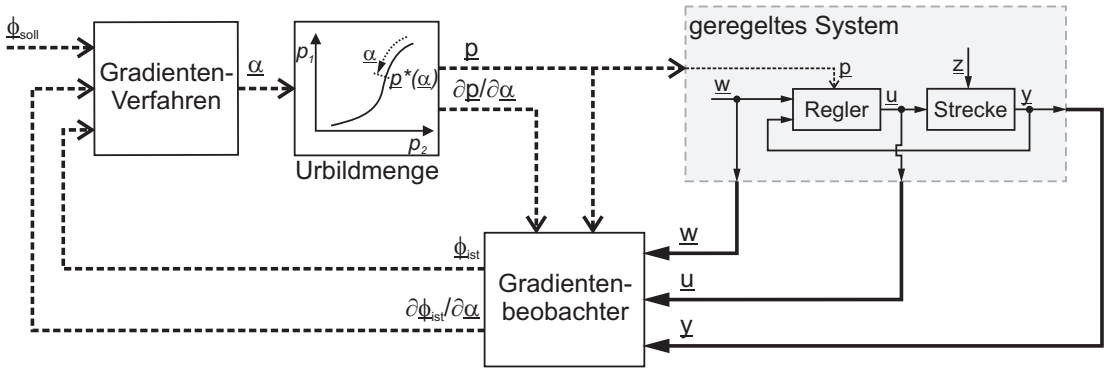


Abbildung 9.21: Aufbau der gradientenbasierten Zielregelung

abhängige Urbildmenge[2] lassen sich unterschiedliche Situationen darstellen und im Gradientenverfahren berücksichtigen. Das Gradientenverfahren arbeitet in diesem Fall auf der Gesamtheit der parameterabhängigen Urbildmenge und liefert hierüber auch eine Schätzung der aktuellen Betriebssituation.

Aufgrund des eingesetzten Gradientenverfahrens übertragen sich dessen Einschränkungen auch auf dieses Regelungskonzept. So können lokale Minima bei der Abbildung von $\underline{\alpha}_{stell}$ nach $\underline{\Phi}_{ist}$ Probleme verursachen, ebenso sollten Unstetigkeiten im Verlauf der Paretomenge vermieden werden.

9.4.1 Gradientenbeobachter

Der Gradientenbeobachter macht sich die Eigenschaft linearer Systeme zunutze, dass die Gradienten dieser Systeme nach den Parametern wiederum lineare Systeme darstellen (siehe Kapitel 5.4.4). Es liegt daher der Gedanke nahe, die Gradienten zur Laufzeit durch einen Beobachter abzuschätzen.

Ausgangspunkt eines solchen Beobachters ist ein lineares Modell der Regelstrecke. Die Zustandsraumdarstellung eines gestörten Systems ohne direkten Durchgriff lässt sich mit:

$$\begin{aligned} \dot{\underline{x}} &= \underline{\mathbf{A}}\,\underline{x} + \underline{\mathbf{B}}\,\underline{u} + \underline{\mathbf{E}}\,\underline{z} \\ \underline{y} &= \underline{\mathbf{C}}\,\underline{x} \end{aligned} \tag{9.31}$$

angeben. Für den Regler wird eine Ausgangsrückführung der Form

$$\underline{u} = -\underline{\mathbf{R}}(\underline{p})\,\underline{y} + \underline{\mathbf{S}}(\underline{p})\,\underline{w} \tag{9.32}$$

angenommen. Die Rückführmatrix $\underline{\mathbf{R}}$ und die Steuermatrix $\underline{\mathbf{S}}$ hängen von den Entwurfsparametern $\underline{p}$ ab. Durch Einsetzen von (9.32) in (9.31) ergibt sich die Zustandsraumdarstellung des geregelten Systems zu:

$$\begin{aligned} \dot{\underline{x}} &= \left(\underline{\mathbf{A}} - \underline{\mathbf{B}}\,\underline{\mathbf{R}}(\underline{p})\,\underline{\mathbf{C}}\right)\underline{x} + \underline{\mathbf{B}}\,\underline{\mathbf{S}}(\underline{p})\,\underline{w} + \underline{\mathbf{E}}\,\underline{z} \\ \underline{y} &= \underline{\mathbf{C}}\,\underline{x} \end{aligned} \tag{9.33}$$

Durch Differentiation nach $\underline{p}$ erhält man analog zu (5.28) die erweitere Zustandsraumdarstellung:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} \underline{x} \\ \hline \nabla_{p_1}\underline{x} \\ \vdots \\ \nabla_{p_n}\underline{x} \end{bmatrix} &= \left[\begin{array}{c|ccc} \underline{\mathbf{A}} - \underline{\mathbf{B}}\,\underline{\mathbf{R}}\,\underline{\mathbf{C}} & \underline{\mathbf{0}} & \cdots & \underline{\mathbf{0}} \\ \hline -\underline{\mathbf{B}}\,\frac{\partial}{\partial p_1}\underline{\mathbf{R}}\,\underline{\mathbf{C}} & \underline{\mathbf{A}} - \underline{\mathbf{B}}\,\underline{\mathbf{R}}\,\underline{\mathbf{C}} & \cdots & \underline{\mathbf{0}} \\ \vdots & \vdots & \ddots & \vdots \\ -\underline{\mathbf{B}}\,\frac{\partial}{\partial p_n}\underline{\mathbf{R}}\,\underline{\mathbf{C}} & \underline{\mathbf{0}} & \cdots & \underline{\mathbf{A}} - \underline{\mathbf{B}}\,\underline{\mathbf{R}}\,\underline{\mathbf{C}} \end{array}\right]\begin{bmatrix} \underline{x} \\ \hline \nabla_{p_1}\underline{x} \\ \vdots \\ \nabla_{p_n}\underline{x} \end{bmatrix} + \begin{bmatrix} \underline{\mathbf{S}} \\ \hline \frac{\partial}{\partial p_1}\underline{\mathbf{S}} \\ \vdots \\ \frac{\partial}{\partial p_n}\underline{\mathbf{S}} \end{bmatrix}\underline{w} + \begin{bmatrix} \underline{\mathbf{E}} \\ \hline \underline{\mathbf{0}} \\ \vdots \\ \underline{\mathbf{0}} \end{bmatrix}\underline{z} \\ \begin{bmatrix} \underline{y} \\ \hline \nabla_{p_1}\underline{y} \\ \vdots \\ \nabla_{p_n}\underline{y} \end{bmatrix} &= \left[\begin{array}{c|ccc} \underline{\mathbf{C}} & \underline{\mathbf{0}} & \cdots & \underline{\mathbf{0}} \\ \hline \underline{\mathbf{0}} & \underline{\mathbf{C}} & \cdots & \underline{\mathbf{0}} \\ \vdots & \vdots & \ddots & \vdots \\ \underline{\mathbf{0}} & \underline{\mathbf{0}} & \cdots & \underline{\mathbf{C}} \end{array}\right]\begin{bmatrix} \underline{x} \\ \hline \nabla_{p_1}\underline{x} \\ \vdots \\ \nabla_{p_n}\underline{x} \end{bmatrix} \end{aligned} \tag{9.34}$$

[2] Unterschiedliche Betriebssituationen können innerhalb eines Modells über entsprechende Parameter eingestellt werden. Diese Parameter werden durch die Anregung oder die jeweilige Betriebsbedingung vorgegeben und stellen keine Entwurfsparameter dar. Die Optimierung über die verschiedenen Situationen liefert demnach eine Schar an Paretomengen, eine sog. parameterabhängige Paretomenge.

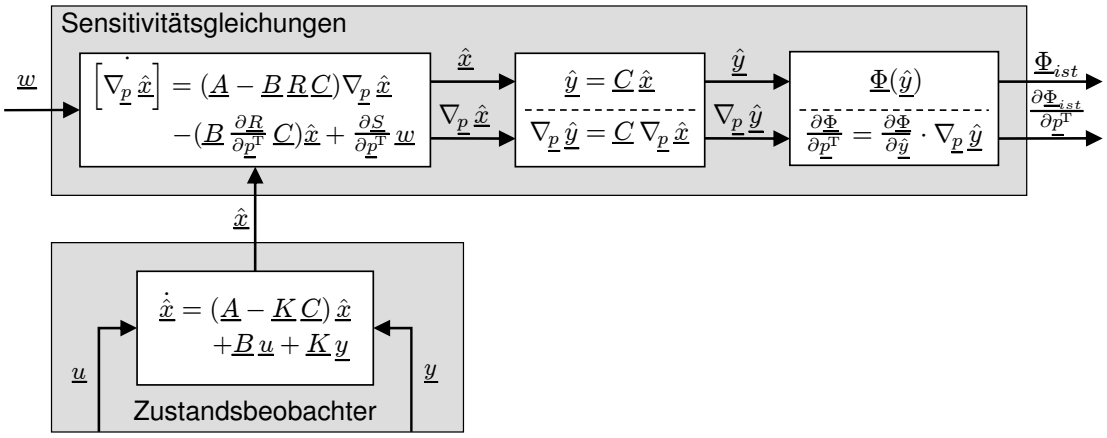


Abbildung 9.22: Aufbau des Gradientenbeobachters

welche neben den ursprünglichen Modellgleichungen auch die Sensitivitätsgleichungen nach den Reglerparametern $\underline{p}$ enthält.

Betrachtet man das Gleichungssystem (9.34) genauer, so ist erkennbar, dass die Zustände des ursprünglichen Systems $\underline{x}$ unabhängig von den Gradienten $\nabla_{p_i}\underline{x}$ sind. Da lediglich die Ausgänge $\underline{y}$ gemessen werden können, folgt daraus, dass die Gradienten $\nabla_{p_i}\underline{x}$ strukturell nicht beobachtbar sind. Allerdings unterliegen diese Größen aus zwei Gründen nicht der Beobachtbarkeitsbedingung:

- Die Anfangswerte der Zustände $\underline{x}(t = 0)$ sind unabhängig von den Parametern $\underline{p}$. Die Anfangswerte der Gradienten sind daher bekannt und können mit $\nabla_{p_i}\underline{x}(t = 0) = \underline{0}$ für alle $i = 1, \ldots, n$ angegeben werden.

- Weder die Störeingangsmatrix $\underline{\mathbf{E}}$ noch der Störeingang $\underline{z}$ hängen von den Reglerparametern $\underline{p}$ ab. Wegen des fehlenden Durchgriffs üben die Störungen keinen direkten Einfluss auf die Gradienten $\nabla_{p_i}\underline{x}$ aus. Die Gradienten unterliegen keiner Störung und sind nur von den Zuständen $\underline{x}$ und dem Steuervektor $\underline{w}$ abhängig, die prinzipiell beobachtbar sind.

Die Gradienten $\nabla_{p_i}\underline{x}$ stellen im Grund keine Zustände des Systems dar. Aus den oben genannten Gründen müssen lediglich die Zustände $\underline{x}$ über einen Zustandsbeobachter ermittelt werden. Die gewünschten Gradienten lassen sich durch eine Erweiterung des Beobachters um die Sensitivitätsgleichungen aus (9.34) abschätzen. Abbildung 9.22 zeigt diesen erweiterten Zustandsbeobachter. Die Ableitungen der nichtlinearen Zielkriterien können durch Anwendung der Methode der Algorithmischen Differentiation ermittelt und in die Sensitivitätsgleichungen integriert werden.

Wie in [MT06] gezeigt wurde, liegen die relativen Schätzfehler der Gradienten $\partial\underline{\Phi}_{ist}/\partial\underline{p}^{\mathrm{T}}$ in etwa in der gleichen Größenordnung wie die Fehler der beobachteten Zielgrößen $\underline{\Phi}_{ist}$.

9.4.2 Mehrziel-Gradientenverfahren

Die Anpassung des gewählten Paretopunktes geschieht mit Hilfe eines ableitungsbehafteten Optimierungsverfahrens. Aufgrund des Einsatzes in einem Zielregelkreis ergeben sich die folgenden Anforderungen an dieses Optimierungsverfahren:

- Die Auswahl des Paretopunktes erfolgt zur Laufzeit. Die Optimierung wird demnach nicht an einem Modell sondern direkt am System durchgeführt.

- Das System unterliegt veränderlichen Anregungsverhältnissen. Dies führt, wie auf Seite 139 dargelegt, zu sich ständig verändernden Zielfunktionen.

  Das Optimierungsverfahren muss geeignet sein, mit den sich ändernden Zielfunktionen umzugehen. Der optimale Punkt kann unter solchen Verhältnissen im Grunde nie erreicht werden. Es kann jedoch in jedem Schritt eine Annäherung der beobachteten Systemziele $\underline{\Phi}_{ist}$ an die Zielvorgaben $\underline{\Phi}_{soll}$ erfolgen. Das Optimierungsverfahren kann daher als ein komplexer Regler aufgefasst werden, der neben der Regeldifferenz auch die Gradienteninformation des Ist-Signals zur Berechnung des Stellsignals nutzt.

  Optimierungsverfahren, die zur Beschleunigung der Konvergenz Informationen aus vorangegangenen Optimierungsschritten nutzen, wie z. B. Verfahren der konjugierten Gradienten oder Quasi-Newton-Verfahren, können in diesem Zusammenhang nur bedingt eingesetzt werden. Insbesondere bei ständigen Veränderungen der Anregungsverhältnisse erweisen sich diese Informationen als unbrauchbar, da sie für jeweils andere Situationen und somit für andere Zielfunktionen ermittelt wurden.

Für die Anpassung des Systemverhaltens wird an dieser Stelle ein Mehrziel-Gradientenverfahren vorgeschlagen. Es setzt auf Ideen aus [Mün01, Mün03] auf und wurde erstmals in [HMS04] vorgestellt und für die Online-Optimierung einer Hydraulikpumpe in einem selbstoptimierenden Operator-Controller-Modul eingesetzt.

