52

Karlsruher Schriftenreihe Fahrzeugsystemtechnik

Mathias Cabrera Cano

Neuronale Netze mit externen Laguerre-Filtern zur automatischen numerischen Vereinfachung von Getriebemodellen

KIT Scientific Publishing

Mathias Cabrera Cano

Neuronale Netze mit externen Laguerre-Filtern zur automatischen numerischen Vereinfachung von Getriebemodellen

Karlsruher Schriftenreihe Fahrzeugsystemtechnik
Band 52

Herausgeber
FAST Institut für Fahrzeugsystemtechnik
Prof. Dr. rer. nat. Frank Gauterin
Prof. Dr.-Ing. Marcus Geimer
Prof. Dr.-Ing. Peter Gratzfeld
Prof. Dr.-Ing. Frank Henning

Das Institut für Fahrzeugsystemtechnik besteht aus den Teilinstituten Bahnsystemtechnik, Fahrzeugtechnik, Leichtbautechnologie und Mobile Arbeitsmaschinen.

Eine Übersicht aller bisher in dieser Schriftenreihe erschienenen Bände finden Sie am Ende des Buchs.

Neuronale Netze mit externen Laguerre-Filtern zur automatischen numerischen Vereinfachung von Getriebemodellen

von
Mathias Cabrera Cano

KIT Scientific Publishing

Dissertation, Karlsruher Institut für Technologie (KIT)
Fakultät für Maschinenbau, 2016

Impressum

Karlsruher Institut für Technologie (KIT)
KIT Scientific Publishing
Straße am Forum 2
D-76131 Karlsruhe

www.ksp.kit.edu

Print on Demand 2017 – Gedruckt auf FSC-zertifiziertem Papier

ISSN 1869-6058
ISBN 978-3-7315-0621-8
DOI: 10.5445/KSP/1000064462

Vorwort des Herausgebers

Die Simulation ist heute ein etabliertes Werkzeug um Entwicklungszeit und -kosten zu reduzieren. Sie wird auf verschiedenen Ebenen der Fahrzeugentwicklung eingesetzt. Beispiele sind die Auslegung eines Getriebes, die Berechnung des Verhaltens eines Fahrzeugs oder die Absicherung einer Steuerung. Problematisch ist heute jedoch die Verwendung durchgängiger Simulationsmodelle in allen genannten Gebieten. Heute werden für die unterschiedlichen Anforderungen spezifische Modelle entwickelt.

Die Karlsruher Schriftenreihe Fahrzeugsystemtechnik geht der Fragestellung einer optimierten Fahrzeugentwicklung nach. Für die Fahrzeuggattungen Pkw, Nfz, Mobile Arbeitsmaschinen und Bahnfahrzeuge werden in der Schriftenreihe Forschungsarbeiten vorgestellt, die Fahrzeugtechnik auf vier Ebenen beleuchten: das Fahrzeug als komplexes mechatronisches System, die Fahrer-Fahrzeug-Interaktion, das Fahrzeug im Verkehr und Infrastruktur sowie das Fahrzeug in Gesellschaft und Umwelt.

Am Beispiel eines Stufenautomaten entwickelt Herr Cabrera Cano im vorliegenden Band 52 eine Methode zur automatischen numerischen Vereinfachung von Getriebemodellen. Er greift damit eine sehr aktuelle Entwicklungsproblematik auf. Konkret zeigt er, wie statische, neuronale Netze in Kombination mit Laguerre-Filtern für die dynamische Getriebesimulation eingesetzt werden können.

Die neuronalen Netze können mit Hilfe von Trainingsdaten aus dem detaillierten Simulationsmodell trainiert werden und erhalten durch die Kombination mit den Laguerre Filtern eine Dynamik. So ist es möglich, ein vereinfachtes Simulationsmodell automatisiert abzuleiten, das eine hohe Modellgüte, im untersuchten Fall von $\mathrm{RMSE}_{\mathsf{TS}} = 0,19$, besitzt.

Karlsruhe, im November 2016 *Prof. Dr.-Ing. Marcus Geimer*

Neuronale Netze mit externen Laguerre-Filtern zur automatischen numerischen Vereinfachung von Getriebemodellen

Zur Erlangung des akademischen Grades eines
Doktors der Ingenieurwissenschaften
(Dr.-Ing.)

bei der Fakultät für Maschinenbau
des Karlsruher Instituts für Technologie (KIT)

genehmigte
DISSERTATION

von

Dipl.-Phys. Mathias Cabrera Cano

Datum der mündlichen Prüfung:	15.12.2016
Referent:	Prof. Dr. Marcus Geimer
Korreferent:	Prof. Dr. Oliver Nelles

Abstract

Today's passenger cars are equipped with many different functions, so that the fuel consumption is decreased or the driving safety and comfort are increased. These functions are realized using different mechatronic systems. In order to support the development of mechatronic systems and functions, simulation has become an important tool. The use of simulations allows an evaluation of new concepts to realize new functions at early stages of the development process by modeling the interaction between control and the physical environment of the mechatronic systems. Furthermore, it is possible to ensure that the developed functions work as specified. A special type of simulation is the usage of a real controler with a model of the physical environment. This type of simulation is called hardware-in-the-loop-simulation and the used model has to be real-time capable.

In many of the performed simulations during the development of these functions and mechatronic systems, models of automatic transmissions with different levels of datail are needed. For the development of the control of an automatic transmission, detailed models of the electrohydraulic actuators are need, so that a thorough study of the resulting shift dynamics is possible. However these detailed models of an automatic transmission are not used for other simulations taks where models with a lower level of detail are needed due to the numerical complexity of the detailed models. The numerical complexity of detailed models of automatic transmission is governed by coupling the fast dynamics of the hydraulic model and the slow dynamics of the mechanic model leading to a stiff system. In addition the clutches which enable the change from one gear

to another are modeled using a state machine. The result of the two properites is that during the simulation of the shift dynamics, very small step sizes of the used numerical solver are needed to resolve the shift dynamic correctly. The result is that often new models of an automatic transmission are created. These models have a lower level of detail and numerical complexity and are used for example in a real-time simulation or for the validation of different functions. Furthermore, the level of detail is also low in that sense that also a simplified model of the transmission has to be used because the electrohydraulic model is neglected. Thus the simplified model describes only the static ratios of the automatic transmission and dynamic effects are replaced by a very simple model.

Hence using a scheme for an automatic reduction of the numerical complexity of detailed transmission models can help to reduce the modeling costs. One way to do so is using modeling experience and figure out which physical effects can be neglected. This leads in many cases to a model with reduced numerical complexity. The disadvantage of this method is that it is not easy to automate.
Another possiblity allowing an automatic reduction of the numerical complexity is the usage of mathematical model order reduction. These methods are applied to the model equations with the aim to reduce the number of states of these equations. In the case of the models describing the shift dynamics of automatic transmission it is important to select a method that is suitable for equations describing nonlinear dynamics. This is a crucial constraint because the development of methods for model order reduction of nonlinear dynamic system is part of current research. Furthermore, as described above, the numerical complexity is not generated by too many states. Thus applying mathematical model order reduction does not reduce significantly the numerical complexity of detailed models of an automatic transmission.
Therefore in this work the aim is to create a black box model which describes the shift dynamics with lower numerical complexity. To create this black box

model data-based modeling is used. The challenge here is to find a suitable model structure which can capture the nonlinear shift dynamics of an automatic transmission.
In order to reduce modeling costs, a scheme for an automatic creation of the black box model is needed.

In this work it is shown that neural networks with external Laguerre-filters are suitable for modeling the shift dynamics of an automatic transmission. This type of neural network also reduces the numerical complexity. Further, it is figured out how these type of neural networks can be created automatically.
Therefore the parameters of the neural networks are separated into model parameters and hyperparameters. Model parameters are determined using linear or nonlinear optimization methods by minimizing the square sum of modeling errors. The hyperparameters describe the structure of the used type of neural network such as number of neurons and the properties of the external Laguerre-filters. Finding optimal values for the hyperparameter is equivalent to finding the neural net with the best generalization properties. Therefore different created neural networks with different hyperparameters have to be compared to each other. Finally, the set of hyperparameter is selected which led to the best generalization properties.

The method for automatic numerical simplification presented in this work allows an automatic determination of the model parameters and hyperparameters. In this work, the derived method is applied for numerical simplifcation of a detailed transmission model with nine gears. The result is that the needed time for simulation can be reduced by 10 to 13 times. In addition, a fixed step solver with $0,01[\mathrm{s}]$ can be used. In this example this allows to use the created neural networks in a real-time simulation. Furthermore, the use of the created neural networks allows to use the same model of the transmission control as it is possible using a detailed model of an automatic transmission.

Zusammenfassung

Heutige PKW verfügen über zahlreiche Funktionen, die den Fahrkomfort und die Fahrsicherheit erhöhen und den Kraftstoffverbrauch senken. Zur Realisierung der verschiedenen Funktionen werden zahlreiche mechatronische Systeme benötigt, die sich durch ein Wechselspiel aus Software bzw. Steuergerät und physikalischer Umgebung auszeichnen. Zur Unterstützung der Entwicklung neuer Funktionen und der dazu verwendeten mechatronischen Systeme, in denen dieses Wechselspiel untersucht wird, ist die Simulation ein wichtiges Werkzeug geworden. Dies ermöglicht die frühzeitige Bewertung von Konzepten zur Systemauslegung und die Absicherung der Funktionen. Zudem ermöglicht die Verwendung einer Simulation Kosten bei der Entwicklung von Prototypen einzusparen. Eine Besonderheit stellt dabei die Simulation eines realen Steuergeräts mit einer virtuellen physikalischen Umgebung dar, die sogenannte HiL-Simulation. Dabei muss das verwendete Modell der physikalischen Umgebung echtzeitfähig sein.