Das Gradientenverfahren arbeitet nach der Iterationsvorschrift (3.2). Analog zu der Goal-Attainment-Methode aus Kapitel 3.4.4, löst es ein Min-Max-Problem der Form:

$$\min_{\underline{x}} \max_{i} \vartheta_i(\underline{x}) \tag{9.35}$$

mit den Erfüllungsgraden

$$\vartheta_i = \frac{f_i(\underline{x}) - P_i}{s_i}, \quad i = 1, \ldots, m \tag{9.36}$$

Hierbei beschreibt $\underline{P}$ den Zielpunkt und $\underline{s}$ den Gewichtungsvektor. Die Aufgabe dieses „Reglers“ besteht also darin, in jedem Optimierungsschritt k eine Verringerung des größten Erfüllungsgrades ϑ_i zu erreichen.

Der Kern des Verfahrens besteht in der Berechnung einer Abstiegsrichtung $\underline{d}_k$, die zu einer Verringerung der Maximumfunktion in (9.35) führt. Wendet man ein Gradientenverfahren direkt auf (9.35) an, so besitzt immer nur ein einzelnes Ziel einen maximalen Erfüllungsgrad. Schreitet man in Abstiegsrichtung fort, so hat dies besonders in der Nähe der Lösung einen ständigen Wechsel des gerade betrachteten Erfüllungsgrades zur Folge. Hieraus resultieren im Allgemeinen extrem kleine Schrittweiten. Um diese zu vermeiden werden für die Abstiegsrichtung mehrere Ziele gemeinsam berücksichtigt. Die Festlegung dieser Ziele erfolgt mit Hilfe einer *Active-Set*-Strategie, d. h. es wird zwischen *aktiven* und *passiven* Zielen unterschieden. Die Berechnung der Abstiegsrichtung erfolgt nur anhand der aktiven Ziele. Diese weisen die größten Erfüllungsgrade ϑ_i auf, die Erfüllungsgrade der passiven Ziele sind demgegenüber klein und werden nicht berücksichtigt.

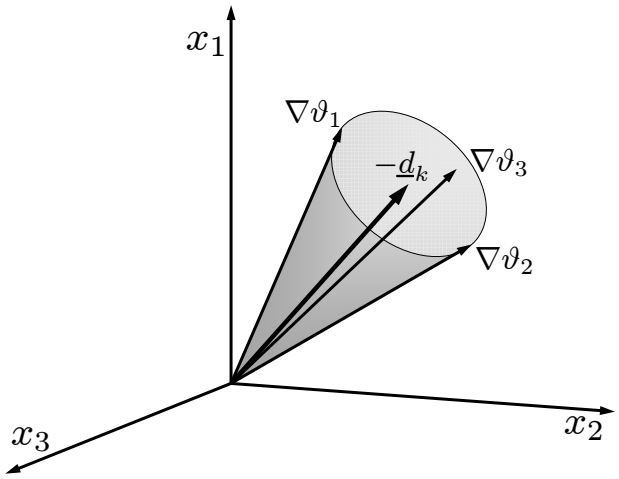


Abbildung 9.23: Geometrischer Ansatz zur Bestimmung der Abstiegsrichtung im Urbildraum

Bestimmung der Abstiegsrichtung

Die Abstiegsrichtung wird über einen geometrisch motivierten Ansatz aus [Mün01] bestimmt. Gesucht wird eine Richtung $\underline{d}_k$, in der alle aktiven Erfüllungsgrade eine Verringerung erfahren. Für einen einzelnen Erfüllungsgrad ϑ_i kann diese Richtung leicht über den Gradienten angegeben werden. Spannt man nun einen Kegel auf, der die Gradienten aller aktiven Erfüllungsgrade möglichst eng umschließt, so liegt die gesuchte Abstiegsrichtung $\underline{d}_k$ in entgegengesetzter Richtung zur Kegelachse. Abbildung 9.23 skizziert diesen Ansatz im Urbildraum für drei Zielgrößen und Optimierungsparameter.

Der Kegel wird über die Minimierung des folgenden quadratischen Optimierungsproblems bestimmt:

$$\min_{\underline{x}_k} \underline{d}_k{}^{\mathrm{T}} \underline{d}_k \tag{9.37}$$

mit den Nebenbedingungen:

$$\nabla\vartheta_i{}^{\mathrm{T}}\, \underline{d}_k \leq -|\nabla\vartheta_i|\,, \quad i \in \mathcal{I} \tag{9.38}$$

und der Indexmenge $\mathcal{I}$ der aktiven Ziele.

Bestimmung der aktiven Ziele

Die Indexmenge der aktiven Zielgrößen $\mathcal{I}$ wird über eine ϵ-Umgebung um den maximalen Wert $\vartheta_{max} = \max_j \vartheta_j$ bestimmt:

$$\mathcal{I} = \left\{ i :\ \vartheta_i \geq \max_j \vartheta_j - \epsilon \right\} \tag{9.39}$$

Typischer Weise startet eine Optimierung mit relativ großen Unterschieden zwischen den Erfüllungsgraden ϑ_i. Mit der Annäherung an den gewünschten Paretopunkt werden die Differenzen zwischen den Erfüllungsgraden der aktiven Ziele zwangsläufig immer geringer. Eine Adaption der ϵ-Umgebung ist hier sinnvoll. Weitab der Lösung kann ein großes ϵ gewählt werden, mit der Annäherung an den gesuchten Paretopunkt muss ϵ gegen Null streben.

Ein Maß für die Annäherung an die Lösung ist der Winkel γ zwischen dem Kegelmantel und der Kegelachse. Dieser ergibt sich aus (9.37) und (9.38) zu:

$$\cos\gamma = \frac{1}{\sqrt{\underline{d}_k{}^{\mathrm{T}}\,\underline{d}_k}} \tag{9.40}$$

Bei $\gamma = \frac{\pi}{2}$ sind die Gradienten der aktiven Ziele voneinander linear abhängig und der gesuchte Paretopunkt ist erreicht.

Diese Beziehung wird bspw. von der folgenden stufenweise arbeitenden Adaptionsvorschrift genutzt:

$$\epsilon_{k+1} = \begin{cases} 2\,\epsilon_k & : \quad \epsilon_k < \dfrac{\epsilon_0}{\underline{d}_k{}^{\mathrm{T}}\,\underline{d}_k} \\ \frac{1}{2}\,\epsilon_k & : \quad \epsilon_k > \dfrac{2\,\epsilon_0}{\underline{d}_k{}^{\mathrm{T}}\,\underline{d}_k} \\ \epsilon_k & : \quad \text{andernfalls} \end{cases} \tag{9.41}$$

Mit diese Vorschrift konnte in den bisher betrachteten Anwendungen gute Ergebnisse erzielt werden (vgl. [HMS04, EA06]):

10 Zusammenfassung

Diese Arbeit ist entstanden im Rahmen des Sonderforschungsbereiches 614 „Selbstoptimierende Systeme des Maschinenbaus“. Das Ziel bestand darin, Konzepte für eine selbstoptimierende Regelung in hierarchisch verteilten mechatronischen Systemen zu entwickeln und zu erproben.

Den Ausgangspunkt der Arbeit bildet die modular-hierarchische OCM-Architektur für intelligente mechatronische Systeme. Diese Struktur ergibt sich aus der Verknüpfung einzelner Aggregate zu einem komplexen Gesamtsystem. Bei selbstoptimierenden Systemen kommt der Informationsverarbeitung eine besondere Rolle zu. Das Konzept des Operator-Controller-Moduls (OCM) ist geeignet, den speziellen Anforderungen selbstoptimierender geregelter Systeme gerecht zu werden, da eine klare Trennung zwischen regelungstechnischen Bestandteilen im Controller und den selbstoptimierenden Prozessen im kognitiven Operator erreicht wird. Jedes Aggregat wird über ein eigenes OCM angesteuert, überwacht und optimiert. Die Verknüpfung der OCM führt zu der hierarchisch organisierten OCM-Architektur.

Die verteilte Selbstoptimierung der unteren aktornahen Ebenen stellt besondere Anforderungen an die Methoden und Verfahren, da aufgrund starker physikalischer Wechselwirkungen eine unabhängige Betrachtung der einzelnen Aggregate nicht möglich ist. Bei bisherigen Ansätzen zur Realisierung selbstoptimierender Systeme stand die Betrachtung einzelner Funktionsmodule im Vordergrund; hierarchische Ansätze wurden lediglich für die höheren Ebenen der lose gekoppelten autonomen oder vernetzten Systeme entwickelt (siehe z. B. [Klö09]). Diese Lücke konnte in dieser Arbeit mit dem Konzept einer hierarchischen, über die OCM-Architektur verteilten Wissensbasis ein Stück weit geschlossen werden. Mit diesem Konzept gelingt es, die gegenseitigen Wechselwirkungen durch mathematische Verhaltensmodelle abzubilden und in der Optimierung des Systemverhaltens zu berücksichtigen.

Die Wissensbasis setzt auf einem hierarchischen Modell auf, welches auf jeder Ebene der Aggregatstruktur das Verhalten des gesamten unterlagerten Teilsystems beschreibt. Bei dem Aufbau des Modells wird eine starke Kapselung verfolgt. So verfügen die einzelnen OCM initial nur über Informationen ihres eigenen Verhaltens. Durch den Austausch reduzierter Verhaltensmodelle ergibt sich erst zur Laufzeit ein Bild des Umfelds mit den unterlagerten und überlagerten Aggregaten. Aufgrund dieser Modellreduktion bleiben die Modelle auch auf den höheren Ebenen ausgewogen: detaillierte Effekte der unteren Ebenen werden verallgemeinert und entfernt.

Die Wissensbasis dient als zentraler Speicherort für Modelle, Systemkonfigurationen, Anregungs- und Situationsdaten sowie weiteren Informationen. Verschiedene Prozesse, z. B. Optimierungsverfahren, Selbstoptimierungsprozesse und Planungsverfahren, erhalten gemeinsam über definierte Schnittstellen Zugriff auf aktuelle Daten.

Paretooptimale Systemkonfigurationen werden mit Hilfe einer verteilten Optimierung des hierarchischen Systems ermittelt. Hierfür wurden zwei Ansätze vorgestellt: die Top-Level-Optimierung und die Multi-Objective-Bottom-Up-Optimierung (MOBU). Mit der

MOBU-Methode werden robuste Ergebnisse erzielt, da die Zielfunktionen aller Subsysteme in das Ergebnis einfließen, zudem bleibt die starke Kapselung der OCM erhalten. Aus diesen Gründen stellt sie in dieser Arbeit die bevorzugte Optimierungsmethode dar.

Der Prozess der Selbstoptimierung kann auf vielfältige Art und Weise innerhalb des OCM realisiert werden. In dieser Arbeit wurde das Konzept einer selbstoptimierenden Regelung auf Basis paretooptimaler Systemkonfigurationen verfolgt. Dieses Konzept setzt auf eine Trennung der Mehrzieloptimierung von dem eigentlichen Selbstoptimierungsprozess. Für die Einhaltung der geforderten Systemziele bei wechselnden Anregungsverhältnissen wurde ein sogenannter Ziel-Regelkreis vorgeschlagen, der im Prinzip analog zu einer klassischen digitalen Regelung arbeitet. So können einige Analyse- und Synthesemethoden für digitale Regelungen hier zum Einsatz kommen.

Notwendige Voraussetzung für die hierarchische Optimierung und den Ziel-Regelkreis sind leistungsfähige Verfahren der Mehrzieloptimierung. Betrachtet wurden schnell konvergierende Optimierungsverfahren, die auf Basis von Gradienteninformationen arbeiten. Die Genauigkeit, mit der die Gradienten verfügbar sind, wirkt sich entscheidend auf die Konvergenz, die Exaktheit der Ergebnisse, vor allem aber auch auf die Robustheit der Verfahren aus. So konnten viele Optimierungsprobleme in dieser Arbeit nur durch die Verfügbarkeit hochgenauer Ableitungen erfolgreich gelöst werden. Eine fortschrittliche Methode die Ableitungen von Funktionen zu ermitteln, die in Form eines Computerprogramms oder Algorithmus vorliegen, ist die Algorithmische Differentiation (AD). Es wurde dargestellt, wie die Ableitungen von Simulationsprogrammen erfolgreich berechnet werden können.

Als Anwendungsbeispiel dient das Prüfstandsmodell eines neuartigen Unterflur-Federungssystems für ein Schienenfahrzeug. Das Konzept des hierarchischen Modells und der hierarchischen MOBU-Optimierung wurden an diesem Beispiel umgesetzt. Zwei konkrete Ziel-Regelkreise mit unterschiedlichen Regelungsaufgaben, die Regelung auf ein Zielverhältnis und die Regelung auf absolute Zielvorgaben, wurden auf der Grundlage der Optimierungsergebnisse aufgebaut und verifiziert.

Neben diesen beiden Ziel-Regelkreisen wurde das Konzept einer gradientenbasierten Zielregelung vorgestellt. Den Kern des Verfahrens bildet ein Mehrziel-Gradientenverfahren, welches die Annäherung des aktuell eingestellten Paretopunktes an die Zielvorgaben erreicht. Die für das Verfahren notwendigen Gradienten werden zur Laufzeit mit Hilfe eines sog. Gradientenbeobachters abgeschätzt. Dieser Gradientenbeobachter greift die Idee der Algorithmischen Differentiation auf, um die Sensitivitätsgleichungen des Modells der Regelstrecke zu erstellen. Ein klassischer Zustandsbeobachter kann mit diesen Gleichungen erweitert und zu Abschätzung der Gradienten genutzt werden.