Während der Entwicklung der Funktionen von mechatronischen Systemen kommen sehr unterschiedliche Simulationsfragestellungen auf, in denen unter anderem Modelle von Stufenautomaten in unterschiedlichen Detaillierungsgraden benötigt werden. So wird beispielsweise für die Entwicklung der Funktionssoftware des Getriebesteuergeräts ein detailliertes Modell der elektrohydraulischen Ansteuerung benötigt, bei dem die Ansteuerströme des realen Steuergeräts oder eines Modells des Steuergeräts berücksichtigt werden können. Die detaillierten Modelle werden in der Fachabteilung für die Getriebeentwicklung überwiegend durch theoretische Modellbildung erstellt und beschreiben deshalb sehr valide

die Schaltdynamik. Allerdings haben sie den Nachteil, dass die Modellgleichungen durch gemeinsame Modellierung von hydraulischen und mechanischen Anteilen steif sind und Unstetigkeiten durch die Modellierung der Reibung in den Schaltelementen beinhalten, so dass der numerische Aufwand in einer Simulation stark erhöht ist.

Zur Absicherung des Getriebesteuergeräts in einer Echtzeitsimulation oder zur Absicherung weiterer Funktionen der PKW werden daher in den meisten Fällen numerisch vereinfachte Modelle benötigt. Diese werden häufig neu erstellt, wodurch zusätzlicher Modellierungsaufwand während der Entwicklung entsteht. Ein weiteres Problem ist dabei, dass häufig nicht klar ist, wie valide die vereinfachten Getriebemodelle sind, da diese Modelle oftmals nicht vom Fachbereich der Getriebeentwicklung erstellt werden. Zudem ermöglichen die vereinfachten Getriebemodelle keine Berücksichtigung der Ansteuerströme der elektrohydraulischen Ansteuerung. Deshalb sollen bereits validierte Getriebemodelle für andere Simulationsfragestellungen während der Entwicklung so angepasst werden, dass diese dort schließlich verwendet werden können.

Für die dazu notwendige numerische Vereinfachung detaillierter Getriebemodelle gibt es verschiedene Möglichkeiten, zum Beispiel durch Anwendung von Expertenwissen das Modell so anzupassen, dass ein numerisch weniger aufwändiges Modell entsteht. Allerdings lässt sich dies nur schwer automatisieren, so dass der Modellierungsaufwand auch hier erhöht ist.
Eine weitere Möglichkeit ist die Anwendung mathematischer Verfahren, wodurch eine automatische numerische Vereinfachung erreicht werden kann. Dabei werden die vorliegenden Modellgleichungen so angepasst, dass ein Modell mit weniger Zustandsgleichungen entsteht. Da jedoch der Hauptgrund für die numerischen Schwierigkeiten detaillierter Getriebemodelle nicht eine zu hohe Anzahl von Zuständen ist, ist durch Anwendung dieser Verfahren keine signifikante numerische Vereinfachung möglich.

Deshalb wird als weitere Möglichkeit die automatische Ableitung eines numerisch vereinfachten Ersatzmodells verwendet.
Die Erstellung dieses Ersatzmodells erfolgt durch Methoden der experimentellen Modellbildung. Als Ersatzmodell wird eine spezielle Klasse von dynamischen neuronalen Netzen verwendet, bei der externe Laguerre-Filter mit statischen neuronalen Netzen kombiniert werden. Als statisches neuronales Netz kann dabei zwischen MLP-Netzen und LM-Netzen ausgewählt werden. Neben der Optimierung der Modellparameter der neuronalen Netze müssen auch optimale Werte für die Hyperparameter bestimmt werden. Die Hyperparameter sind dabei Parameter, die die Struktur des neuronalen Netzes bestimmen, wodurch dessen Generalisierungsfähigkeit beeinflusst wird. Um eine automatische numerische Vereinfachung zu ermöglichen, wird daher in dieser Arbeit eine Methode erarbeitet, die eine automatische Struktur- und Parameteroptimierung dieser Klasse von neuronalen Netzen ermöglicht, wodurch der Modellierungsaufwand zur Erstellung von numerisch vereinfachten Getriebemodellen gesenkt werden kann. Bei der Strukturoptimierung werden die Hyperparameter so bestimmt, dass das neuronale Netz die beste Generalisierungsfähigkeit besitzt.

Die erarbeitete Methode zur numerischen Vereinfachung wird schließlich an einem detaillierten Modell eines Stufenautomaten mit neun Gängen angewandt. Dieses Modell zeichnet sich durch eine relativ große Anzahl von Schaltelementen aus, so dass es sich als Benchmark für andere Getriebemodelle eignet. Dabei wird gezeigt, wie die für die Simulation benötigten neuronalen Netze automatisch erstellt werden und dass das Schaltverhalten des detaillierten Modells valide approximiert wird. Zudem wird gezeigt, dass die automatisch erstellten neuronalen Netze in einer Simulation zur Untersuchung der Schaltdynamik genutzt werden können, wobei weiter die Ansteuerströme der elektrohydraulischen Ansteuerung verwendet werden können. Das Ergebnis ist, dass die Simulationszeit um den Faktor 10 bis 13 verkürzt werden kann. Zudem ermöglicht die Verwendung der erstellten neuronalen Netze die Auswahl eines

Simulationsverfahrens mit deutlich größerer, fester Simulationsschrittweite von $0,01$[s], ohne dass es zu instabilem Verhalten kommt. So ist in diesem Beispiel eine Verwendung der neuronalen Netze in einer Echtzeitsimulation möglich.

Danksagung

Diese Doktorarbeit entstand in Zusammenarbeit mit der Forschung der Daimler AG am Standort Böblingen. Daher möchte ich mich bei der Daimler AG für die Rahmenbedingungen bedanken, die es mir möglich gemacht haben, diese Arbeit zu verfassen. Besonderer Dank gilt dabei meinem Betreuer Dr. Dietmar Neumerkel, der mir immer fachlich mit gutem Rat zur Seite stand. Bei den weiteren Kollegen der Daimler AG möchte ich mich für zahlreiche Diskussionen bedanken, die mich thematisch weitergebracht haben.

Weiter möchte ich meinem Doktorvater Prof. Dr. Marcus Geimer meinen Dank für die Betreuung und die konstruktiven Diskussionen aussprechen. Durch die thematische Offenheit für das in dieser Arbeit behandelte Thema der Modellierung mit neuronalen Netzen ist diese Arbeit überhaupt erst möglich geworden.

Mit der Übernahme des Korreferats durch Prof. Dr. Oliver Nelles sind weitere Anregungen hinzugekommen, besonders durch seine Erfahrung mit der Modellierung mit lokalen Modellnetzen. Dabei waren besonders die Diskussionen über die Modellierung mit lokalen Modellnetzen und deren Anwendung hilfreich, wofür ich mich bedanken möchte.

Zu guter Letzt möchte ich mich bei meiner Familie und meinen Freunden für die Unterstützung bedanken, vor allem in der Phase, nachdem die Anstellung als Doktorand zu Ende ging und das Weiterführen der Schreibarbeit nebenberuflich erfolgte. Dabei möchte ich besonderen Dank Kathrin Fritsch aussprechen, die mich in dieser anstrengenden Phase immer unterstützt und motiviert hat, diese Arbeit zu Ende zu führen.

Wichtige verwendete Symbole, Einheiten und Abkürzungen

Symbole

Lateinische Buchstaben

BP_j	Bester ermittelter Wert für den j-ten Hyperparameter eines neuronalen Netzes durch die Liniensuche
$\mathbf{c}_j$	Koordinate des Zentrums der j-ten Aktivierungsfunktion von LM-Netzen
f_D	Schwingungsfrequenz der Sprungantwort eines linearen dynamischen Systems zweiter Ordnung
$\mathbf{i}_{\mathrm{G,A}}$	Ansteuerströme zwischen Stufenautomat und Steuergerät des Stufenautomaten
$i_{\mathrm{G,A,j}}$	j-ter Ansteuerstrom zwischen Stufenautomat und Steuergerät des Stufenautomaten
$\mathbf{i}_{\mathrm{G,S}}$	Sensorströme zwischen Stufenautomat und Steuergerät des Stufenautomaten
$\mathbf{i}_{\mathrm{F,S}}$	Ansteuerströme zwischen Fahrwerk und den verschiedenen Steuergeräten des Fahrwerks
$\mathbf{i}_{\mathrm{F,A}}$	Sensorströme zwischen Fahrwerk und den verschiedenen Steuergeräten des Fahrwerks
$\mathbf{i}_{\mathrm{M,A}}$	Ansteuerströme zwischen Verbrennungsmotor und Steuergerät des Motors