A Anhang

A.1 Modellreduktionsverfahren für lineare Systeme

Im Folgenden werden die Verfahren zur Reduktion von LZI-Modellen vorgestellt, die im Rahmen dieser Arbeit untersucht wurden. Die beschriebenen Verfahren setzen auf der linearen Zustandsraumdarstellung auf. Das System wird über unterschiedliche Ansätze derart transformiert, dass es in relevante Zustandsgrößen $\underline{x}_1$ und nicht relevante Zustandsgrößen $\underline{x}_2$ eingeteilt werden kann:

$$\begin{aligned} \begin{bmatrix} \dot{\underline{x}}_1 \\ \dot{\underline{x}}_2 \end{bmatrix} &= \begin{bmatrix} \underline{\mathbf{A}}_{11} & \underline{\mathbf{A}}_{12} \\ \underline{\mathbf{A}}_{21} & \underline{\mathbf{A}}_{22} \end{bmatrix} \begin{bmatrix} \underline{x}_1 \\ \underline{x}_2 \end{bmatrix} + \begin{bmatrix} \underline{\mathbf{B}}_1 \\ \underline{\mathbf{B}}_2 \end{bmatrix} \underline{u} \\ \underline{y} &= \begin{bmatrix} \underline{\mathbf{C}}_1 & \underline{\mathbf{C}}_2 \end{bmatrix} \begin{bmatrix} \underline{x}_1 \\ \underline{x}_2 \end{bmatrix} + \underline{\mathbf{D}}\,\underline{u} \end{aligned} \tag{A.1}$$

Das Modell kann so um die nicht relevanten Zustände reduziert werden und man erhält das reduzierte Modell:

$$\begin{aligned} \dot{\underline{x}}_r &= \underline{\mathbf{A}}_r\,\underline{x}_r + \underline{\mathbf{B}}_r\,\underline{u} \\ \underline{y}_r &= \underline{\mathbf{C}}_r\,\underline{x}_r + \underline{\mathbf{D}}_r\,\underline{u} \end{aligned} \tag{A.2}$$

A.1.1 Modale Ordnungsreduktion

Modale Ordnungsreduktionsverfahren beruhen auf der Transformation eines linearen Modells auf Modalform. Über die Transformationsmatrix $\underline{\mathbf{V}}$ und den Modalkoordinaten $\underline{z}$ ergibt sich mit:

$$\underline{x} = \underline{\mathbf{V}}\,\underline{z} \tag{A.3}$$

das lineare Modell:

$$\begin{aligned} \dot{\underline{z}} &= \underline{\mathbf{\Lambda}}\,\underline{z} + \underline{\mathbf{B}}^*\,\underline{u} \\ \underline{y} &= \underline{\mathbf{C}}^*\,\underline{z} + \underline{\mathbf{D}}\,\underline{u} \end{aligned} \tag{A.4}$$

Die Transformationsmatrix ist mit den Eigenvektoren von $\underline{\mathbf{A}}$ besetzt. Die transformierten Matrizen der Zustandsraumdarstellung berechnen sich aus:

$$\underline{\mathbf{\Lambda}} = \underline{\mathbf{V}}^{-1}\,\underline{\mathbf{A}}\,\underline{\mathbf{V}} = \operatorname{diag}(\lambda_1, \ldots, \lambda_n) \tag{A.5}$$

$$\underline{\mathbf{B}}^* = \underline{\mathbf{V}}^{-1}\,\underline{\mathbf{B}} \tag{A.6}$$

$$\underline{\mathbf{C}}^* = \underline{\mathbf{C}}\,\underline{\mathbf{V}} \tag{A.7}$$

Die Matrix $\underline{\mathbf{\Lambda}}$ enthält die Eigenwerte von $\underline{\mathbf{A}}$ auf der Hauptdiagonalen[1]. Diese Lineartransformation verändert das Verhalten des Systems nicht.

[1] Bei mehrfachen Eigenwerten sind diese komplex. Um eine komplexe Matrix $\underline{\mathbf{\Lambda}}$ zu vermeiden, wird diese meist in Jordanscher Normalform dargestellt. Die untere Nebendiagonale ist dann mit eins besetzt.

Das System (A.4) liegt als Parallelschaltung von n entkoppelten Subsystemen vor. Durch Einteilung des Systems in dominante und nichtdominante Eigenwerte erhält man:

$$\begin{aligned}\begin{bmatrix}\dot{\underline{z}}_1\\ \dot{\underline{z}}_2\end{bmatrix} &= \begin{bmatrix}\underline{\mathbf{\Lambda}}_1 & 0\\ 0 & \underline{\mathbf{\Lambda}}_2\end{bmatrix}\begin{bmatrix}\underline{z}_1\\ \underline{z}_2\end{bmatrix} + \begin{bmatrix}\underline{\mathbf{B}}_1^*\\ \underline{\mathbf{B}}_2^*\end{bmatrix}\underline{u}\\ \underline{y} &= \begin{bmatrix}\underline{\mathbf{C}}_1^* & \underline{\mathbf{C}}_2^*\end{bmatrix}\begin{bmatrix}\underline{z}_1\\ \underline{z}_2\end{bmatrix} + \underline{\mathbf{D}}\,\underline{u}\end{aligned} \tag{A.8}$$

Die Idee der modalen Ordnungsreduktion besteht darin, die nichtdominanten Eigenwerte einfach zu entfernen. Die verbleibende Differentialgleichung bildet das reduzierte Modell:

$$\begin{aligned}\dot{\underline{z}}_1 &= \underline{\mathbf{\Lambda}}_1\,\underline{z}_1 + \underline{\mathbf{B}}_1^*\,\underline{u}\\ \underline{y}_r &= \underline{\mathbf{C}}_1^*\,\underline{z}_1 + \underline{\mathbf{D}}\,\underline{u}\end{aligned} \tag{A.9}$$

Eine Schwierigkeit besteht in der Trennung zwischen dominanten und nichtdominanten Eigenwerten. Litz stellt in [Lit79a, Lit79b] ein Dominanzmaß vor, das neben der Lage des Eigenwertes auch die Steuerbarkeit und Beobachtbarkeit berücksichtigt. Das Dominanzmaß D_{kij} für einen Übertragungspfad wird für einen Eigenwert λ_k über die Antwort eines Ausgangs y_j auf einen Sprung am Eingang u_i wie folgt bestimmt:

$$D_{kij} = \left|\frac{b_{ki}\,c_{jk}}{\lambda_k}\right| \tag{A.10}$$

Die Koeffizienten b_{ki} und c_{jk} entsprechen den zum Übertragungspfad gehörenden Elementen der Matrizen $\underline{\mathbf{B}}^*$ und $\underline{\mathbf{C}}^*$.

Üblicherweise wird die Dominanz eines Eigenwertes über das Maximum der Übertragungspfade:

$$D_{max,k} = \max_i\left(\max_j D_{kij}\right) \tag{A.11}$$

oder über die Summe der Übertragungspfade:

$$D_{sum,k} = \sum_i\sum_j D_{kij} \tag{A.12}$$

bestimmt. Die Eigenwerte mit den größten Dominanzmaßen werden in das reduzierte Modell übernommen.

Das Modell (A.9) ist im Allgemeinen nicht in der Lage, die stationäre Endlage des Originalmodells wiederzugeben. Um diesen stationären Fehler zu vermeiden, muss der Zustandsvektor $\underline{z}_2$ im reduzierten Modell berücksichtigt werden. In der Literatur sind verschiedene Ansätze zu finden, die diesen Nachteil durch eine Korrektur der Eingangs- und/oder Ausgangsmatrix beheben (siehe z. B. [Mar66, Lit79b, Gut91]).

A.1.2 Balanciertes Abschneiden

Die Ordnungsreduktion mit einer balancierten Zustandsraumdarstellung wurde zuerst von Moore in [Moo81] vorgestellt. Die Idee einer balancierten Darstellung besteht darin, das Modell derart zu transformieren, dass jeder Zustand in gleichem Maße steuerbar und beobachtbar ist.

Als Maß für die Steuerbarkeit und Beobachtbarkeit werden hier die Gramsche Steuerbarkeitsmatrix $\underline{\mathbf{P}}$ und Beobachtbarkeitsmatrix $\underline{\mathbf{Q}}$ herangezogen, die sich über die Ljapunow-Gleichungen:

$$\underline{\mathbf{A}}\,\underline{\mathbf{P}} + \underline{\mathbf{P}}\,\underline{\mathbf{A}}^{\mathrm{T}} = -\,\underline{\mathbf{B}}\,\underline{\mathbf{B}}^{\mathrm{T}} \tag{A.13}$$

$$\underline{\mathbf{A}}^{\mathrm{T}}\underline{\mathbf{Q}} + \underline{\mathbf{Q}}\,\underline{\mathbf{A}} = -\,\underline{\mathbf{C}}^{\mathrm{T}}\underline{\mathbf{C}} \tag{A.14}$$

bestimmen lassen.

Ein System gilt genau dann als balanciert, wenn für die Beobachtbarkeits- und Steuerbarkeitsmatrizen gilt:

$$\underline{\tilde{\mathbf{P}}} = \underline{\tilde{\mathbf{Q}}} = \underline{\boldsymbol{\Sigma}} = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n) \tag{A.15}$$

Die Diagonale von $\underline{\boldsymbol{\Sigma}}$ ist hierbei mit den sogenannten Singulärwerten besetzt. Diese Singulärwerte sind absteigend der Größe nach sortiert und werden als „Dominanzmaß“ für die Ordnungsreduktion herangezogen. Niedrige Werte stehen für eine „schlechte“, hohe Werte für eine „gute“ Steuer- und Beobachtbarkeit. Für die Reduktion der Ordnung werden Zustände mit niedrigen Singulärwerten entfernt, da diese nur einen geringen Einfluss auf das Systemverhalten haben.

Es existieren verschiedene Algorithmen zur Berechnung einer balancierten Darstellung. Im Folgenden wird stellvertretend der Algorithmus von Laub vorgestellt [LHPW87]:

1. Bestimmung der Steuerbarkeitsmatrix $\underline{\mathbf{P}}$ und der Beobachtbarkeitsmatrix $\underline{\mathbf{Q}}$ des Ausgangssystems und Cholesky-Zerlegung dieser Matrizen:

$$\underline{\mathbf{P}} = \underline{\mathbf{L}}_c\,\underline{\mathbf{L}}_c^{\mathrm{T}} \tag{A.16}$$

$$\underline{\mathbf{Q}} = \underline{\mathbf{L}}_o\,\underline{\mathbf{L}}_o^{\mathrm{T}} \tag{A.17}$$

2. Singulärwertzerlegung des Produktes der Cholesky-Faktoren:

$$\underline{\mathbf{L}}_o^{\mathrm{T}}\underline{\mathbf{L}}_c = \underline{\mathbf{U}}\,\underline{\boldsymbol{\Sigma}}\,\underline{\mathbf{V}}^{\mathrm{T}} \tag{A.18}$$

3. Die Transformationsmatrix und ihre Inverse berechnen sich über:

$$\underline{\mathbf{T}} = \underline{\mathbf{L}}_c\,\underline{\mathbf{V}}\,\underline{\boldsymbol{\Sigma}}^{-1/2} \tag{A.19}$$

$$\underline{\mathbf{T}}^{-1} = \underline{\boldsymbol{\Sigma}}^{-1/2}\underline{\mathbf{U}}^{\mathrm{T}}\underline{\mathbf{L}}_o^{\mathrm{T}} \tag{A.20}$$

4. Schließlich erhält man das balancierte Zustandsraummodell:

$$\underline{\tilde{\mathbf{A}}} = \underline{\mathbf{T}}^{-1}\,\underline{\mathbf{A}}\,\underline{\mathbf{T}} = \underline{\boldsymbol{\Sigma}}^{-1/2}\underline{\mathbf{U}}^{\mathrm{T}}\,\underline{\mathbf{L}}_o^{\mathrm{T}}\,\underline{\mathbf{A}}\,\underline{\mathbf{L}}_c\,\underline{\mathbf{V}}\,\underline{\boldsymbol{\Sigma}}^{-1/2} \tag{A.21}$$

$$\underline{\tilde{\mathbf{B}}} = \underline{\mathbf{T}}^{-1}\,\underline{\mathbf{B}} \quad = \underline{\boldsymbol{\Sigma}}^{-1/2}\underline{\mathbf{U}}^{\mathrm{T}}\,\underline{\mathbf{L}}_o^{\mathrm{T}}\,\underline{\mathbf{B}} \tag{A.22}$$

$$\underline{\tilde{\mathbf{C}}} = \underline{\mathbf{C}}\,\underline{\mathbf{T}} \quad = \underline{\mathbf{C}}\,\underline{\mathbf{L}}_c\,\underline{\mathbf{V}}\,\underline{\boldsymbol{\Sigma}}^{-1/2} \tag{A.23}$$

Die Diagonalmatrix $\underline{\boldsymbol{\Sigma}}$ enthält anschließend die sogenannten Singulärwerte. Die Besonderheit dieses Verfahrens liegt darin, dass es mit den Singulärwerten bereits ein Dominanzmaß für die Reduktion mitliefert. Anhand der Singulärwerte kann man erkennen, an welcher Stelle sich ein System gut zerlegen lässt. Weisen zwei benachbarte Singulärwerte

einen großen Abstand auf $\sigma_k \gg \sigma_{k+1}$, so markiert dies in der Regel eine geeignete Stelle zum „Abschneiden“ der niedrigen Singulärwerte.

Die reduzierten Modelle erreichen eine sehr gute Approximation des Systemverhaltens bei gleichzeitig geringer Modellordnung. In der Regel ist es hierin den modalen Ordnungsreduktionsverfahren überlegen. Allerdings kommt es beim balancierten Abschneiden zu einer Verschiebung der Eigenwertlagen.