$\mathbf{i}_{\mathsf{M,S}}$	Sensorströme zwischen Verbrennungsmotor und Steuergerät des Motors
$\mathsf{itr}_{\max}$	Maximale Anzahl der Iterationen der Liniensuche
k	diskretisierte Zeit
M	Anzahl der lokalen Modelle bzw. Anzahl der Neuronen pro verdeckter Schicht
$\mathsf{m_s}$	Steigung der Straße
N	Anzahl der Abtastzeitpunkte von aufgezeichneten Daten
n	Anzahl der verdeckten Schichten von MLP-Netzen
$\mathsf{N_O}$	Anzahl der Suchdurchläufe der Liniensuche
NP_j	Anzahl der Werte des j-ten Suchraums
n_{w}	Anzahl der Modellparameter eines neuronalen Netzes
P_j	j-ter Hyperparameter eines neuronalen Netzes
p	Anzahl der Eingangssignale von statischen bzw. dynamischen Systemen
p_{ad}	Arbeitsdruck der Getriebehydraulik
$\mathsf{p_s}$	Druck in der hydraulischen Zuleitung eines Schaltelements
$\mathsf{p_w}$	Druck in der hydraulischen Zuleitung der WÜK
px	Anzahl der Signale im **x**-Regressor bzw. Dimension des **x**-Regressors
pz	Anzahl der Signale im **z**-Regressor bzw. Dimension des **z**-Regressors
$\mathbf{Q}$	Gewichtungsmatrix für die Least-Squares-Schätzung von LM-Netzen
q^{-1}	Zeitverzögerungsoperator
R	Übersetzung zwischen der Winkelgeschwindigkeit des Motors und der Antriebswelle
r_{min}	kleinster möglicher Abstand zwischen zwei Zentren der Aktivierungsfunktion von LM-Netzen, die beim inkrementellen Clustering-Verfahren ermittelt werden

SP_j	Suchraum für den j-ten Hyperparameter
t	kontinuierliche Zeit
T_{at}	Drehmoment an der Antriebswelle des Fahrzeugs
T_{brk}	Drehmoment der Bremse
T_{K}	Zeitkonstante eines dynamischen Systems
T_{m}	Drehmoment des Motors
T_{S}	Abtastzeit der Simulationsdaten
T_{r}	Drehmoment der Reifen
T_{w}	Drehmoment des Wandlers
$\mathbf{u}$	Eingangssignale eines statischen bzw. dynamischen Systems
$\mathbf{u}_s$	Eingangssignal mit statischem Einfluss
$\mathbf{u}_d$	Eingangssignal mit dynamischem Einfluss
$\mathbf{u}_{f,j}$	Vektor der gefilterten Signale des j-ten Eingangssignals durch einen OBF-Filter
$\dot{V}_{\mathsf{s}}$	Volumenstrom in der hydraulischen Zuleitung eines Schaltelements
$\dot{V}_{\mathsf{w}}$	Volumenstrom in der hydraulischen Zuleitung der WÜK
$\mathbf{w}$	Vektor der Modellparameter eines neuronalen Netzes
$\mathbf{X}$	Matrix der ausgewählten Simulationsdaten für den $\mathbf{x}$-Regressor der lokalen Modelle
$\mathbf{x}_{\mathsf{d}}$	Signal mit dynamischem Einfluss des $\mathbf{x}$-Regressors
$\mathbf{x}_{\mathsf{s}}$	Signal mit statischem Einfluss des $\mathbf{x}$-Regressors
$\mathbf{y}$	Vektor des Ausgangssignals zu den gegebenen Abtastzeiten
$\hat{\mathbf{y}}$	Vektor des approximierten Ausgangssignals zu den gegebenen Abtastzeiten
$\mathbf{Z}$	Matrix der ausgewählten Simulationsdaten für den $\mathbf{z}$-Regressor der lokalen Modelle
$\mathbf{z}_{\mathsf{d}}$	Signal mit dynamischem Einfluss des $\mathbf{z}$-Regressors
$\mathbf{z}_{\mathsf{s}}$	Signal mit statischem Einfluss des $\mathbf{z}$-Regressors

Griechische Buchstaben

α	Pol von Laguerre-Filtern
ΔP	Rauschamplitude, die bei der Liniensuche mit stochastischen Elementen verwendet wird
η	Ordnung von Laguerre-Filtern
η_z	Maximal zulässige Ordnung von durch Laguerre-Filter gefilterten Eingangssignalen im $\mathbf{z}$-Regressor von LM-Netzen
Σ	Kovarianzmatrix von Gaußfunktionen
σ	Einheitlicher Skalierungsfaktor für die Aktivierungsfunktionen von LM-Netzen
τ	Abklingzeit der Sprungantwort eines linearen dynamischen Systems zweiter Ordnung
μ_j	j-te Aktivierungsfunktion von RBF-Netzen bzw. nicht normierte j-te Aktivierungsfunktion von LM-Netzen
ω_{at}	Winkelgeschwindigkeit der Antriebswelle des Fahrzeugs
ω_{m}	Winkelgeschwindigkeit des Motors
ω_{r}	Winkelgeschwindigkeit der Reifen
ω_{w}	Winkelgeschwindigkeit des Wandlers

Einheiten

[Hz]	Frequenzangabe in Hertz
[h]	Zeitangabe in Stunden
[min]	Zeitangabe in Minuten
[Nm]	Drehmoment in Newton-Meter
[s]	Zeitangabe in Sekunden
$\left[\frac{\text{rad}}{\text{s}}\right]$	Winkelgeschwindigkeit in Rad pro Sekunde

Abkürzungen

AT	(engl. Automatic Transmission) Stufenautomat
FIR	Finite-Impulse-Response
HiL	Hardware-in-the-Loop
HILOMOT	Hierarchical-Local-Model-Tree
LM-Netz	Lokales Modellnetz
LOLIMOT	Local-Linear-Model-Tree
IIR	Infinite-Impulse-Response
MiL	Model-in-the-Loop
MLP-Netz	Multi-Layer-Perceptron-Netz
MIMO	Multiple-Input-Multiple-Output
MISO	Multiple-Input-Single-Output
SG	Steuergerät eines mechatronischen Systems
SiL	Software-in-the-Loop
SISO	Single-Input-Single-Output
OBF	Orthonormal-Basis-Function (Funktionen, die eine vollständige, normierte Basis bilden)
PKW	Personenkraftwagen
PT_2	Lineares dynamisches System zweiter Ordnung
RMSE	Root-Mean-Square-Error (Wurzel des mittleren quadratischen Fehlers)
$RMSE_T$	RMSE der Trainingsdaten
$RMSE_{TS}$	RMSE der Testdaten
$RMSE_V$	RMSE der Validierungsdaten
V-Modell	Modell einer Vorgehensweise zur Entwicklung mechatronischer Systemen
WÜK	Wandlerüberbrückungskupplung

Mathematische Symbole

$f_i(q)$	i-ter OBF-Filter
$L(q, \alpha, \eta)$	Laguerre-Filter der Ordnung η mit Pol α
$\mathrm{linspace}(\mathrm{lb}, \mathrm{ub}, \mathsf{NP})$	Gleichmäßige, lineare Aufteilung des Intervalls $[\mathrm{lb}, \mathrm{ub}]$ in NP Teilintervalle
$\max(\mathbf{r})$	Maximaler Wert eines beliebigen Vektors $\mathbf{r}$
$\min(\mathbf{r})$	Minimaler Wert eines beliebigen Vektors $\mathbf{r}$
$\mathsf{NN}(\cdot)$	Neuronales Netz
$\Delta\mathsf{P} \cdot \mathsf{randn}$	Gaußverteilte Zufallsvariable mit Standardabweichung $\Delta\mathsf{P}$
$\mathsf{Train}(\cdot)$	Bestimmung der Modellparameter eines neuronalen Netzes

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Heutige PKW[1] bestehen aus einer Vielzahl mechatronischer Systeme, durch die zahlreiche neuartige Funktionen realisiert werden. Diese Funktionen ermöglichen es beispielsweise, Fahrkomfort und Fahrsicherheit zu erhöhen bzw. den Kraftstoffverbrauch zu senken. Beispiele für mechatronische Systeme sind das Motorsteuergerät zur Regelung des Drehmoments oder das Getriebesteuergerät zur automatischen Durchführung von Schaltvorgängen.

In zunehmendem Maße werden neue Funktionen nicht nur durch zusätzliche mechatronische Systeme realisiert, sondern auch dadurch, dass bereits vorhandene mechatronische Systeme besser untereinander vernetzt werden, wodurch diese miteinander interagieren können. Ein Beispiel dafür ist der Abstandsregelautomat, der auf mechatronische Systeme des Antriebsstrangs zugreifen kann, so dass Schaltvorgänge durchgeführt werden können oder das Drehmoment des Motors angepasst wird. Entsprechend kann auf die mechatronische Ansteuerung der Bremsen zugegriffen werden, um notfalls auch Bremsungen durchzuführen.

Die Funktionsweise der im Fahrzeug verbauten mechatronischen Systeme wird durch ein interdisziplinäres Zusammenspiel von Elektrotechnik, Maschinenbau und Informatik bzw. Informationstechnik realisiert (siehe Abbildung 1.1). Dabei zeichnen sich mechatronische Systeme durch eine Wechselwirkung zwischen

[1] PKW ... Personenkraftwagen

einem oder mehreren Steuergerät(en) und der physikalischen Umgebung der bzw. des Steuergräte(s) aus.

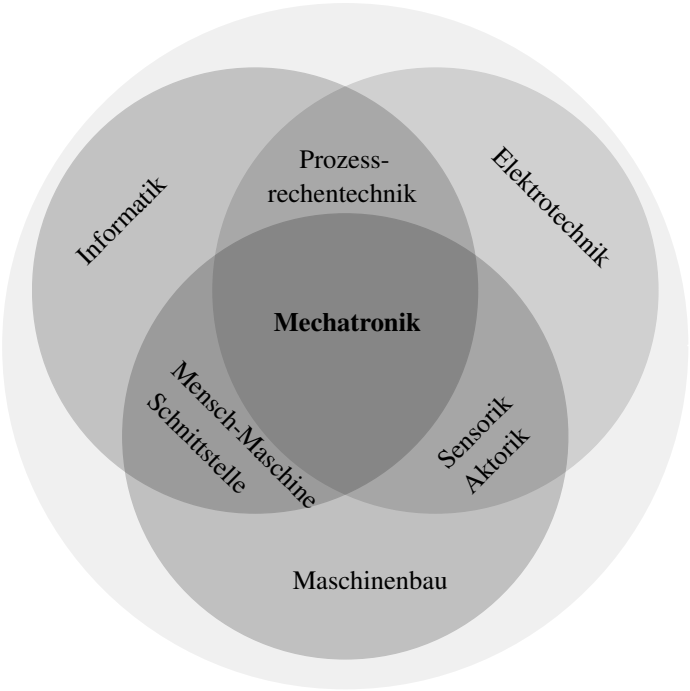


Abbildung 1.1: Interdisziplinäres Zusammenspiel verschiedener Fachbereiche bei mechatronischen Systemen

Hier ist zu beobachten, dass durch die Steigerung des Anteils der Informationstechnik der Aufwand der Softwareentwicklung weiter zunimmt. So ist zum Beispiel die Realisierung eines Abstandsregelautomaten mit einem großen Anteil von Softwareentwicklung verbunden.
Um diese Entwicklung zu unterstützen, werden vermehrt Simulationen eingesetzt, wodurch es möglich ist, in frühen Phasen der Entwicklung Fehler in der

Umsetzung zu erkennen und zu vermeiden oder mit der Softwareentwicklung zu beginnen, wenn noch keine Steuergeräte verfügbar sind.
Durch das interdisziplinäre Zusammenspiel mechatronischer Systeme reicht es jedoch bei der Simulation nicht aus, nur die Abläufe innerhalb der Software bzw. des Steuergeräts zu betrachten, sondern es muss auch die physikalische Umgebung betrachtet werden, in der die Software bzw. das Steuergerät eingebettet ist. Dies trifft vor allem auf die Entwicklung von Funktionen zu, die durch eine stärkere Vernetzung verschiedener mechatronischer Systeme realisiert werden. Durch die Verwendung von Simulation können daher bereits in einer frühen Entwicklungsphase Konzeptuntersuchungen durchgeführt werden.

Für diese unterschiedlichen Simulationsfragestellungen werden unterschiedliche Modelle verwendet. Die Software, die in frühen Phasen vorliegt und in den Simulationen zum Einsatz kommt, ist noch unvollständig und besteht nur aus einem Teil der späteren Software. Daher können die in den Simulationen verwendeten Modelle unterteilt werden in Modelle von Software bzw. Steuergeräten und in Modelle der physikalischen Umgebung. Je nach Simulationsfragestellung werden auch unterschiedlich detaillierte Modelle der physikalischen Umgebung verwendet.
So kommen am Anfang der Entwicklung weniger detaillierte Modelle zum Einsatz, da zunächst prinzipielle Fragestellungen beantwortet werden. Erst mit Fortschreiten der Entwicklung, wenn der Ansatz zur Funktionsrealisierung klarer wird, werden nach und nach detailliertere Modelle verwendet.
Einen besonderen Typ von Simulationsfragestellungen stellt die Absicherung eines realen Steuergeräts dar. Für diese Simulationen wird ein echtzeitfähiges Modell der physikalischen Umgebung benötigt. Um die Echtzeitfähigkeit zu erreichen, müssen bestimmte numerische Kriterien des Modells erfüllt sein, so dass eine Berechnung innerhalb eines vorgegeben Zeitintervalls möglich wird. Um die Anforderungen an die Modelle in den verschiedenen Simulationsfragestellungen gezielt unterstützen zu können, kommen Simulationswerkzeuge zum

Einsatz, in denen aus Modellbibliotheken unterschiedlich detaillierte Modelle der einzelnen mechatronischen Systeme des PKWs erstellt bzw. bereits vorhandene ausgewählt werden können.
Um den Entwicklungsprozess mechatronischer Systeme zu unterstützen, ist es daher von Bedeutung, Methoden zu entwickeln, die die Modellierung und Simulation unterstützen. Durch diese Methoden ist es zum Beispiel möglich, detaillierte Modelle numerisch zu vereinfachen, bestimmte Simulationswerkzeuge, die jeweils auf eine bestimmte Domäne spezialisiert sind, miteinander zu koppeln oder Modelle aus verschiedenen Simulationswerkzeugen zusammenzuführen.

1.2 Problemstellung

In dieser Arbeit liegt der Fokus auf der Modellierung der physikalischen Umgebung von Stufenautomaten und auf der Entwicklung einer Methode zur numerischen Vereinfachung detaillierter Getriebemodelle[2]. So werden in der Fachabteilung, in der die Getriebeentwicklung durchgeführt wird, detaillierte Modelle der Schaltdynamik benötigt, wobei hier der Fokus auf der Simulation des Schaltverhaltens liegt. Eine komfortable Erstellung dieser Modelle wird durch entsprechende Simulationswerkzeuge ermöglicht. Da detaillierte Modelle der Schaltdynamik numerisch aufwändig sind, ist der Aufwand für eine numerische Simulation erhöht. Dies hat folgende Ursachen:
Es wird ein detailliertes Modell der elektrohydraulischen Ansteuerung sowie der Reibung in den Schaltelementen erstellt. Dabei entsteht ein steifes Differentialgleichungssystem. Zudem wird die Reibung zustandsbasiert modelliert. Dies hat zur Folge, dass während der Simulation eines Schaltvorgangs sehr kleine Simulationszeitschrittweiten verwendet werden müssen, um die Schaltdynamik

[2] In dieser Arbeit bezeichnet Getriebemodell ein Modell eines Stufenautomaten.

berechnen zu können. Daher werden detaillierte Getriebemodelle meist in anderen Simulationsfragestellungen nicht verwendet.
In den Fachabteilungen, die Fahrerassistenzsysteme wie beispielsweise einen Abstandsregelautomaten entwickeln, steht nämlich eine Untersuchung des Verhaltens des Gesamtfahrzeugs im Vordergrund. In solch einer Simulationsfragestellung spielt die Schaltdynamik eine weniger wichtige Rolle und es werden numerisch einfachere Getriebemodelle verwendet, bei denen zum Beispiel nur die statischen Übersetzungen berücksichtigt und die Schaltübergänge vereinfacht modelliert werden.
Durch die Vernetzung mechatronischer Systeme steigen jedoch die Anforderungen zur Absicherung. Damit steigen auch die Anforderungen an die Modelle, die in den Simulationen zur Absicherung verwendet werden, so dass auch einer detaillierten Modellierung der Schaltdynamik auf Gesamtfahrzeugebene eine größere Bedeutung zukommen wird. So stellen beispielsweise die Entwicklung hybrider Antriebsstränge und die Auslegung des Zusammenspiels von elektrischem Antrieb, Verbrennungsmotor und Getriebe eine Herausforderung dar.
In dieser Arbeit wird deshalb speziell untersucht, wie detaillierte Getriebemodelle numerisch vereinfacht werden können, um diese in weiteren Simulationsfragestellungen verwenden zu können. Um den Modellierungsaufwand zur Entwicklung mechatronischer Systeme im PKW zu senken, wird weiter untersucht, wie sich die numerische Vereinfachung automatisieren lässt.
Eine Möglichkeit zur numerischen Vereinfachung ist die Verwendung von Wissen, um bestimmte physikalische Effekte zu nähern oder zu vernachlässigen. Allerdings ist es schwer möglich, die dazu notwendigen Schritte zu automatisieren.

Um eine automatische nummerische Vereinfachung erreichen zu können, gibt es Folgende zwei Vorgehensweisen:

1. Anwendung eines mathematischen Verfahrens zur Vereinfachung der Modellgleichungen
2. Erstellung eines numerisch einfacheren Ersatzmodells mit Hilfe experimenteller Modellbildung

Für beide Vorgehensweisen sind keine Standards in der Literatur bekannt, da es sich bei der Schaltdynamik um ein nichtlineares dynamisches System handelt. Zudem ist die Entwicklung von Methoden zur numerischen Vereinfachung nichtlinearer dynamischer Systeme Teil aktueller Forschung.
Bei Anwendung eines mathematischen Verfahrens zur numerischen Vereinfachung ist die Idee, die gegebenen Modellgleichungen so zu vereinfachen, dass der numerische Aufwand sinkt. Dies wird erreicht, indem die Anzahl der Zustände der Modellgleichungen reduziert wird. Dieses Vorgehen ist für lineare dynamische Systeme sehr gut erforscht und findet dort eine breite Verwendung. Für nichtlineare dynamische Systeme werden die Ansätze, die zur numerischen Vereinfachung linearer dynamischer Systeme verwendet werden, entsprechend erweitert, um diese verwenden zu können. Dabei ist vom jeweiligen Fall abhängig, ob eine Anwendung möglich ist. Prinzipiell besteht die Möglichkeit, solch ein Verfahren zur automatischen numerischen Vereinfachung von Getriebemodellen zu verwenden. Da jedoch der numerische Aufwand nicht von einer zu hohen Anzahl von Zuständen herrührt, kann hier kaum eine signifikante numerische Vereinfachung erzielt werden. In manchen Fällen ist zudem eine Anwendung dieser Verfahren nicht möglich, da die Modellgleichungen nur in Form von Software vorliegen.
Daher wird in dieser Arbeit gezeigt, wie durch Erstellung eines Ersatzmodells eine automatische numerische Vereinfachung detaillierter Getriebemodelle er-

möglicht werden kann. Um die nichtlineare Schaltdynamik zu erfassen, ist die Verwendung neuronaler Netze ein geeigneter Ansatz als Ersatzmodell.