Da der Zustandsvektor $\underline{x}_2$ im reduzierten System nicht berücksichtigt wird, ist das Ergebnis, ähnlich wie bei den modalen Reduktionsverfahren, nicht stationär genau. Der stationäre Fehler kann auch hier bspw. mit der Korrektur von Guth [Gut91] oder in Kombination mit der nachfolgend beschriebenen *singulären Pertubation* beseitigt werden.

A.1.3 Singuläre Pertubation

Der Ansatz der *singulären Pertubation* ist geeignet Systeme zu vereinfachen, die in ein „schnelles“ und ein „langsames“ Teilsystem zerlegt werden können. In einer Übergangsphase des langsamen Teilsystems erreicht das schnelle Teilsystem frühzeitig seinen stationären Endwert. Bei der singulären Pertubation nimmt man an, dass dieser Endwert augenblicklich, quasi „unendlich schnell“ erreicht wird. Das schnelle Teilsystem kann daher aus dem Modell entfernt werden; der Einfluss des schnellen Teilsystems auf das reduzierte Modell wird über den stationären Endwert abgebildet.

Setzt man den stationären Endwert $\underline{x}_{2s}$ mit $\dot{\underline{x}}_{2s} = 0$ in die Zustandsgleichung ein, erhält man:

$$\begin{bmatrix} \dot{\underline{x}}_1 \\ 0 \end{bmatrix} = \begin{bmatrix} \underline{\mathbf{A}}_{11} & \underline{\mathbf{A}}_{12} \\ \underline{\mathbf{A}}_{21} & \underline{\mathbf{A}}_{22} \end{bmatrix} \begin{bmatrix} \underline{x}_1 \\ \underline{x}_{2s} \end{bmatrix} + \begin{bmatrix} \underline{\mathbf{B}}_1 \\ \underline{\mathbf{B}}_2 \end{bmatrix} \underline{u} \tag{A.24}$$

Durch Umformen kann der stationäre Endwert mit:

$$\underline{x}_{2s} = -\underline{\mathbf{A}}_{22}^{-1} \left(\underline{\mathbf{A}}_{21}\, \underline{x}_1 + \underline{\mathbf{B}}_2\, \underline{u} \right) \tag{A.25}$$

bestimmt werden. Setzt man (A.25) wieder in die Zustandsraumdarstellung ein, ergibt sich das reduzierte System:

$$\begin{aligned} \dot{\tilde{\underline{x}}}_1 &= \underbrace{\left(\underline{\mathbf{A}}_{11} - \underline{\mathbf{A}}_{12}\, \underline{\mathbf{A}}_{22}^{-1}\, \underline{\mathbf{A}}_{21} \right)}_{\underline{\mathbf{A}}_r} \tilde{\underline{x}}_1 + \underbrace{\left(\underline{\mathbf{B}}_1 - \underline{\mathbf{A}}_{12}\, \underline{\mathbf{A}}_{22}^{-1}\, \underline{\mathbf{B}}_2 \right)}_{\underline{\mathbf{B}}_r} \underline{u} \\ \tilde{\underline{y}} &= \underbrace{\left(\underline{\mathbf{C}}_1 - \underline{\mathbf{C}}_2\, \underline{\mathbf{A}}_{22}^{-1}\, \underline{\mathbf{A}}_{21} \right)}_{\underline{\mathbf{C}}_r} \tilde{\underline{x}}_1 + \underbrace{\left(\underline{\mathbf{D}} - \underline{\mathbf{C}}_2\, \underline{\mathbf{A}}_{22}^{-1}\, \underline{\mathbf{B}}_2 \right)}_{\underline{\mathbf{D}}_r} \underline{u} \end{aligned} \tag{A.26}$$

Prinzipiell kann bei der singulären Pertubation die Stabilität des Systems nicht gewährleistet werden. Die Berücksichtigung des schnellen Teilsystems über die stationären Endwerte resultiert u. a. in einer Modifikation der Durchgriffsmatrix. Bei Systemen, die ursprünglich keinen Durchgriff besitzen, gilt nach der Reduktion im Allgemeinen $\underline{\mathbf{D}}_r \neq \underline{\mathbf{0}}$. Im Zusammenhang mit einem hierarchischen Modell ist dies jedoch weniger störend, da die reduzierten Modelle als Submodelle in überlagerten Modellen eingebettet werden und der Durchgriff hier im Gesamtsystem aufgelöst wird.

Dieses Verfahren kann mit anderen Reduktionsverfahren kombiniert werden, um eine stationäre Genauigkeit zu erzielen. Bspw. kann die singuläre Pertubation auf ein balanciertes Modell angewendet werden [Hip92]. Hierdurch wird der stationäre Fehler beim balancierten Abschneiden nach Moore beseitigt.

In Kombination mit einer Modaltransformation eignet sich dieses Verfahren gut, gezielt hohe Eigenwerte aus dem Modell zu entfernen. Ausgangspunkt ist das lineare Zustandsraummodell in Modalform (A.4). Anstatt eine Dominanzanalyse durchzuführen, sortiert man die Gleichungen dieses Modells nach langsamen und schnellen Eigenwerten. Analog zur modalen Ordnungsreduktion erhält man eine Zustandsraumdarstellung der Form (A.8). Durch Einsetzen der Matrizen aus (A.8) in (A.26) erhält man das reduzierte Modell:

$$\begin{aligned} \dot{\underline{\tilde{x}}}_1 &= \underline{\mathbf{\Lambda}}_1 \, \underline{\tilde{x}}_1 + \underline{\mathbf{B}}_1^* \, \underline{u} \\ \underline{\tilde{y}} &= \underline{\mathbf{C}}_1^* \, \underline{\tilde{x}}_1 + \left(\underline{\mathbf{D}} - \underline{\mathbf{C}}_2^* \, \underline{\mathbf{\Lambda}}_2^{-1} \, \underline{\mathbf{B}}_2^* \right) \underline{u} \end{aligned} \tag{A.27}$$

Der stationäre Endwert wird bei dieser Kombination ausschließlich über die Durchgriffsmatrix berücksichtigt. Die Eigenwerte bleiben somit erhalten und die Stabilität des Systems ist gewährleistet.

A.2 Differentiation numerischer Lösungsverfahren

Verfahren zur Lösung verschiedener numerischer Probleme sind fester Bestandteil vieler Berechnungsprogramme. Die Algorithmische Differentiation der Lösungsalgorithmen ist aus zwei Gründen problematisch:

- Die Differentiation des Lösungsalgorithmus vervielfacht den Rechenaufwand.
- Die Konvergenz der Gradienten kann nicht in jedem Fall garantiert werden. Oft kommen iterative Methoden zum Einsatz, die bestimmte Fehlerschranken berücksichtigen. Diese Schranken beziehen sich jedoch nur auf den Funktionswert. Die Gradienten werden nicht berücksichtigt und können beim Abbruch der Iteration noch große Fehler aufweisen [GBC+93, Gri00].

Die Anwendung der AD auf den Lösungsalgorithmus kann vermieden werden, wenn die äußere Ableitung analytisch auf Basis der Lösung bestimmt werden kann. Mit Hilfe der Kettenregel kann diese in den Berechnungsablauf integriert werden. Im Folgenden werden die Ableitungen von Lösungsverfahren für lineare Gleichungssysteme, Eigenwertprobleme und nichtlineare Gleichungssysteme angegeben. Ein Auflistung weiterer Ableitungen findet sich in [Kas85].

Lineare Gleichungssysteme

Gesucht wird der Gradient $d\underline{x}/dp$, der sich aus der Lösung des linearen Gleichungssystems ergibt:

$$\underline{\mathbf{A}}(p)\underline{x} - \underline{b}(p) = \underline{0} \tag{A.28}$$

Die Lösung kann z. B. über die Inverse $\underline{\mathbf{A}}^{-1}$ angegeben werden:

$$\underline{x} = \underline{\mathbf{A}}^{-1}\underline{b} \tag{A.29}$$

Die Differentiation von Gleichung (A.28) nach p ergibt:

$$\frac{\partial}{\partial p}\underline{\mathbf{A}} \, \underline{x} + \underline{\mathbf{A}}\frac{\partial \underline{x}}{\partial p} - \frac{\partial \underline{b}}{\partial p} = \underline{0} \tag{A.30}$$

Durch Umstellen von (A.30) gelangt man zum gesuchten Gradienten $\nabla \underline{x}$:

$$\nabla \underline{x} = \frac{d\underline{x}}{dp} = \underline{\mathbf{A}}^{-1} \left(\frac{\partial \underline{b}}{\partial p} - \frac{\partial}{\partial p} \underline{\mathbf{A}}\, \underline{x} \right) \tag{A.31}$$

Die Inverse $\underline{\mathbf{A}}^{-1}$ tritt sowohl in Gleichung (A.29) als auch in (A.31) auf und muss nur einmal bestimmt werden. Anstelle die Inverse zu bestimmen wird die Lösung üblicherweise über eine LR-Zerlegung oder QR-Zerlegung ermittelt. Den Vektor $\underline{x}$ und den Gradienten $\nabla \underline{x}$ erhält man dabei durch Auswerten mit unterschiedlichen rechten Seiten.

Eigenwerte

Die Ableitung der Eigenwerte und Eigenvektoren nach einem Parameter p sind für eine Sensitivitätsanalyse, für eine Optimierung der Eigenwertlagen aber auch für den Einsatz der modalen Ordnungsreduktion (vgl. A.1.1) bei der hierarchischen Optimierung (Kapitel 7.4) interessant. Im Folgenden wird die Differentiation der Eigenwerte vorgestellt [Kas85, VMM07].

Die Eigenwerte $\underline{\lambda} \in \mathbb{Z}^n$ einer symmetrischen Matrix $\underline{\mathbf{A}}(p) \in \mathbb{Z}^{n \times n}$ sind über das Eigenwertproblem:

$$\underline{\mathbf{A}}\, \underline{\mathbf{U}} - \underline{\mathbf{U}}\, \underline{\mathbf{\Lambda}} = \underline{\mathbf{0}}, \quad \underline{\mathbf{V}}^\star \underline{\mathbf{A}} - \underline{\mathbf{\Lambda}}\, \underline{\mathbf{V}}^\star = \underline{\mathbf{0}} \tag{A.32}$$

definiert, mit den Rechtseigenvektoren $\underline{\mathbf{U}} \in \mathbb{Z}^{n \times n}$, den Linkseigenvektoren $\underline{\mathbf{V}} \in \mathbb{Z}^{n \times n}$ und der Eigenwertmatrix $\underline{\mathbf{\Lambda}} = \operatorname{diag} \underline{\lambda}$, $\underline{\mathbf{\Lambda}} \in \mathbb{Z}^{n \times n}$. Die Linkseigenvektoren werden so gewählt, dass gilt:

$$\underline{\mathbf{V}}^\star\, \underline{\mathbf{U}} = \underline{\mathbf{I}} \tag{A.33}$$

Die Differentiation von (A.32) nach p ergibt:

$$\frac{\partial}{\partial p} \underline{\mathbf{A}}\, \underline{\mathbf{U}} - \underline{\mathbf{U}}\, \frac{\partial}{\partial p} \underline{\mathbf{\Lambda}} = -\underline{\mathbf{A}}\, \frac{\partial}{\partial p} \underline{\mathbf{U}} + \frac{\partial}{\partial p} \underline{\mathbf{U}}\, \underline{\mathbf{\Lambda}} \tag{A.34}$$

und durch die linksseitige Multiplikation mit $\underline{\mathbf{V}}^\star$ erhält man:

$$\underline{\mathbf{V}}^\star \frac{\partial}{\partial p} \underline{\mathbf{A}}\, \underline{\mathbf{U}} - \underbrace{\underline{\mathbf{V}}^\star\, \underline{\mathbf{U}}}_{\underline{\mathbf{I}}}\, \frac{\partial}{\partial p} \underline{\mathbf{\Lambda}} = -\underline{\mathbf{V}}^\star \underline{\mathbf{A}}\, \frac{\partial}{\partial p} \underline{\mathbf{U}} + \underline{\mathbf{V}}^\star \frac{\partial}{\partial p} \underline{\mathbf{U}}\, \underline{\mathbf{\Lambda}} \tag{A.35}$$

Führt man die Substitution $\frac{\partial}{\partial p} \underline{\mathbf{U}} = \underline{\mathbf{U}}\, \underline{\mathbf{C}}$ auf (A.35) durch, so lassen sich die Terme der rechten Seite vereinfachen und es folgt:

$$\underline{\mathbf{V}}^\star \frac{\partial}{\partial p} \underline{\mathbf{A}}\, \underline{\mathbf{U}} - \frac{\partial}{\partial p} \underline{\mathbf{\Lambda}} = -\underline{\mathbf{\Lambda}}\, \underline{\mathbf{C}} + \underline{\mathbf{C}}\, \underline{\mathbf{\Lambda}} \tag{A.36}$$

Da die Diagonale von $-\underline{\mathbf{\Lambda}}\, \underline{\mathbf{C}} + \underline{\mathbf{C}}\, \underline{\mathbf{\Lambda}}$ verschwindet, können die Ableitungen der Eigenwerte über die Diagonalelemente:

$$\frac{\partial \lambda_i}{\partial p} = \left[\underline{\mathbf{V}}^\star \frac{\partial}{\partial p} \underline{\mathbf{A}}\, \underline{\mathbf{U}} \right]_{i,i} \tag{A.37}$$

bestimmt werden.

Die Ableitungen der Eigenvektoren können über die nicht verschwindenden Elemente des Ausdrucks $-\underline{\mathbf{\Lambda}}\, \underline{\mathbf{C}} + \underline{\mathbf{C}}\, \underline{\mathbf{\Lambda}}$ ermittelt werden. Da sich der Algorithmus hierfür etwas aufwändiger gestaltet, sei hierzu auf [VMM07] verwiesen.