1.3 Zielsetzung und Aufbau der Arbeit

Ziel der Arbeit ist daher die Entwicklung einer Methode, um detaillierte Getriebemodelle automatisiert numerisch zu vereinfachen, wozu neuronale Netze verwendet werden. Durch Verwendung eines neuronalen Netzes in der Simulation sind die numerisch aufwändigen Teilmodelle des ursprünglichen Getriebemodells nicht mehr enthalten, so dass auf diese Weise eine numerische Vereinfachung geschieht.
Um das dynamische Verhalten von Getriebemodellen zu beschreiben, werden entsprechende Eingangssignale und Ausgangssignale vorgegeben und in der Simulation aufgezeichnet. Das neuronale Netz wird dann dazu verwendet, diese zu approximieren. Dazu werden die Parameter des neuronalen Netzes so bestimmt, dass die gegebenen Eingangs- und Ausgangsdaten mit dem kleinstmöglichen quadratischen Fehler beschrieben werden. Damit die numerische Vereinfachung automatisiert geschehen kann, muss die Erstellung des neuronalen Netzes automatisch erfolgen. Eine wichtige Aufgabe ist dabei, dass die Optimierung der Parameter des neuronalen Netzes automatisch erfolgt.
Eine weitere wichtige Aufgabe bei der automatischen Erstellung neuronaler Netze besteht darin, dass auch die Struktur des neuronalen Netzes optimiert werden muss. Das bedeutet, dass unterschiedliche Strukturen bei der Modellierung des nichtlinearen Verhaltens zu unterschiedlichen Modellgüten und einer unterschiedlichen Generalisierungsfähigkeit führen. Ziel ist es, ein neuronales Netz mit einer möglichst hohen Güte und Generalisierungsfähigkeit und so automatisiert wie möglich zu erstellen.
Um die Schaltdynamik beschreiben zu können, muss eine dynamische Modellbildung auf Basis eines neuronalen Netzes geschehen, wodurch weitere

Freiheitsgrade der Struktur des neuronalen Netzes entstehen. All diese zusätzlichen Struktureigenschaften eines neuronalen Netzes werden durch zusätzliche Parameter vorgegeben, die in dieser Arbeit als Hyperparameter bezeichnet werden. Die zur Modellerstellung zusätzlich notwendige Strukturoptimierung erfolgt dabei durch eine Optimierung der Hyperparameter.
Zusammenfassend ist eine automatische Erstellung von neuronalen Netzen möglich, wenn folgende Fragestellungen geklärt werden:

- Auswahl eines geeigneten Ansatzes zur dynamischen Modellierung der Schaltdynamik mit neuronalen Netzen
- Verfahren zur automatischen Strukturoptimierung
- Auswahl eines geeigneten Verfahrens zur automatischen Optimierung der Modellparameter
- Verwendung von speziellem Wissen der Schaltdynamik von Getrieben zur Reduktion der Freiheitsgrade der Struktur des neuronalen Netzes

Um diese Fragestellungen zu beantworten, ist die Arbeit ist wie folgt aufgebaut:

In **Kapitel 2** wird auf Grundlagen der Entwicklung mechatronischer Systeme eingegangen. Dazu gehören der Aufbau von mechatronischen Systemen und verschiedene Aspekte der Funktions- und Softwareentwicklung und wie diese durch Simulationen unterstützt wird. Dabei werden die Grundlagen der Modellbildung und Simulation zusammengefasst, die für die simulationsgestützte Entwicklung relevant sind. Insbesondere wird beschrieben, wie diese zur Fahrzeugmodellierung und in Kombination mit Getriebemodellen verwendet werden.

In **Kapitel 3** wird auf den Aufbau und die Entwicklung von Stufenautomaten eingegangen. Dabei steht insbesondere die Dynamik von Schaltvorgängen im Fokus, die als Lastschaltvorgänge durchgeführt werden. Anschließend wird ein

Überblick über verschiedene Ansätze zur Modellbildung von Stufenautomaten gegeben, mit denen die Dynamik von Schaltvorgängen in unterschiedlichen Detaillierungsgraden beschrieben werden kann. Außerdem wird beschrieben, welche Modellierungswerkzeuge zum Einsatz kommen und welche Simulationsfragestellungen jeweils betrachtet werden. Es werden dabei die jeweiligen numerischen Eigenschaften der Modelle und Ansätze beschrieben, mit denen eine numerische Vereinfachung ermöglicht wird. Dabei wird gezeigt, dass ein Bedarf zur Entwicklung einer Methode zur automatischen numerischen Vereinfachung besteht, und welche Anforderungen diese erfüllen muss.

In **Kapitel 4** werden die zuvor abgeleiteten Anforderungen, die die Methode erfüllen muss, aufgegriffen und dazu verwendet, die Ziele der automatischen numerischen Vereinfachung näher zu definieren. Dabei wird gezeigt, dass sich dazu neuronale Netze eigenen. Zudem wird ein Überblick über die Erstellung und den Aufbau neuronaler Netze gegeben. Es wird auf Verfahren zur Parameteroptimierung und Ansätze zur Strukturoptimierung eingegangen und diskutiert, wie sich eine automatische Erstellung von neuronalen Netzen realisieren lässt. Dazu wird vorgestellt, welche Hyperparameter bei der Strukturoptimierung neuronaler Netze berücksichtigt werden müssen. Außerdem wird gezeigt, dass durch die Verwendung neuronaler Netze mit externen Laguerre-Filtern die Ziele der Methode zur numerischen Vereinfachung erreicht werden können.

In **Kapitel 5** wird die Erstellung neuronaler Netze mit externen Laguerre-Filtern vertieft. Dabei wird zunächst beleuchtet, wie Laguerre-Filter zur Modellierung linearer dynamischer Systeme verwendet werden können. Die Verwendung in Kombination mit einem statischen neuronalen Netz erlaubt schließlich die Modellierung nichtlinearer dynamischer Systeme. Dabei wird vorgestellt, welche Parameter und Hyperparameter bei der Modellerstellung automatisch bestimmt werden müssen. Es wird zudem diskutiert, welche Ansätze dies ermöglichen. Als Ergebnis wird dazu eine Liniensuche vorgestellt.

In **Kapitel 6** werden die einzelnen Schritte zur Erstellung numerisch vereinfachter Modelle von Stufenautomaten beschrieben. Besonders wichtig ist dabei, dass generische Modellstrukturen bei der Modellerstellung verwendet werden können. Durch Verwendung von Wissen über die Schaltdynamik von Stufenautomaten, das in Kapitel 3 beschrieben ist, kann die Anzahl der zu bestimmenden Hyperparameter reduziert werden, so dass der Aufwand der Strukturoptimierung gesenkt werden kann.

In **Kapitel 7** wird die vorgestellte Methodik verwendet, um ein Modell eines Stufenautomaten mit neun Gängen numerisch zu vereinfachen. Die Präsentation der Ergebnisse gliedert sich dabei in zwei Teile. Im ersten Teil wird auf den Verlauf der automatischen Modellerstellung und die Modellgüte der erstellten neuronalen Netze eingegangen. Anschließend werden im zweiten Teil die numerischen Eigenschaften und die Genauigkeit der vereinfachten neuronalen Netze in einer Simulation diskutiert. Dabei wird gezeigt, dass in diesem Beispiel durch die Verwendung der erstellten neuronalen Netze eine Echtzeitsimulation möglich ist.

In **Kapitel 8** werden die Ergebnisse dieser Arbeit zusammengefasst und diskutiert, für welche Simulationsfragestellungen die durch die vorgestellte Methode erstellten numerisch vereinfachten Modelle verwendet werden können.
Weiter wird diskutiert, wie die Methode angepasst werden kann, um diese in anderen Anwendungen zur numerischen Vereinfachung einzusetzen.

2 Grundlagen der Entwicklung mechatronischer Systeme

In diesem Kapitel wird ein Überblick über verschiedene Aspekte der simulationsgestützten Entwicklung mechatronischer Systeme gegeben, wobei der Fokus auf der Softwareentwicklung liegt. Der Überblick besteht dabei aus folgenden zwei Teilen: Im ersten Teil wird zunächst auf den Aufbau mechatronischer Systeme eingegangen, und es werden im PKW vorkommende Beispiele genannt. Ein wichtiger Aspekt dabei ist das Vorgehen zur Entwicklung mechatronischer Systeme, und wie dies durch Simulationen unterstützt werden kann.
Im zweiten Teil wird auf die Grundlagen der Modellbildung und Simulation eingegangen. Dabei werden verschiedene Ansätze der Modellbildung thematisiert, die bei der Entwicklung mechatronischer Systeme zum Einsatz kommen. Ein wichtiger Teil sind numerische Verfahren, die zur Simulation verwendet werden, und Ansätze, durch die die numerischen Eigenschaften von Modellen vereinfacht werden können. Zum Abschluss wird ein Überblick über Modellierungs- und Simulationswerkzeuge gegeben, die in der simulationsgestützten Entwicklung zum Einsatz kommen. Dabei wird beleuchtet, wie Modelle auch von anderen Fachabteilungen verwendet werden können.

2.1 Simulationsgestützte Entwicklung mechatronischer Systeme

Mechatronische Systeme zeichnen sich durch das interdisziplinäre Zusammenspiel unterschiedlicher Fachbereiche aus (siehe Abbildung 1.1). Die verschie-

denen Fachbereiche sind dabei Informatik, Elektrotechnik und Maschinenbau [Ise08b]. Ein mechatronisches System besteht aus einem oder mehreren Steuergerät(en), einer Aktuatorik, einer Strecke und einer Sensorik. Dabei ist das Steuergerät der Informatik zugeordnet, während die Schnittstellen zu Aktuatorik und Sensorik der Elektrotechnik zugeordnet sind. Die Strecke kann dabei ein mechanisches System oder ein weiteres mechatronisches System sein. Letztere Möglichkeit entspricht der Vernetzung mechatronischer Systeme, wodurch sich weitere Funktionen durch den Fachbereich der Informatik realisieren lassen.
Der Ablauf in einem mechatronischen System ist dabei wie folgt: Das Steuergerät bekommt über den Sensor Informationen über die Strecke. Diese Informationen werden im Steuergerät verarbeitet, um über die Aktuatorik gezielt das Verhalten der Strecke zu beeinflussen. Eine weitere Eigenschaft eines mechatronischen Systems ist eine Schnittstelle, die eine Bedienung durch Vorgabe von Stellwerten ermöglicht (siehe Abbildung 2.1).

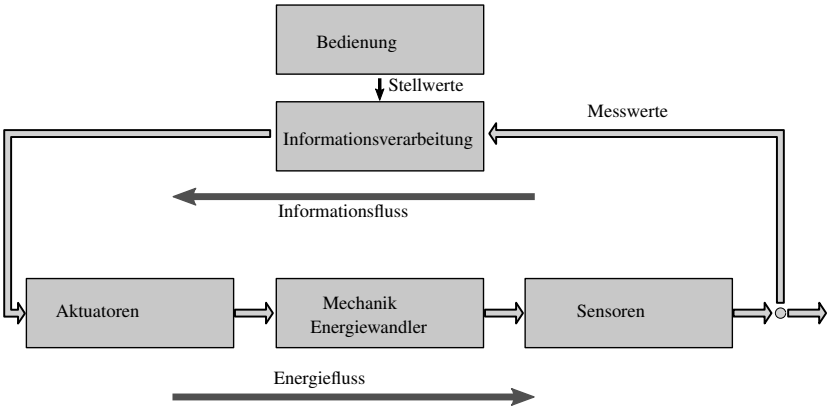


Abbildung 2.1: Aufbau mechatronischer Systeme

Um das dynamische Verhalten eines mechanischen Systems bzw. der Strecke gezielt beeinflussen zu können, werden physikalische Effekte als Aktuatoren

verwendet, die eine Stellkraft erzeugen. Ein Beispiel für einen Aktuator ist die Umwandlung elektrischer Ströme unter Verwendung zusätzlicher Energie in eine mechanische Stellkraft, wobei das elektromechanische Prinzip verwendet wird [Ise08b; Jan09]. Durch die Stellkräfte wird ein Eingreifen des Steuergeräts in den dynamischen Prozess ermöglicht.
Die Sensorik verwendet ebenfalls physikalische Effekte zur Messung von Größen, jedoch in umgekehrter Weise. So wird zum Beispiel der piezoelektrische Effekt zur Messung von Kräften verwendet [Jan09]. Dabei entstehen elektrische Ströme, die im Steuergerät entsprechend aufbereitet und ausgewertet werden. Ein Überblick darüber, welche physikalischen Effekte in der Aktuatorik und Sensorik verwendet werden, ist in [Ise08b; Jan09] gegeben. In heutigen PKW findet sich eine große Anzahl mechatronischer Systeme. Dazu werden an dieser Stelle ein paar Beispiele genannt, die in [Ise08c] zu finden sind:

- Einzelne mechatronische Systeme im PKW:
  - Automatische Schaltgetriebe
  - Mechatronische Ansteuerung der Drosselklappe des Verbrennungsmotors
  - Mechatronische Bremsen zur Regelung des Bremsmoments
- Vernetzung mechatronischer Systeme im PKW:
  - Spurhalteassistent, der Lenkung, Bremsmoment und Motormoment beeinflussen kann
  - Betriebsstrategie hybrider Antriebsstränge, die das Zusammenspiel von Verbrennungsmotor, Elektromotor und Batterie bestimmt

Um mechatronische Systeme besser beschreiben und das Zusammenspiel verstehen zu können, können diese hierarchisch angeordnet werden. Abbildung 2.2 zeigt dazu ein Beispiel für eine Zerlegung eines mechatronischen Systems in drei Hierarchieebenen. Dazu ist das mechatronische System in Komponenten und Subkomponenten aufgeteilt worden [Jan09].

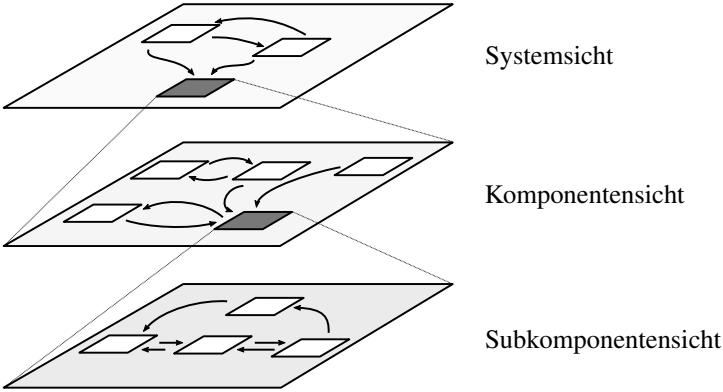


Abbildung 2.2: Hierarchischer Aufbau mechatronischer Systeme

Dies lässt sich auf einen PKW übertragen, wodurch sich zum Beispiel folgende Aufteilung ergibt:
Auf oberster Ebene steht der Fahrer, der auf das Fahrzeug einwirkt. Zudem stehen Fahrzeug und Fahrer mit der Umwelt in Wechselwirkung. So findet zum Beispiel die Kraftübertragung auf die Straße statt, und der Fahrer muss auf Verkehr und Straßenverlauf reagieren. Auf zweiter Ebene stehen die verschiedenen mechatronischen Systeme des Fahrzeugs, wie Motor, Getriebe oder Bremse (siehe Abbildung 2.3). Hier findet durch die Reifen die Kraftübertragung auf die Straße statt, die ein Teil der Umwelt auf der höheren Hierarchieebene ist. Diese Komponenten können weiter in Subkomponenten zerlegt werden. In Kapitel 3 wird dazu auf die Subkomponenten von Stufenautomaten eingegangen und gezeigt, wie Schaltvorgänge realisiert werden. Durch das domänenübergreifende Zusammenwirken ist die Entwicklung mechatronischer Systeme eine besondere Herausforderung. Dabei muss die Komplexität beherrscht werden, die einerseits durch die Heterogenität und andererseits durch die Vernetzung entsteht.

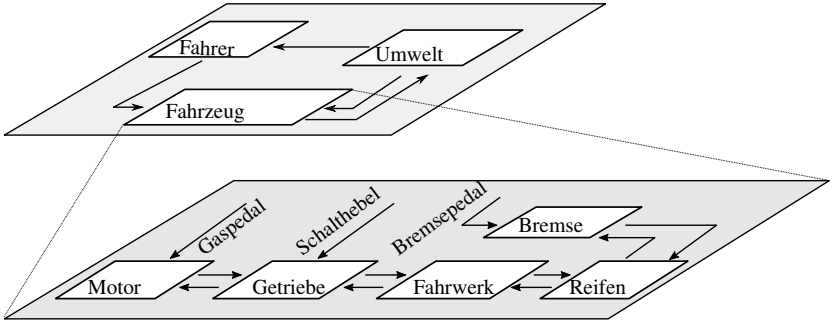


Abbildung 2.3: Beispielhafte Darstellung des hierarchischen Aufbaus mechatronischer Systeme eines Fahrzeugs

Im nächsten Abschnitt wird deshalb ein einheitliches Vorgehen bei der Entwicklung mechatronischer Systeme vorgestellt.

2.1.1 Vorgehen zur Entwicklung mechatronischer Systeme

Zur Entwicklung mechatronischer Systeme hat sich das V-Modell etabliert [SZ13; Ise08b; Jan09]. Das V-Modell ist eine Beschreibung eines möglichen einheitlichen Vorgehens, um mechatronische Systeme zu entwickeln und ist in Abbildung 2.4 dargestellt. Es soll dabei helfen, die Komplexität mechatronischer Systeme zu beherrschen und frühzeitig Fehler in der Entwicklung zu erkennen. Bei der Vorgehensweise wird von der hierarchischen Anordnung mechatronischer Systeme Gebrauch gemacht, wie im Folgenden beschrieben wird. Der Vorgang nach dem V-Modell besteht nach [Ise08b; Jan09] aus folgenden Schritten:
Zunächst werden Anforderungen von Kunden gesammelt, die dazu verwendet werden, weitere Anforderungen abzuleiten, die das mechatronische System erfüllen muss, um die Kundenanforderungen zu erfüllen.
Anschließend findet der Systementwurf statt, bei dem verschiedene Ansätze

zur Realisierung untersucht werden, wie die abgeleiteten Anforderungen erfüllt werden können. Dabei wird das System wie oben beschrieben in verschiedene Hierarchieebenen zerlegt. Der Systementwurf endet, wenn alle benötigten Komponenten, Softwareanteile und Steuergeräte vorliegen.
Danach erfolgt die Systemintegration, bei der die Komponenten, Softwareanteile und Steuergeräte nacheinander zusammengeführt werden und überprüft wird, ob die jeweils abgeleiteten Anforderungen erfüllt werden. Die Systemintegration ist beendet, wenn alle genannten Teile des Systems zusammengeführt und abgesichert sind. Zum Abschluss wird beim fertigen Produkt überprüft, ob die Kundenanforderungen erfüllt sind.