Newton-Raphson-Verfahren

Das Newton-Raphson-Verfahren zählt zu den Standardverfahren der numerischen Mathematik. Es wird eingesetzt, um die Nullstelle $\underline{x}^\star$ eines nichtlinearen Gleichungssystems $\underline{f}:\mathbb{R}^n \to \mathbb{R}^n$ anzunähern:

$$\underline{f}(\underline{x}^\star) = \underline{0} \tag{A.38}$$

Das Newton-Verfahren arbeitet nach der Iterationsvorschrift:

$$\underline{x}_{k+1} = \underline{x}_k - \left(\nabla \underline{f}_k\right)^{-1} \underline{f}_k \tag{A.39}$$

mit

$$\underline{f}_k = \underline{f}(\underline{x}_k) \quad \text{und} \quad \nabla \underline{f}_k = \left.\frac{\partial \underline{f}}{\partial \underline{x}^{\mathrm{T}}}\right|_{\underline{x}_k} \tag{A.40}$$

Oftmals wird in (A.39) aus verschiedenen Gründen eine Abschätzung der Inversen eingesetzt, z. B. die sog. Pseudoinverse. Mit der Näherung der Inversen $(\nabla \underline{f})^+$ ergibt sich die Iterationsvorschrift:

$$\underline{x}_{k+1} = \underline{x}_k - \left(\nabla \underline{f}_k\right)^{+} \underline{f}_k \tag{A.41}$$

Im Folgenden wird die Differentiation der Funktion $\underline{f}(\underline{x}, p)$ nach dem Parameter p betrachtet. Nutz man die Tatsache aus, dass in der Nähe der Lösung $\underline{x}^\star$ die Differenz $\underline{x}_{k+1} - \underline{x}_k$ gegen Null konvergiert, so erhält man durch Differentiation von (A.41) nach p die Iterationsvorschrift:

$$\frac{\partial \underline{x}_{k+1}}{\partial p} = \frac{\partial \underline{x}_k}{\partial p} - \left(\nabla \underline{f}_k\right)^{+} \left(\nabla \underline{f}_k \frac{\partial \underline{x}_k}{\partial p} + \frac{\partial \underline{f}_k}{\partial p}\right) \tag{A.42}$$

zur Annäherung der Ableitung von $\underline{f}(\underline{x}^\star, p) = 0$.

Diese Iterationsvorschrift kann grundsätzlich parallel mit (A.41) ausgewertet werden. Hierbei ist zu beachten, dass neben $\underline{x}$ auch die Konvergenz der Ableitungen $\frac{\partial \underline{x}}{\partial p}$ überwacht werden muss. Im Allgemeinen werden zusätzliche Iterationsschritte benötigt, damit die Ableitungen die geforderte Genauigkeit erreichen.

Aus Rechenzeitgründen bietet sich jedoch eine Berechnung in zwei Phasen an (vgl. [Gri00]). In der ersten Phase wird das Gleichungssystem $\underline{f}(\underline{x}^\star, p) = 0$ gelöst, in der zweiten Phase werden aufbauend auf $\underline{x}^\star$ die Ableitungen $\partial \underline{x}^\star / \partial p$ angenähert:

$$\frac{\partial \underline{x}_{k+1}}{\partial p} = \frac{\partial \underline{x}_k}{\partial p} - \left(\nabla \underline{f}(\underline{x}^\star)\right)^{+} \left(\nabla \underline{f}(\underline{x}^\star) \frac{\partial \underline{x}_k}{\partial p} + \left.\frac{\partial \underline{f}}{\partial p}\right|_{\underline{x}^\star}\right) \tag{A.43}$$

Gilt $(\nabla \underline{f})^+ = (\nabla \underline{f})^{-1}$, so vereinfacht sich (A.43) zu:

$$\frac{\partial \underline{x}^\star}{\partial p} = -\left(\nabla \underline{f}(\underline{x}^\star)\right)^{-1} \left.\frac{\partial \underline{f}}{\partial p}\right|_{\underline{x}^\star} \tag{A.44}$$

Die gesuchten Ableitungen lassen sich hier in einem einzelnen zusätzlichen Berechnungsschritt ermitteln.

A.3 Modellreduktion des Unterflur-Federungsprüfstand

In diesem Abschnitt sind die Ergebnisse der Modellreduktion des Unterflur-Federungsprüfstandes exemplarisch anhand einzelner Konfigurationen aufgeführt.

Hydraulikzylinder

Die Linearisierung der Hydraulikzylinder mit den Parametern $K_p = 2000\,\frac{1}{\mathrm{m}}$ und $K_d = 0{,}2\,\frac{\mathrm{s}}{\mathrm{m}}$ führt zu den folgenden Zustandsraummatrizen:

$$\underline{\mathbf{A}}_{zyl} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ -1{,}421\cdot 10^{05} & -527{,}8 & 0 & 0 & -2{,}843\cdot 10^{06} \\ 6{,}205\cdot 10^{09} & 0 & 0 & 0 & 0 \\ -6{,}459\cdot 10^{09} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1000 \end{bmatrix}, \quad \underline{\mathbf{B}}_{zyl} = \begin{bmatrix} 0 & 0 & 0 \\ 3{,}127\cdot 10^{07} & -3{,}127\cdot 10^{07} & 0 \\ 0 & 0 & -2{,}010\cdot 10^{10} \\ 0 & 0 & 1{,}060\cdot 10^{10} \\ 1000 & -1000 & 0 \end{bmatrix}$$

$$\underline{\mathbf{C}}_{zyl} = \begin{bmatrix} 0 & 0 & -8{,}041\cdot 10^{-04} & 4{,}244\cdot 10^{-04} & 0 \\ 3536 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \underline{\mathbf{D}}_{zyl} = \begin{bmatrix} 0 & 0 & 5000 \\ 0 & 0 & 0 \end{bmatrix} \tag{A.45}$$

Die Transformation in eine Minimalrealisierung resultiert in den reduzierten Zustandsraummatrizen:

$$\underline{\mathbf{A}}_{zyl,r} = \begin{bmatrix} -1000 & 0 & 0 & 0 \\ 284{,}3 & -527{,}8 & -14{,}21 & 0 \\ 0 & 1{,}0\cdot 10^{04} & 0 & 0 \\ 0 & 0 & 850{,}2 & 0 \end{bmatrix}, \quad \underline{\mathbf{B}}_{zyl,r} = \begin{bmatrix} -1000 & 1000 & 0 \\ 3127 & -3127 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -2272 \end{bmatrix}$$

$$\underline{\mathbf{C}}_{zyl,r} = \begin{bmatrix} 0 & 0 & 0 & -9092 \\ 0 & 0 & 3536 & 0 \end{bmatrix}, \quad \underline{\mathbf{D}}_{zyl,r} = \begin{bmatrix} 0 & 0 & 5000 \\ 0 & 0 & 0 \end{bmatrix} \tag{A.46}$$

Aktorgruppen

Die Linearisierung des Basismodells der Aktorgruppen mit den identischen Einstellungen $K_p = 2000\,\frac{1}{\mathrm{m}}$ und $K_d = 0{,}2\,\frac{s}{\mathrm{m}}$ für die drei Servozylinder resultiert in den folgenden Zustandsraummatrizen:

$$\underline{\mathbf{A}}_{ag} = \begin{bmatrix}
-1000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -95{,}39 & 0 & -84{,}97 & 0 & -4{,}999 & 0 \\
284{,}3 & -527{,}8 & -14{,}21 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 298{,}3 & 0 & 265{,}7 & 0 & 15{,}63 & 0 \\
0 & 1{,}000\cdot 10^{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 850{,}2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7{,}927 & 216{,}8 & 9{,}734 & 193{,}1 & 15{,}11 & 11{,}36 \\
0 & 0 & 0 & 0 & -1000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 120{,}4 & 0 & 68{,}43 & 0 & 146{,}6 & 0 \\
0 & 0 & 0 & 0 & 284{,}3 & -527{,}8 & -14{,}21 & 0 & 0 & 0 & 0 & 0 & -376{,}5 & 0 & -214{,}0 & 0 & -458{,}5 & 0 \\
0 & 0 & 0 & 0 & 0 & 1{,}000\cdot 10^{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 850{,}2 & 0 & 0 & 0 & 0 & 0 & -48{,}16 & -273{,}6 & -64{,}16 & -155{,}5 & -58{,}63 & -333{,}2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1000 & 0 & 0 & 0 & 31{,}11 & 0 & 68{,}43 & 0 & 146{,}6 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 284{,}3 & -527{,}8 & -14{,}21 & 0 & -97{,}27 & 0 & -214{,}0 & 0 & -458{,}5 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1{,}000\cdot 10^{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 850{,}2 & 0 & -55{,}11 & -70{,}69 & -65{,}57 & -155{,}5 & -19{,}51 & -333{,}2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -8{,}159\cdot 10^{4} & 0 & 0 & 0 & 8{,}017\cdot 10^{4} & 0 & 0 & 0 & -4{,}215\cdot 10^{4} & -6{,}589\cdot 10^{5} & -1{,}710\cdot 10^{4} & -3{,}937\cdot 10^{5} & -1{,}001\cdot 10^{4} & -4{,}125\cdot 10^{5} & -8304 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & -8{,}597\cdot 10^{4} & 0 & 0 & 0 & -2{,}433\cdot 10^{4} & 0 & 0 & 0 & 2{,}555\cdot 10^{4} & -2{,}618\cdot 10^{5} & -6556 & -3{,}062\cdot 10^{5} & -7981 & -3{,}166\cdot 10^{5} & -4459 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1{,}000 \\
0 & 0 & 0 & 6{,}935\cdot 10^{4} & 0 & 0 & 0 & -8086 & 0 & 0 & 0 & 2{,}026\cdot 10^{4} & 3{,}110\cdot 10^{5} & 7704 & 2{,}306\cdot 10^{5} & 5643 & 2{,}382\cdot 10^{5} & 2220
\end{bmatrix}$$

$$\underline{\mathbf{B}}_{ag} = \begin{bmatrix}
0 & -2{,}992\cdot 10^{-3} & -4{,}364\cdot 10^{-3} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 9{,}355\cdot 10^{-3} & 1{,}364\cdot 10^{-2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 2{,}716\cdot 10^{-3} & -9{,}033\cdot 10^{-3} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -8{,}493\cdot 10^{-3} & 2{,}824\cdot 10^{-2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -8{,}633\cdot 10^{-3} & -1{,}503\cdot 10^{-2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 2{,}699\cdot 10^{-2} & 4{,}700\cdot 10^{-2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1{,}925\cdot 10^{6} & -1{,}347\cdot 10^{6} & 0 & 2{,}459\cdot 10^{4} & -1{,}625\cdot 10^{4} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 7{,}111\cdot 10^{4} & -1{,}071\cdot 10^{6} & 0 & 189{,}7 & -1{,}337\cdot 10^{4} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -7{,}028\cdot 10^{5} & 7{,}976\cdot 10^{5} & 0 & -8759 & 9759
\end{bmatrix}$$

$$\underline{\mathbf{C}}_{ag} = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1{,}935\cdot 10^{4} & -239{,}6 & -9943 & -117{,}2 & -1{,}047\cdot 10^{4} & -123{,}4 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2{,}126\cdot 10^{4} & 257{,}5 & 1{,}878\cdot 10^{4} & 229{,}5 & 1{,}981\cdot 10^{4} & 241{,}5 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3536 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 3536 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 3536 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

$$\underline{\mathbf{D}}_{ag} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6{,}323 \cdot 10^{4} & -3{,}353 \cdot 10^{4} & 0 & 815{,}0 & -400{,}0 \\ 0 & 0 & 0 & 0 & -3{,}353 \cdot 10^{4} & 6{,}344 \cdot 10^{4} & 0 & -400{,}0 & 783{,}0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.47}$$

Die Systemmatrizen der Servozylinder $\underline{\mathbf{A}}_{zyl,r}$ können in der oberen linken Hälfte der Systemmatrix $\underline{\mathbf{A}}_{ag}$ des linearisierten Modells wiedergefunden werden. Die Reduktion des Zustandsraummodells ergibt:

$$\underline{\mathbf{A}}_{ag,r} = \begin{bmatrix} -1630 & -2816 & 1712 & 1564 & 241{,}0 & 98{,}14 & -53{,}31 & -9{,}649 & 338{,}5 & -376{,}7 & 27{,}08 & 62{,}72 \\ 1344 & -1630 & -599{,}2 & -1114 & -141{,}0 & -49{,}14 & 169{,}7 & 17{,}38 & -120{,}8 & 274{,}7 & 4{,}607 & 72{,}77 \\ 0 & 0 & -2760 & -1218 & -157{,}2 & -129{,}5 & 155{,}6 & 74{,}57 & -190{,}2 & 344{,}0 & -11{,}18 & -2{,}721 \\ 0 & 0 & 0 & -1494 & -321{,}3 & -147{,}9 & 160{,}9 & -45{,}59 & -431{,}3 & 610{,}8 & -51{,}04 & -2{,}927 \\ 0 & 0 & 0 & 0 & -206{,}6 & -138{,}3 & -33{,}99 & -16{,}25 & -113{,}7 & -54{,}28 & 218{,}2 & -111{,}0 \\ 0 & 0 & 0 & 0 & 362{,}7 & -206{,}6 & 39{,}80 & -30{,}61 & -72{,}76 & 178{,}6 & -124{,}2 & 47{,}28 \\ 0 & 0 & 0 & 0 & 0 & 0 & -207{,}4 & 125{,}1 & 49{,}39 & 18{,}09 & -75{,}09 & -265{,}5 \\ 0 & 0 & 0 & 0 & 0 & 0 & -388{,}8 & -207{,}4 & -28{,}46 & 100{,}4 & -70{,}39 & -123{,}6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -174{,}9 & -62{,}50 & -26{,}67 & -9{,}981 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 253{,}5 & -174{,}9 & 182{,}4 & 14{,}09 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -113{,}9 & -4{,}661 \cdot 10^{-02} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -115{,}3 \end{bmatrix}$$