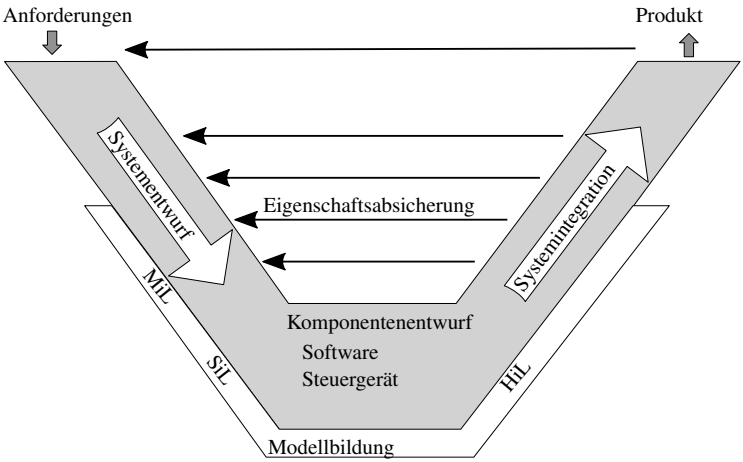


Abbildung 2.4: V-Modell zur Entwicklung mechatronischer Systeme (angelehnt an [Ise08b])

Die Schritte des V-Modells zeigen, dass die Softwareentwicklung nur ein Teil des gesamten Entwicklungsprozesses ist. Daneben müssen auch Aktuatoren, Sensoren und die mechanischen Anteile entwickelt werden. Bei heutigen mechatronischen Systemen des PKW ist jedoch zu beobachten, dass ein immer

größer werdender Anteil der Funktionsrealisierung durch Software stattfindet und diese deshalb eine immer wichtigere Rolle einnimmt. Deshalb liegt der Fokus dieser Arbeit auf der simulationsgestützten Entwicklung von Software. Allerdings muss bei der Entwicklung auch das Zusammenspiel mit dem Steuergerät betrachtet werden. Dabei werden im Steuergerät zum Beispiel folgende Funktionen durch Software realisiert:

- Informationsaustausch mit anderen Steuergeräten, um zum Beispiel vernetzte Funktionen zu realisieren.
- Diagnose und Fehlererkennung, ob es zu einem Ausfall bestimmter Funktionen gekommen ist.
- Signalaufbereitung der von der Sensorik eingespeisten elektrischen Ströme und Ansteuerung der Aktuatoren über elektrische Ströme.
- Umsetzung von Algorithmen zur Regelung, Überwachung oder Steuerung. Dieser Anteil wird auch als Funktionssoftware bezeichnet [SZ13].

Für die Entwicklung von Software wird folgende Sichtweise verwendet [SZ13; Ise08b]:
Das mechatronische System kann in einen Teil unterteilt werden, der durch Informationstechnik geprägt ist, und in einen weiteren, physikalischen Teil, auf den der informationstechnische Anteil einwirkt. Die physikalischen Anteile sind dabei durch das Zusammenspiel verschiedener physikalischer Domänen geprägt, während die informationstechnischen Anteile durch Software und Steuergeräte geprägt sind. Dieses Zusammenspiel ist in Abbildung 2.1 durch den Informationsfluss gekennzeichnet, der für die informationstechnischen Anteile steht, und den Energiefluss, der für die physikalischen Anteile steht. Das Zusammenspiel dieser beiden Flussgrößen bildet die Grundlage für die simulationsgestützte Softwareentwicklung, auf die im nächsten Abschnitt eingegangen wird.

2.1.2 Verfahren zur simulationsgestützten Softwareentwicklung

In diesem Abschnitt wird auf prinzipielle Unterschiede zwischen verschiedenen Simulationsverfahren eingegangen, die zur Unterstützung der Entwicklung der Software nach dem V-Modell verwendet werden. Das Ziel aller Verfahren ist, dass in einer Simulation die Auswirkung der durch Software realisierten Funktion auf eine virtuelle physikalische Umgebung betrachtet werden kann. Die dazu verwendeten Simulationsverfahren werden dabei danach unterschieden, in welchem Reifegrad die Funktionssoftware bzw. das Steuergerät vorliegen [Güh02; Güh03; Jan09]. Dabei wird zwischen folgenden drei Verfahren unterschieden, die bei der Entwicklung nach dem V-Modell zum Einsatz kommen (siehe Abbildung 2.4):

- Model-in-the-Loop (MiL) Simulation:
  Bei dieser Simulation werden einfache Funktionsmodelle mit einer virtuellen physikalischen Umgebung verwendet. Dieses Verfahren wird in frühen Entwicklungsphasen verwendet, während verschiedene Konzepte zum Systementwurf untersucht werden. Es kann beispielsweise bei der Erstellung der funktionalen Logik und der Untersuchung eines neuartigen Regelalgorithmus helfen. Dabei kommen auch häufig vereinfachte Modelle der physikalischen Umgebung zum Einsatz, da noch unbekannt ist, wie diese genau aufgebaut ist.
- Software-in-the-Loop (SiL) Simulation:
  Bei dieser Simulation wird die fertige Funktionssoftware mit einer virtuellen physikalischen Umgebung verwendet. Dann können die Funktionen optimiert und abgesichert werden. Diese Art der Simulation wird hauptsächlich zur Entwicklung der Software einer Komponente verwendet. Bei der Simulation werden meist schon detaillierte Modelle der physikalischen Umgebung verwendet.

- Hardware-in-the-Loop (HiL) Simulation:
  Bei dieser Simulation wird ein reales Steuergerät mit einer virtuellen physikalischen Umgebung verwendet. Dazu sind allerdings echtzeitfähige Modelle der physikalischen Umgebung nötig[1]. Diese Simulationen dienen dazu, weitere Softwareanteile und die Hardwareanteile des Steuergeräts abzusichern. Diese Art der Simulationen wird hauptsächlich zur Unterstützung der Systemintegration verwendet.

In Abbildung 2.5 sind die Unterschiede zwischen den hier vorgestellten Simulationsverfahren graphisch dargestellt.

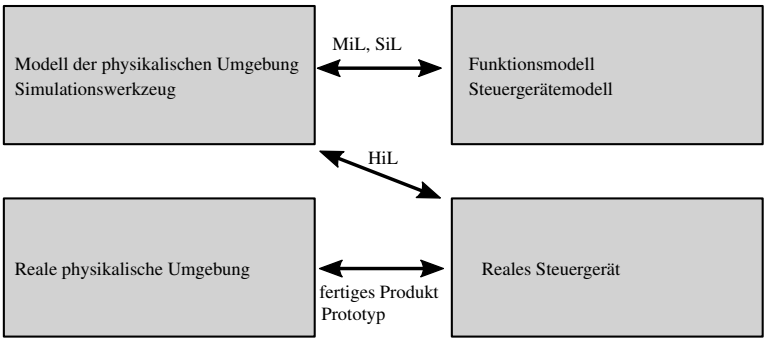


Abbildung 2.5: Verfahren der simulationsgestützten Entwicklung

2.2 Modellbildung und Simulation

In diesem Abschnitt werden Ansätze zur Modellbildung der physikalischen Umgebung von Steuergeräten bzw. Modellen von Steuergeräten vorgestellt. Bei der physikalischen Umgebung handelt es sich im Allgemeinen um ein

[1] Im nächsten Abschnitt wird darauf eingegangen, wann ein Modell aus numerischer Sicht echtzeitfähig ist

dynamisches System. Ein dynamisches System besitzt dabei im Allgemeinen p Eingangssignale und q Ausgangssignale und wird auch als MIMO-System bezeichnet[2]. Sonderfälle von dynamischen Systemen sind das MISO-System[3] und das SISO-System[4]. Ersteres steht für ein dynamisches System mit p Eingangssignalen und einem Ausgangssignal, letzteres für ein dynamisches System mit einem Eingangssignal und einem Ausgangssignal.
Um das zeitliche Verhalten eines dynamischen Systems in Abhängigkeit von Eingangssignalen in einer Simulation zu untersuchen, werden überwiegend Modellgleichungen mit konzentrierten Parametern verwendet. Solche Parameter vernachlässigen die Raumverteilung einer bestimmten physikalischen Größe und konzentrieren diese auf einen Punkt. So wird zum Beispiel statt einer Temperaturverteilung eine einheitliche Gehäusetemperatur verwendet.
Im Gegensatz dazu stehen Modellgleichungen mit verteilten Parametern, die zusätzlich räumliche Abhängigkeiten von physikalischen Größen berücksichtigen, wie zum Beispiel die zuvor erwähnte Temperaturverteilung [Ise08a]. Für die simulationsgestützte Funktionsentwicklung werden diese meist nicht berücksichtigt, da zur Aufstellung der Modellgleichungen geometrische Eigenschaften benötigt werden, die am Anfang der Entwicklung noch unbekannt sind. Zudem ist der numerische Aufwand zur Simulation von Modellgleichungen mit verteilten Parametern erheblich höher [Güh02; Güh03; Dro03].
Bei der Erstellung von Modellen eines dynamischen Systems wird zwischen der theoretischen und der experimentellen Modellbildung unterschieden. In der Praxis werden beide Ansätze je nach Anwendungsfall miteinander kombiniert. Im Folgenden werden diese beiden Ansätze beschrieben.