$$\underline{\mathbf{B}}_{ag,r} = \begin{bmatrix} 0 & 5{,}256 \cdot 10^{-03} & 4{,}234 \cdot 10^{-03} & 0 & -2023 & -1207 & 0 & -27{,}68 & -15{,}71 \\ 0 & 2{,}795 \cdot 10^{-03} & -1{,}612 \cdot 10^{-03} & 0 & 3289 & -7391 & 0 & 38{,}42 & -91{,}41 \\ 0 & -2{,}867 \cdot 10^{-03} & -2{,}321 \cdot 10^{-03} & 0 & 1{,}203 \cdot 10^{04} & -8596 & 0 & 153{,}5 & -103{,}8 \\ 0 & -1{,}839 \cdot 10^{-03} & -6{,}703 \cdot 10^{-03} & 0 & 761{,}9 & 33{,}52 & 0 & 10{,}13 & 0{,}6502 \\ 0 & -1{,}515 \cdot 10^{-03} & 2{,}390 \cdot 10^{-02} & 0 & 25{,}19 & -100{,}4 & 0 & 0{,}2635 & -1{,}248 \\ 0 & -6{,}368 \cdot 10^{-04} & 7{,}489 \cdot 10^{-03} & 0 & 35{,}81 & -120{,}8 & 0 & 0{,}3900 & -1{,}500 \\ 0 & -2{,}338 \cdot 10^{-02} & -7{,}486 \cdot 10^{-03} & 0 & -49{,}62 & 61{,}60 & 0 & -0{,}6148 & 0{,}7552 \\ 0 & 5{,}346 \cdot 10^{-03} & 1{,}266 \cdot 10^{-03} & 0 & 8{,}212 & -81{,}47 & 0 & 5{,}160 \cdot 10^{-02} & -1{,}016 \\ 0 & -2{,}621 \cdot 10^{-03} & -6{,}167 \cdot 10^{-03} & 0 & 102{,}4 & -353{,}1 & 0 & 1{,}109 & -4{,}384 \\ 0 & 6{,}731 \cdot 10^{-03} & 2{,}317 \cdot 10^{-02} & 0 & -64{,}81 & 176{,}2 & 0 & -0{,}7356 & 2{,}183 \\ 0 & -7{,}284 \cdot 10^{-03} & -2{,}808 \cdot 10^{-02} & 0 & 3{,}159 & -8{,}510 & 0 & 3{,}590 \cdot 10^{-02} & -0{,}1054 \\ 0 & -1{,}878 \cdot 10^{-02} & 3{,}870 \cdot 10^{-04} & 0 & -21{,}11 & 6{,}259 & 0 & -0{,}2756 & 7{,}186 \cdot 10^{-02} \end{bmatrix}$$

$$\underline{\mathbf{C}}_{ag,r} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3861 & 6624 & 4676 & -29{,}98 & 3032 & -2551 & 6938 & 7672 & -38{,}77 & -857{,}1 & -26{,}80 & -4211 \\ 4864 & -579{,}2 & -9503 & -3860 & -5912 & 4762 & 525{,}1 & 907{,}2 & -1682 & -1205 & -2817 & 1152 \\ 2106 & -2067 & -2041 & 1{,}162 \cdot 10^{04} & -1{,}803 \cdot 10^{04} & 5{,}041 \cdot 10^{04} & 1{,}695 \cdot 10^{04} & 6{,}104 \cdot 10^{04} & -1{,}171 \cdot 10^{04} & -6080 & -8557 & -3705 \\ 946{,}2 & -3252 & 258{,}9 & -1235 & -3455 & 1{,}456 \cdot 10^{04} & 5064 & 2{,}133 \cdot 10^{04} & 1337 & -2749 & -2521 & -1391 \\ -1451 & 4226 & -2710 & 8115 & -1{,}985 \cdot 10^{04} & 5{,}255 \cdot 10^{04} & -4103 & -1{,}614 \cdot 10^{04} & -4378 & -2214 & -4727 & 3375 \end{bmatrix}$$

$$\underline{\mathbf{D}}_{ag,r} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3{,}933 \cdot 10^{-03} & 6{,}704 \cdot 10^{-04} & 0 & 2{,}909 \cdot 10^{04} & -2131 & 0 & 384{,}3 & -17{,}82 \\ 0 & -7{,}055 \cdot 10^{-03} & -1{,}312 \cdot 10^{-03} & 0 & 2469 & 2{,}741 \cdot 10^{04} & 0 & 52{,}02 & 343{,}4 \\ 0 & -8{,}249 \cdot 10^{-03} & 9{,}242 \cdot 10^{-04} & 0 & 3356 & -9076 & 0 & 38{,}12 & -112{,}5 \\ 0 & -1{,}034 \cdot 10^{-03} & -4{,}070 \cdot 10^{-03} & 0 & 1205 & -3213 & 0 & 13{,}72 & -39{,}81 \\ 0 & 6{,}654 \cdot 10^{-03} & 1{,}293 \cdot 10^{-03} & 0 & 3475 & 4391 & 0 & 49{,}18 & 55{,}96 \end{bmatrix}$$

(A.48)

Literaturverzeichnis

[ADG89] Acker, B., W. Darenberg und H. Gall: *Aktive Feder für Personenwagen.* O+P, Ölhydraulik und Pneumatik, 33(11), 1989.

[ADG+08] Adelt, P., J. Donoth, J. Gausemeier, J. Geisler, S. Henkler, S. Kahl, B. Klöpper, A. Krupp, E. Münch, S. Oberthür, C. Paiz, H. Podlogar, M. Porrmann, R. Radkowski, C. Romaus, A. Schmidt, B. Schulz, H. Vöcking, U. Witkowski, K. Witting und O. Znamenshchykov: *Selbstoptimierende Systeme des Maschinenbaus – Definitionen, Anwendungen, Konzepte.*, Band 234 der Reihe *HNI-Verlagsschriftenreihe, Paderborn.* HNI-Verlagsschriftenreihe, Paderborn, 2008.

[AEH+11] Adelt, Philipp, Natascha Esau, Christian Hölscher, Bernd Kleinjohann, Lisa Kleinjohann, Martin Krüger und Detmar Zimmer: *Hybrid Planning for Self-Optimization in Railbound Mechatronic Systems.* Intelligent Mechatronics, Seiten 169 – 194, Februar 2011.

[Bae86] Bae, D.-S.: *A Recursive Formulation for Constraint Mechanical System Dynamics.* PhD thesis, University of Iowa, Iowa, 1986.

[BB00] Bischof, C. und M. Bücker: *Computing Derivatives of Computer Programs.* In: Grotendorst, J. (Herausgeber): *Modern Methods and Algorithms of Quantum Chemistry*, Band 3 der Reihe *NIC Series*, Seiten 315–327, Jülich, 2000. John von Neumann Institute for Computing.

[Bri66] Bristol, E.: *On a new measure of interaction for multivariable process control.* IEEE Transactions on Automatic Control, 11(1):133–134, 1966.

[BRM97] Bischof, C., L. Roh und A. Mauer: *ADIC – An Extensible Automatic Differentiation Tool for ANSI-C.* In: *Software-Practice and Experience*, 1997.

[Bro87] *Brockhaus. Die Enzyklopädie.* Bibliographisches Institut & F. A. Brockhaus AG, 1987.

[BSKF06] Böcker, Joachim, Bernd Schulz, Tobias Knoke und Norbert Fröhleke: *Self-Optimization as a Framework for Advanced Control Systems.* In: *The 32nd Annual Conference of the IEEE Industrial Electronics Society (IECON'06)*, Paris, France, 2006.

[BSMM95] Bronstein, Ilja N., Konstantin A. Semendjajew, Gerhard Musiol und Heiner Mühlig: *Bronštejn, Il'ja N.: Taschenbuch der Mathematik.* Verlag Harri Deutsch, Thun, Frankfurt am Main, 1995.

[CDB80] Conte, Samuel Daniel und Carl W. De Boor: *Elementary Numerical Analysis: An Algorithmic Approach*. McGraw-Hill Higher Education, 1980.

[CFG+02] Corliss, G., C. Faure, A. Griewank, L. Hascoët und U. Naumann (Herausgeber): *Automatic Differentiation or Algorithms – From Simulation to Optimization*. Springer-Verlag, New York, USA, 2002.

[CL02] Coello, C.A.C. und M.S. Lechunga: *MOPSO: A Proposal for Multi Objective Particle Swarm Optimization*. In: *IEEE World Congress on Computational Intelligence*, Hawaii, 2002.

[ÇNM+07] Çinkaya, Hüseyin, Rene Nölle, Eckehard Münch, Ansgar Trächtler und Stephanie Toepper: *Selbstkalibrierung des 6 DOF Parallelroboters TRIPLANAR durch Identifikation der Geometrieparameter mit einem einfachen Messverfahren*. In: *Mechatronik 2007: Innovative Produktentwicklung*, Wiesloch, 2007.

[ÇNMT07] Çinkaya, Hüseyin, Rene Nölle, Eckehard Münch und Ansgar Trächtler: *Self-Calibration of the 6-dof Parallel Robot TriPlanar by Identification of the Geometry Parameters*. In: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2007)*, Zürich, 2007.

[DB05] Dorf, R. C. und R. H. Bishop: *Modern Control Systems*. Pearson Educational, Inc., Upper Saddle River, New Jersey, 10. Auflage, 2005.

[DD96a] Das, I. und J. Dennis: *A Closer Look at Drawbacks of Minimizing Weighted Sums of Objectives for Pareto Set Generation in Multicriteria Optimzation Problems*. Technischer Bericht 96-36, Dept. of Computational and Applied Mathematics, Rice University, Houston, Texas, 1996.

[DD96b] Das, I. und J. Dennis: *Normal-boundary intersection: A new method for generating Pareto optimal points in multicriteria optimzation problems*. Technischer Bericht 96-11, Dept. of Computational and Applied Mathematics, Rice University, Houston, Texas, 1996.

[Deb01] Deb, Kalyanmoy: *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, USA, 2001.

[Del08] Dell'Aere, Alessandro: *Numerical Methods for the Solution of Bi-Level Multi-Objective Optimization Problems*. Dissertation, Universität Paderborn, Universitätsbibliothek Paderborn, 2008.

[Dep06] Deppe, Markus: *Verteilte Online-Mehrziel-Parameter-Optimierung in mechtronischen Systemen*. Dissertation, Universität Paderborn, Buchbinderei Wolfram Schmidt, Braunschweig, 2006.

[DG97] Dorigo, M. und L. M. Gambardella: *Ant Colonies for Traveling Salesman Problem*, Band 43. J. Biosystems, 1997.

[DHK+09] Dell'Aere, A., M. Hirsch, B. Klöpper, M. Koester, A. Krupp, M. Krüger, T. Müller, S. Oberthür, S. Pook, C. Priesterjahn, C. Romaus, A. Schmidt, C. Sondermann-Wölke, M. Tichy, H. Vöcking und D. Zimmmer: *Verlässlichkeit selbstoptimierender Systeme – Potenziale nutzen und Risiken vermeiden*, Band 235 der Reihe *HNI-Verlagsschriftenreihe, Paderborn*. HNI-Verlagsschriftenreihe, Paderborn, 2009.

[Dig04] Dignath, Florian: *Zur Optimierung mechatronischer Systeme mit nichtdifferenzierbaren Kriterien*. Dissertation, Universität Stuttgart, VDI-Fortschritt-Berichte, Reihe 8, Nr. 1031, VDI-Verlag, 2004.

[Dix71] Dixon, L. C. W.: *The choice of step length, a crucial factor in the performance of variable metric algorithms*. In: Lootsma, F. A. (Herausgeber): *Numerical Methods for Nonlinear Optimization*, Seiten 149–170. Academic Press, 1971.

[DP04] Dittmar, Rainer und Bernd-Markus Pfeiffer: *Modellbasierte prädiktive Regelung – Eine Einführung für Ingenieure*. Oldenbourg Wissenschaftsverlag GmbH, München, 2004.

[Du06] Du, Yu: *Zur Optimierung des dynamischen Verhaltens von Gesamtfahrzeugen mit mechatronischen Komponenten*. Dissertation, TU Chemnitz, Shaker Verlag, Aachen, 2006.

[Dym12] *Dymola – Multi-Engineering Modeling and Simulation*. `http://www.3ds.com/products/catia/portfolio/dymola`, Stand 06. März 2012.

[EA06] El-Aasar, Alaa: *Design of a Self-Tuning System with Online-Identification for a Hydraulic Test-Bench*. Masterarbeit, Universität Paderborn, 2006.

[EB99] Eberhard, Peter und Christian H. Bischof: *Automatic differentiation of numerical integration algorithms*. Mathamatics of Computation, 68(226):717–731, 1999.

[Ebe96] Eberhard, Peter: *Zur Mehrkriterienoptimierung von Mehrkörpersystemen*. Dissertation, Universität Stuttgart, VDI-Fortschritt-Berichte, Reihe 11, Nr. 227, VDI-Verlag, 1996.

[EHO01] Ettingshausen, Clemens, Thorsten Hestermeyer und Stefan Otto: *Aktive Spurführung und Lenkung von Schienenfahrzeugen*. In: *6. Magdeburger Maschinenbautage, Intelligente technische Systeme und Prozesse – Grundlagen, Entwurf, Realisierung*, 2001.

[FF93] Fonseca, C. M. und P. J. Fleming: *Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization*. In: Forrest, S. (Herausgeber): *Proceedings of the Fifth International Conference on Genetic Algorithms*, Seiten 416–423, San Mateo, California, 1993. Morgan Kaufman Publishers.

[FF98] Fonseca, Carlos M. und Peter J. Fleming: *Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms I: A Unified Formulation.* IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, 28:26–37, 1998.

[FGK+04] Frank, Ursula, Holger Giese, Florian Klein, Oliver Oberschelp, Andreas Schmidt, Bernd Schulz, Henner Vöcking und Katrin Witting: *Selbstoptimierende Systeme des Maschinenbaus – Definitionen und Konzepte.*, Band 155 der Reihe *HNI-Verlagsschriftenreihe, Paderborn.* Heinz Nixdorf Institut, Universität Paderborn, Paderborn, 2004.

[FGW02] Forsgren, Anders, Philip E. Gill und Margaret H. Wright: *Interior Methods for Nonlinear Optimization.* SIAM Review, 44(4):525–597, 2002.

[Föl94] Föllinger, Otto: *Regelungstechnik, Einführung in die Methoden und ihre Anwendungen.* Hüthig GmbH, Heidelberg, 1994.

[FPL05] Fleming, Peter J., Robin C. Purshouse und Robert J. Lygoe: *Many-Objective Optimization: An Engineering Design Perspective.* In: *Third International Conference on Evolutionary MultiCriteria Optimisation (EMO 2005)*, Seiten 14–32, Mexico, 2005.

[Fri03] Fritzson, Peter: *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1.* Wiley-IEEE Computer Society Press, 2003.

[GBC+93] Griewank, A., C. Bischof, G. Corliss, A. Carle und K. Willamson: *Derivative convergence of iterative equation solvers.* Optimization Methods and Software, 2:321–355, 1993.

[GDT+06] Galassi, M., J. Davies, J. Theiler, B. Gough, G. Jungman, M. Booth und F. Rossi: *GNU Scientific Library Reference Manual – Revised Second Edition (v1.8).* Network Theory Limited, Bristol, 2006.

[Gei05] Geisler, Jens: *Auslegung und Implementierung der verteilten Aktor- und Aufbauregelung für ein aktiv gefedertes Schienenfahrzeug.* Masterarbeit, Universität Paderborn, 2005.

[Gem74] Gembicki, F. W.: *Vector Optimization for Control with Performance and Parameter Sensitivity Indices.* PhD thesis, Case Western Reserve University, Cleveland, Ohio, 1974.

[GJU96] Griewank, A., D. Juedes und J. Utke: *ADOL-C: a package for the automatic differentiation of algorithms written in C/C++.* ACM Transactions on Mathematical Software, 22:131–167, 1996.

[GME05] Gielen, Georges G. E., Trent McConaghy und Tom Eeckelaert: *Performance Space Modeling for Hierarchical Synthesis of Analog Integrated Circuits.* In: *IEEE Design Automation Conference (DAC 2005)*, Seiten 881–886, Anaheim, California, USA, 2005.

[GMW81] Gill, Philip E., Walter Murray und Margaret H. Wright: *Practical Optimization.* Academic Press, Inc.; London, New York, 1981.

[GN90] Göpfert, A. und R. Nehse: *Vektoroptimierung.* BSB Teubner Verlagsgesellschaft, Leipzig, 1990.

[Gri00] Griewank, A.: *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation.* SIAM: The Society for Industrial and Applied Mathematics, Philadelphia, 2000.

[Gut91] Guth, R.: *Stationär genaue Ordnungsreduktion balancierter Zustandsraummodelle.* at – Automatisierungstechnik, 39(8):286–290, 1991.

[GWTD08a] Geisler, Jens, Katrin Witting, Ansgar Trächtler und Michael Dellnitz: *Multiobjective Optimization of Control Trajectories for the Guidance of a Rail-bound Vehicle.* In: *17th World Congress, The International Federation of Automatic Control (IFAC)*, Seoul, Korea, July 2008.

[GWTD08b] Geisler, Jens, Katrin Witting, Ansgar Trächtler und Michael Dellnitz: *Self-Optimization of the Guidance Module of a Rail-bound Vehicle.* In: *7th International Heinz Nixdorf Symposium: Self-optimizing Mechatronic Systems: Designing the Future*, Paderborn, February 2008.

[Hah99] Hahn, Martin: *OMD – Ein Objektmodell für den Mechatronikentwurf.* Dissertation, Universität Paderborn, VDI-Fortschritt-Berichte, Reihe 20, Nr. 299, VDI-Verlag, 1999.

[Har02] Harrer, Heinrich: *Ordnungsreduktion. Vom komplexen Strukturmodell zur vereinfachten Beschreibung technischer Systeme.* Richard Pflaum Verlag, München, 2002.

[HD09] Harmse, M. und R. Dittmar: *Robuste Einstellung dezentraler PID-Regler in einer Mehrgrößenumgebung – Modellbasiert und praxisnah zum optimalen Ergebnis.* atp edition – Auomatisierungstechnische Praxis, 51(12):68–78, 2009.

[Hen97] Henrichfreise, H.: *Prototyping of a LQG Compensator for a Compliant Positioning System with Friction.* In: Gausemeier, J. (Herausgeber): *1. Workshop TRANSMECHATRONIK – Entwicklung und Transfer von Entwicklungssystemen der Mechatronik*, Band 23, Paderborn, 1997. HNI-Verlagsschriftenreihe.

[Hen03] Henke, Markus: *Antrieb mit doppelgespeistem Linearmotor für ein spurgeführtes Bahnfahrzeug.* Dissertation, Universität Paderborn, VDI-Fortschritt-Berichte, Reihe 12, Nr. 533, VDI-Verlag, 2003.

[Hes06] Hestermeyer, Thorsten: *Strukturierte Entwicklung der Informationsverarbeitung für die aktive Federung eines Schienenfahrzeugs.* Dissertation, Universität Paderborn, Verlag Dr. Hut, München, 2006.

[HHR94] Hahn, M., C. Homburg und J. Richert: *DSS-DSL-DSC – Die drei Ebenen einer Modellbeschreibungssprache für mechatronische Systeme.* In: *ASIM - 9. Symposium Simulationstechnik*, Braunschweig, 1994.

[Hil01] Hillermeier, Claus: *Nonlinear Multiobjective Optimization, A Generalized Homotopy Approach.* Birkhäuser Verlag, Basel, 2001.

[Hip92] Hippe, P.: *Balancierte Realisierung und stationär genaue Modelle.* at – Automatisierungstechnik, 40(7):268–270, 1992.

[HMS04] Hestermeyer, Thorsten, Eckehard Münch und Erika Schäfer: *Model-Based Online Parameter Optimization.* In: *3rd IFAC Symposium on Mechatronic Systems*, Sydney, Australia, 2004.

[Hol08] Holoborodko, Pavel: *Numerical Differentiation.* `http://www.holoborodko.com/pavel/numerical-methods/numerical-derivative/`, Stand 26. September 2008.

[HSNL97] Honekamp, Uwe, Ralf Stolpe, Rolf Naumann und Joachim Lückel: *Structuring Approach for Complex Mechatronic Systems.* In: *30th ISATA Conference on Mechatronics (ISATA 97)*, Florence, Italy, 1997.

[Hua08] Huang, Song: *Hierarchische Optimierung der Aufbaudynamik eines aktiven Federungssystems bei verschiedenen Anregungen.* Bachelorarbeit, Universität Paderborn, 2008.

[ILM92] Isermann, Rolf, Karl-Heinz Lachmann und Drago Matko: *Adaptive Control Systems.* Prentice Hall, New York, 1992.

[Ise92a] Isermann, Rolf: *Identifikation dynamischer Systeme: Besondere Methoden, Anwendungen.* Springer-Verlag, Berlin, 2. Auflage, 1992.

[Ise92b] Isermann, Rolf: *Identifikation dynamischer Systeme: Grundlegende Methoden.* Springer-Verlag, Berlin, 2. Auflage, 1992.

[iXt01a] iXtronics GmbH: *CAMeL-View Reference Guide*, 2001.

[iXt01b] iXtronics GmbH: *CAMeL-View User Guide*, 2001.

[JBL+02] Joos, H.-D., J. Bals, G. Looye, K. Schnepper und A. Varga: *A multi-objective optimisation based software environment for control systems design.* In: *Proceedings of 2002 IEEE International Conference on Control Applications and International Symposium on Computere Aided Control Systems Design, CCA/CACSD*, Glasgow, Scotland, U.K., 2002.

[Joo03] Joos, Hans-Dieter: *MOPS – Multi-Objective Parameter Synthesis, User's Guide V3.0.* DLR Deutsches Zentrum für Luft- und Raumfahrt, Institut für Robotik und Mechatronik, Oberpfaffenhofen, 2003.

[Jun97] Junker, Frank: *Eine modular-hierarchisch organsisierte Modellbildung machanischer Komponenten der Mechanik.* Dissertation, Universität Paderborn, VDI-Fortschritt-Berichte, Reihe 20, Nr. 261, VDI-Verlag, 1997.

[Kar78] Karnopp, C. D.: *Are Active Suspensions really Necessary?* ASME Publication, 78-WA/DE-12, 1978.

[Kas85] Kasper, Roland: *Entwicklung und Erprobung eines instrumentellen Verfahrens zum Entwurf von Mehrgrößenregelungen.* Dissertation, Universität Paderborn, VDI-Fortschritt-Berichte, Reihe 8, Nr. 90, VDI-Verlag, 1985.

[Klö09] Klöpper, Benjamin: *Ein Beitrag zur Verhaltensplanung für interagierende intelligente mechatronische Systeme in nicht-deterministischen Umgebungen.* Dissertation, Universität Paderborn, Band 253, HNI-Verlagsschriftenreihe, Paderborn, 2009.

[KLJS90] Kasper, R., J. Lückel, K.-P. Jäker und J. Schröder: *CACE tool for multi-input, multi-output systems using a new vector optimization method.* International Journal of Control, 51(5):963–993, 1990.

[Koc05] Koch, Thorsten: *Integration von Konstruktion und mechatronischer Komposition während des Entwurfs mechatronischer Systeme am Beispiel eines integrierten Radmoduls.* Dissertation, Universität Paderborn, VDI-Fortschritt-Berichte, Reihe 208, Nr. 401, VDI-Verlag, 2005.

[KRB+06] Knoke, T., C. Romaus, J. Böcker, A. Dell'Aere und K. Witting: *Energy Management for an Onboard Storage System Based on Multi-Objective Optimization.* In: *International Electronics Conference (IECON)*, Seiten 4677–4682, Paris, 2006.

[KS79] Kreisselmeier, G. und R. Steinhauser: *Systematic Control Design by Optimising a vector performance index.* In: *IFAC Symposium on Computer Aided Design of Control Systems*, Seiten 113–117, Zürich, 1979. Pergamon Press.

[KST10] Krüger, Martin, Ingo Scharfenbaum und Ansgar Trächtler: *Parametrische Modellreduktion in hierarchisch modellierten selbstoptimierenden Systemen.* In: *7. Paderborner Workshop Entwurf mechatronischer Systeme (EMS)*, Paderborn, 2010. HNI-Verlagschriftenreihe.

[KT51] Kuhn, H. und A. Tucker: *Nonlinear Programming.* In: *Proceedings of 2nd Berkeley Symposium*, Seiten 481–492, Berkeley, 1951. University of California Press.

[Lüc92] Lückel, Joachim: *The Concept of Mechatronic Function Modules applied to Compound Active Suspension Systems.* In: *Research Issues in Automotive Integrated Chassis Control Systems*, International Symposium for Vehicle System Dynamics, Herbertov, CSFR, 1992.

[Löf08] Löffler, Alexander: *Hierarchische Mehrzieloptimierung mechatronischer Systeme am Beispiel des SUrF-Prüfstandes.* Diplomarbeit, Universität Paderborn, 2008.

[LG99] Li, Hong und Roger M. Goodall: *Linear and non-linear skyhook damping control laws for active railway suspensions.* Control Engineering Practice, 7:843–850, 1999.

[LGH+09] Lückel, Joachim, Horst Grotstollen, Markus Henke, Thorsten Hestermeyer und Xiaobo Liu-Henke: *RailCab System: Engineering Aspects.* In: Maier, Giulio, Jean Salençon, Wilhelm Schneider, Bernhard Schrefler, Paolo Serafini und Werner Schiehlen (Herausgeber): *Dynamical Analysis of Vehicle Systems*, Band 497 der Reihe *CISM International Centre for Mechanical Sciences*, Seiten 237–281. Springer Vienna, 2009.

[LH05] Liu-Henke, Xiabo: *Mechatronische Entwicklung der aktiven Feder-/Neigetechnik für das Schienenfahrzeug RailCab.* Dissertation, Universität Paderborn, VDI-Fortschritt-Berichte, Reihe 12, Nr. 589, VDI-Verlag, 2005.

[LHL01] Lückel, Joachim, Thorsten Hestermeyer und Xiaobo Liu-Henke: *Generalization of the Cascade Principle in View of a Structured Form of Mechatronic Systems.* In: *International Conference on Advanced Intelligent Mechatronics (AIM 2001)*, Como, Italy, 2001. IEEE/ASME.

[LHPW87] Laub, Alan J., Michael T. Heatch, Chris C. Paige und Robert C. Ward: *Computation of System Balancing Transformations and Other Applications of Simultaneous Diagonalization Algorithms.* IEEE Transactions on Automatic Control, AC-32(2):115–122, 1987.

[Lit79a] Litz, Lothar: *Ordnungsreduktion linearer Zustandsraummodelle durch Beibehaltung der dominanten Eigenbewegung.* Regelungstechnik, 27:80–86, 1979.

[Lit79b] Litz, Lothar: *Reduktion der Ordnung linearer Zustandsraummodelle mittels modaler Verfahren.* HochschulSammlung Ingenieurwissenschaft Datenverarbeitung, Band 4, Hochschul-Verlag, Stuttgart, 1979.

[LK85] Lückel, Joachim und Roland Kasper: *Optimization of the disturbance and reference characteristics of linear timeinvariant systems by stationary compensation of unstable excitation models.* International Journal of Control, 41(1):259–269, 1985.

[LK05] Lückel, Joachim und Thorsten Koch: *Funktionsorientierter Entwurf mechatronischer Systeme.* Skript zur Vorlesung, Universität Paderborn, 2005.

[Loh94] Lohmann, Boris: *Ordnungsreduktion und Dominanzanalyse nichtlinearer Systeme.* at – Automatisierungstechnik, 42(10):466–474, 1994.

[LS04] Lohmann, Boris und Behnam Salimbahrami: *Ordnungsreduktion mittels Krylov-Unterraummethoden.* at – Automatisierungstechnik, 52(1):30–38, 2004.

[Lun06] Lunze, Jan: *Regelungstechnik 2: Mehrgrößensysteme, Digitale Regelung.* Springer-Verlag, Berlin, 4. Auflage, 2006.

[LZT97] Lawrence, C. T., J. L. Zhou und A. L. Tits: *User's Guide for CFSQP Version 2.5: A C Code for Solving (Large Scale) Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying All Inequality Constraints.* Technical Report TR-94-16r1, Institute for Systems Research, University of Maryland, 1997.

[MAK+08] Münch, Eckehard, Philipp Adelt, Martin Krüger, Bernd Kleinjohann und Ansgar Trächtler: *Hybrid planning and hierarchical optimization of mechatronic systems.* In: *International Conference on Control, Automation and Systems (ICCAS 2008)*, Seiten 1047–1054, Seoul, Südkorea, 2008.

[Mar66] Marshall, S. A.: *An Approximate Method for Reducing the Order of a Linear System.* Control, 10:642–643, 1966.

[Mat07a] The Mathworks: *Genetic Algorithm and Direct Search Toolbox – User's Guide*, 2007.

[Mat07b] The Mathworks: *Optimization Toolbox – User's Guide*, 2007.

[MC03] Masse, John und Thierry Cambois: *Differentiation, Sensitivity Analysis and Identification of Hybrid Models – Symbolic derivative of Simulink models.* In: *Réunion Française des Utilisateurs de Logiciels dédiés à la Modélisation et au Calcul Scientifique*, Rueil-Malmaison, France, 2003.

[MÇT07] Münch, Eckehard, Hüseyin Çinkaya und Ansgar Trächtler: *Realisierung eines Werkzeuges zur algorithmischen Differenzierung von Modellen mechatronischer Systeme.* In: *Paderborner Workshop Entwurf Mechatronischer Systeme (EMS)*, Paderborn, 2007. HNI-Verlagsschriftenreihe.

[MHO04a] Münch, Eckehard, Thorsten Hestermeyer und Oliver Oberschelp: *Sollbahn-Planung für schienengebundene Fahrzeuge.* In: *VDI-Tagung Berechnung und Simulation im Fahrzeugbau*, Würzburg, 2004.

[MHO+04b] Münch, Eckehard, Thorsten Hestermeyer, Oliver Oberschelp, Peter Scheideler und Andreas Schmidt: *Distributed optimization of reference trajectories for active suspension with multi-agent systems.* In: *European Simulation Multiconference 2004 – Networked Simulations and Simulated Networks*, Seiten 343–350, Magdeburg, Juni 2004. SCS.

[Mün01] Münch, Eckehard: *Fortentwicklung und Realisierung eines Verfahrens zur gleichzeitigen Optimierung mehrerer Zielgrößen.* Studienarbeit, Universität Paderborn, 2001.

[Mün03] Münch, Eckehard: *Mehrgrößenoptimierung – Algorithmusentwicklung und Anwendung an der Spurführung der NBP (Neue Bahntechnik Paderborn).* Diplomarbeit, Universität Paderborn, 2003.

[Moo81] Moore, B. C.: *Principal component analysis in linear system: controllability, observability and model reduction.* IEEE Transactions on Automatic Control, AC-26:17–32, 1981.

[Mos04] Mostaghim, Sanaz: *Multi-Objective Evolutionary Algorithms – Data Structures, Convergence, and Diversity.* Dissertation, Universität Paderborn, Universitätsbibliothek Paderborn, 2004.

[MT06] Münch, Eckehard und Ansgar Trächtler: *Observation of Gradients, by Means of Algorithmic Differentiation.* In: *32nd Annual Conference of the IEEE Industrial Electronics Society (IECON'06)*, Paris, France, 2006.

[MVH05] Münch, Eckehard, Henner Vöcking und Thorsten Hestermeyer: *Self-Learning Disturbance Compensation for Active Suspension Systems.* In: *International Conference on Informatics in Control, Automation and Robotics*, Barcelona, Spain, 2005.

[Nag02] The Numerical Algorithms Group: *NAG C Library Manual, Mark 7*, 2002.

[Nau00] Naumann, Rolf: *Modellierung und Verarbeitung vernetzter intelligenter mechatronischer Systeme.* Dissertation, Universität Paderborn, VDI-Fortschritt-Berichte, Reihe 20, Nr. 318, VDI-Verlag, 2000.

[Nau02] Naumann, Tino: *Wissensbasierte Optimierungsstrategien für elektronische Steuergeräte an Common-Rail-Dieselmotoren.* Dissertation, Technische Universität Berlin, Universitätsbibliothek der TU Berlin, 2002.

[NEO10] *NEOS-Wiki.* `http://www.neos-guide.org/NEOS/`, Stand 18. Dez. 2010.

[Obe08] Oberschelp, Oliver: *Strukturierter Entwurf selbstoptimierender mechatronischer Systeme.* Dissertation, Universität Paderborn, Band 241, HNI-Verlagsschriftenreihe, Paderborn, 2008.

[OBR05] Oberthür, Simon, Carsten Böke und Franz Josef Rammig: *Ein selbstoptimierendes Echtzeitbetriebssystem für verteilte selbstoptimierende Systeme.* In: *Echtzeitaspekte bei der Koordinierung Automomer Systeme (PEARL 2005)*, 2005. Boppard am Rhein, Deutschland.

[OEC96] Otter, Martin, Hilding Elmqvist und François E. Cellier: *„Relaxing" – A Symbolic Sparse Matrix Method Exploiting the Model Structure in Generating Efficient Simulation Code.* In: *Symposium on Modelling, Analysis and Simulation, CESA'96, IMACS MultiConference on Computational Engineering in Systems Applications*, Lille, France, 1996.

[OHG04a] Oberschelp, Oliver, Thorsten Hestermeyer und Holger Giese: *Structured Information Processing for Self-Optimizing Mechatronic Systems.* In: *1st International Conference on Informatics in Control, Automation and Robotics (ICINCO 2004)*, Setubal, Portugal, 2004.

[OHG04b] Oberschelp, Oliver, Thorsten Hestermeyer und Holger Giese: *Strukturierte Informationsverarbeitung für selbstoptimierende mechatronische Systeme.* In: *2. Paderborner Workshop „Intelligente mechatronische Systeme"*, HNI, Paderborn, 2004.

[OMH+08] Osmic, Semir, Eckehard Münch, Stefan Henkler, Wilhelm Schäfer, Holger Giese, Martin Hirsch und Ansgar Trächtler: *Safe Online-Reconfiguration of Self-Optimizing Mechatronic Systems.* In: *Proceedings of the 7th international Heinz Nixdorf Symposium*, Band 223 der Reihe *HNI-Verlagschriftenreihe*, Seiten 411–428, Paderborn, 2008.

[OTC03] *Optimization Technology Center.* `http://www.ece.northwestern.edu/OTC/`, Stand 7. Nov. 2003.

[OZL10] Oberthür, Simon, Leszek Zaramba und Hermann-Simon Lichte: *Flexible Resource Management for Self-X Systems: An Evaluation.* In: *Proceedings of First IEEE Workshop on Self-Organizing Real-Time Systems – SORT 2010.* IEEE CS Press, Mai 2010.

[Pap96] Papageorgiou, Markos: *Optimierung – Statische, dynamische, stochastische Verfahren für die Anwendung.* Oldenbourg Verlag, München, 2. Auflage, 1996.

[Pir01] Piram, Udo: *Zur Optimierung elastischer Mehrkörpersysteme.* Dissertation, Universität Stuttgart, VDI-Fortschritt-Berichte, Reihe 11, Nr. 298, VDI-Verlag, 2001.

[Rai94] Raisch, Jörg: *Mehrgrößenregelung im Frequenzbereich.* Oldenbourg Verlag, München, 1994.

[RC10] *RailCab – Neue Bahntechnik Paderborn.* `http://www.railcab.de/`, Stand 18. Dez. 2010.

[RGR07] Rutenbar, Rob A., Georges G. E. Gielen und Jaijeet Roychowdhury: *Hierarchical Modeling, Optimization and Synthesis for System-Level Analog and RF Desing.* In: *Proceedings of the IEEE*, Band 97, Seiten 640–669, 2007.

[Rut98] Rutz, Rüdiger: *Prozeßbasierte Entwurfswerkzeuge für mechatronische Systeme und Anwendungen in der Fahrwerksregelung.* Dissertation, Universität Paderborn, 1998.

[SB89] Sastry, Shankar und Marc Bodson: *Adaptive Control: Stability, Convergence, and Robustness.* Prentice-Hall Advanced Reference Series (Engineering). Prentice-Hall Englewood Cliffs, New Jersey, 1989.

[SB00] Stoer, Josef und Roland Bulisch: *Numerische Mathematik 2.* Springer-Verlag, Berlin, Heidelberg, New York, 3. Auflage, 2000.

[Sch85] Schittkowski, K.: *NLQPL: A FORTRAN-Subroutine Solving Constrained Nonlinear Programming Problems.* Annals of Operations Research, 5(6):485–500, 1985.

[Sch04] Schütze, Oliver: *Set Oriented Methods for Global Optimization.* Dissertation, Universität Paderborn, Universitätsbibliothek Paderborn, 2004.

[Sch06] Schlautmann, Philipp: *Entwicklung eines neuartigen dreidimensionalen aktiven Federungssystems für ein Schienenfahrzeug.* Dissertation, Universität Paderborn, Universitätsbibliothek Paderborn, 2006.

[Sch09] Schäfer, Erika: *Modular-hierarchische modellbasierte Entwicklung und Optimierung einer Regelung für ein aktives Federungssystem.* Dissertation, Universität Paderborn, Universitätsbibliothek Paderborn, 2009.

[SFB01] *Sonderforschungsbereich 1799: Selbstoptimierende Systeme des Maschinenbaus – DFG-Finanzierungsantrag.* Universität Paderborn, 2001.

[SFB04] *Sonderforschungsbereich 614: Selbstoptimierende Systeme des Maschinenbaus – DFG-Fortsetzungsantrag.* Universität Paderborn, 2004.

[SFB08] *Sonderforschungsbereich 614: Selbstoptimierende Systeme des Maschinenbaus – DFG-Fortsetzungsantrag.* Universität Paderborn, 2008.

[SFB10] *Sonderforschungsbereich 614: Selbstoptimierende Systeme des Maschinenbaus.* `http://www.sfb614.de/`, Stand 18. Dez. 2010. Homepage.

[SP96] Skogestad, Sigurd und Ian Postlethwaite: *Multivariable Feedback Control – Analysis and Design Using Frequency-domain Methods.* John Wiley & Sons, 1996.

[Sto99] Stoer, Josef: *Numerische Mathematik 1.* Springer-Verlag, Berlin, Heidelberg, New York, 3. Auflage, 1999.

[Str96] Streiter, Ralph H.: *Entwicklung und Realisierung eines analytischen Regelkonzeptes für eine aktive Federung.* Dissertation, Technische Universität Berlin, 1996.

[TMV06] Trächtler, Ansgar, Eckehard Münch und Henner Vöcking: *Iterative Learning and Self-Optimization Techniques for the Innovative RailCab-System.* In: *32nd Annual Conference of the IEEE Industrial Electronics Society (IECON'06)*, Paris, France, 2006.

[Toe02] Toepper, Stephanie: *Die mechatronische Entwicklung des Parallelroboters TriPlanar.* Dissertation, Universität Paderborn, VDI-Fortschritt-Berichte, Reihe 8, Nr. 966, VDI-Verlag, 2002.

[Trä06] Trächtler, Ansgar: *Railcab – mit innovativer Mechatronik zum Schienenverkehrssystem der Zukunft.* In: *VDE Kongress*, Aachen, Deutschland, 2006.

[Unb00] Unbehauen, Heinz: *Regelungstechnik II – Zustandsregelungen, digitale und nichtlineare Regelsysteme.* Vieweg Verlag, 8. Auflage, 2000.

[VDI02] *VDI-Richtlinie 2057 Blatt 1: Einwirkung mechanischer Schwingungen auf den Menschen – Ganzkörper-Schwingungen*, Düsseldorf, 2002. Verein Deutscher Ingenieure.

[VDI04] *VDI-Richtlinie 2206: Entwicklungsmethodik für mechatronische Systeme*, Düsseldorf, 2004. Verein Deutscher Ingenieure.

[VMM07] Van der Aa, N., H. ter Morsche und R. Mattheij: *Computation of eigenvalue and eigenvector derivatives for a general complex-valued eigensystem.* Electronic Journal Of Linear Algebra, 16(October 2006):300–314, 2007.

[VT08] Vöcking, Henner und Ansgar Trächtler: *Self-optimization of an Active Suspension System Regarding Energy Requirements.* In: *International Conference on Control, Automation and Systems 2008 (ICCAS)*, Seoul, Korea, October 2008.

[Wäc02] Wächter, Andreas: *An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering.* PhD thesis, Carnegie Mellon University Pittsburgh, Pennsylvania, 2002.

[WKG05] Walther, Andrea, Andreas Kowarz und Andreas Griewank: *ADOL-C: A Package for the Automatic Differentiation of Algorithms Written in C/C++, Version 1.10.0.* Dresden, Berlin, 2005.