[2] MIMO ... Multiple-Input-Multiple-Output
[3] MISO ... Multiple-Input-Single-Output
[4] SISO ... Single-Input-Single-Output

2.2.1 Theoretische Modellbildung

Im Folgenden werden Eigenschaften der theoretischen Modellbildung basierend auf [Ise08a; Dro03] vorgestellt. Bei der theoretischen Modellbildung werden physikalische Erhaltungssätze verwendet, um die Modellgleichungen abzuleiten. Für die Simulation dynamischer Systeme wird eine sogenannte Netzwerkmodellierung durchgeführt, wie es von elektrischen Schaltkreisen bekannt ist. Dies hat den Vorteil, dass Analogien zwischen den verschiedenen physikalischen Domänen verwendet werden können, wodurch die Modellbildung erleichtert wird. Auf diesen Aspekt wird weiter unten bei der Vorstellung verschiedener Modellierungswerkzeuge noch einmal eingegangen.
Die dabei zu Grunde liegenden physikalischen Erhaltungssätze sind zum Beispiel die Kirchhoffschen Regeln, durch die unter anderem die Erhaltung der Energie in einem System berücksichtigt wird. Durch die Verwendung physikalischer Erhaltungssätze ist eine sehr genaue Beschreibung der Wirkzusammenhänge der physikalischen Umgebung möglich, was zu einem erhöhten Verständnis derselben führen kann. Deshalb werden diese Modelle auch White-Box-Modelle genannt. Des Weiteren müssen häufig auch Annahmen darüber getroffen werden, welche Effekte vernachlässigt werden können, um ein akzeptables Modell zu erhalten. Das Ergebnis ist eine detaillierte Beschreibung des dynamischen Verhaltens der Aktuatoren und des dynamischen Prozesses. Zudem besitzen die Modelle einen großen Gültigkeitsbereich und können daher auch für Vorhersagen verwendet werden, um zum Beispiel eine Regelung zu optimieren oder die Auswirkung einer Regelung besser zu verstehen.
In der Praxis erfordert es jedoch ein gewisses Maß an Erfahrung, um geeignete Modelle durch theoretische Modellbildung abzuleiten. So muss zum Beispiel bekannt sein, durch welche physikalischen Effekte ein Verhalten beeinflusst wird, damit das dynamische System in der gewünschten Genauigkeit modelliert werden kann. Dabei ist auch wichtig zu wissen, welche physikalischen Effekte

relevant sind und welche vernachlässigt werden können, da sonst schnell unbrauchbare Modelle entstehen.
Die Modellgleichungen, die bei eindimensionalen Modellen mit konzentrierten Parametern entstehen, haben im Allgemeinen die Struktur eines nichtlinearen dynamischen Systems mit den Eingangssignalen **u**, den Ausgangssignalen **y** und den Zustandsgrößen **x** und sind gegeben durch [Ise08a]

$$\begin{aligned} \dot{\mathbf{x}}(t) = &\quad f(\mathbf{x}(t), \mathbf{u}(t), t) \\ \mathbf{y}(t) = &\quad g(\mathbf{x}(t), \mathbf{u}(t), t). \end{aligned} \tag{2.1}$$

Die dabei auftretenden Funktionen f und g sind durch verschiedene Naturgesetze gegeben, wie zum Beispiel die Druckaufbaugleichung in der Hydraulik oder die Newtonschen Bewegungsgleichungen in der Mechanik. Dieses System von Differentialgleichungen besitzt im Allgemeinen keine analytische Lösung. Später wird darauf eingegangen, wie durch numerische Verfahren diese Gleichungen in der Simulation verwendet werden können, um näherungsweise eine Lösung zu finden, damit das Systemverhalten untersucht werden kann.

2.2.2 Experimentelle Modellbildung

Im Folgenden wird das Vorgehen bei der experimentellen Modellbildung vorgestellt. Dabei dienen die Arbeiten von [Lju98; Nel01] als Grundlage. Bei der experimentellen Modellbildung werden zu bestimmten Zeitpunkten aufgezeichnete Daten und eine parametrierbare Modellstruktur verwendet. Die aufgezeichneten Daten beschreiben das statische und das dynamische Verhalten der physikalischen Umgebung bzw. des dynamischen Systems, die bzw. das modelliert werden soll. Mit Hilfe dieser Daten werden die Parameter der Modellstruktur so bestimmt, dass die vorliegenden Daten mit dem kleinstmöglichen quadratischen Fehler beschrieben werden. Dazu muss je nach Modellstruktur ein geeignetes Optimierungsverfahren verwendet werden. Bei der Erstellung

des Modells muss beachtet werden, dass das Modell nur das beschreiben kann, was auch in den Daten vorhanden ist. Deshalb muss bei der Erstellung der Daten darauf geachtet werden, dass das zu modellierende Verhalten auch angeregt wurde. Dabei macht es einen Unterschied, ob das statische oder zusätzlich auch das dynamische Verhalten berücksichtigt werden soll.
Für die experimentelle Modellbildung stehen verschiedene Modellstrukturen zur Auswahl, die nach folgenden Kriterien ausgewählt werden:

- Nichtlineares oder lineares Verhalten
- Dynamisches oder statisches Verhalten.

Zur Modellierung nichtlinearer Zusammenhänge stehen unterschiedliche Ansätze wie Polynome oder neuronale Netze zur Auswahl. Sollen auch dynamische Zusammenhänge modelliert werden, werden statische Modellstrukturen mit zusätzlichen Zeitverzögerungsoperatoren erweitert, wodurch zeitdiskrete Modellgleichungen entstehen. Dadurch bekommen die Modellstrukturen ein Gedächtnis, wodurch eine dynamische Modellierung ermöglicht wird.
Im Gegensatz zur theoretischen Modellbildung haben Parameter der zur Auswahl stehenden Modellstruktur im Allgemeinen keine physikalische Bedeutung und lassen deshalb auch keine physikalische Interpretation zu. Durch die experimentelle Modellierung wird ein Modell erstellt, das nicht die Komplexität des zu modellierenden dynamischen Systems berücksichtigt. Dadurch wird auch wenig dazu beigetragen, das Verständnis für die Wirkzusammenhänge des dynamischen Systems zu erhöhen. Zudem ist der Gültigkeitsbereich auf den Bereich eingeschränkt, der durch die verwendeten Daten gegeben ist. Deshalb werden diese Modelle auch Black-Box-Modelle genannt.
Die so erstellten Modelle eignen sich für Anwendungen, in denen nur eine Nachbildung des Verhaltens der physikalischen Umgebung benötigt wird. Ein Beispiel dafür ist die Absicherung eines Steuergeräts, nachdem der Entwurf der Regelung abgeschlossen ist. Der Vorteil der experimentellen Modellbildung ist

jedoch, dass schon mit geringem Wissen über den dynamischen Prozess ein Modell erstellt werden kann. Die Modellgleichungen, die bei der experimentellen Modellbildung entstehen, sind im Allgemeinen zeitdiskret und gegeben durch

$$\begin{aligned} \mathbf{x}(k+1) = &\quad f(\mathbf{x}(k),\mathbf{u}(k),k) \\ \mathbf{y}(k) = &\quad g(\mathbf{x}(k),\mathbf{u}(k),k). \end{aligned} \tag{2.2}$$

2.2.3 Numerische Simulation

Im Folgenden werden die Grundlagen der numerischen Simulation basierend auf [SWP12] vorgestellt. Das Ziel der numerischen Simulation ist das Lösen von zeitkontinuierlichen Modellgleichungen, um das statische und dynamische Verhalten eines Systems in einer Simulation untersuchen zu können. Dazu wird das in Gleichung (2.1) beschriebene System aus gewöhnlichen Differentialgleichungen verwendet. Die zu lösenden Differentialgleichungen sind gegeben durch

$$\dot{\mathbf{x}} = f(\mathbf{x},\mathbf{u},t) \text{ mit } \mathbf{x}(t_0) = \mathbf{x}_0, \tag{2.3}$$

wobei $\mathbf{x}(t_0)$ die Anfangsbedingung ist.
Die numerischen Verfahren zur Lösung unterscheiden sich in der Art des Integrationsverfahrens und in der Anzahl der verwendeten Stützstellen zur Integration. Bei Verwendung einer Stützstelle liegt ein Ein-Schritt-Verfahren vor, ansonsten von einem Mehr-Schritt-Verfahren. Außerdem können die Integrationsverfahren in explizite und implizite Verfahren unterteilt werden.
Um die numerischen Eigenschaften zu diskutieren, reicht es an dieser Stelle aus, auf das explizite und das implizite Ein-Schritt-Verfahren einzugehen. Um die obige Differentialgleichung zu lösen, wird zunächst die Zeit t folgendermaßen diskretisiert

$$t(k) = t_0 + k \cdot T_S. \tag{2.4}$$

Dabei ist die Abtastzeit T_S das Zeitintervall eines Simulationszeitschritts.
Der Wert $\mathbf{x}(k+1)$ kann durch folgende Integrationsvorschrift berechnet werden

$$\mathbf{x}(k+1) = \mathbf{x}(k) + \int_{t(k)}^{t(k+1)} f(\mathbf{x}, \mathbf{u}, t)dt. \tag{2.5}$$

Ziel ist es nun, durch ein explizites bzw. implizites Ein-Schritt-Integrationsverfahren das Integral zu approximieren. So kann das Integral beispielsweise als Rechteck approximiert werden. Das bedeutet, dass die Funktion im Intervall T_S als konstant angesehen wird. Beim expliziten Verfahren wird dabei der Funktionswert zum Zeitpunkt $t(k)$ und beim impliziten Verfahren zum Zeitpunkt $t(k+1)$ verwendet. Die Lösungen von Gleichung (2.5) sind somit gegeben durch

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{x}(k) + T_S \cdot f(\mathbf{x}(k), \mathbf{u}(k), k) \\ \mathbf{x}(k+1) &= \mathbf{x}(k) + T_S \cdot f(\mathbf{x}(k+1), \mathbf{u}(k+1), k+1). \end{aligned} \tag{2.6}$$

Diese Lösungsvorschriften heißen explizites bzw. implizites Euler-Verfahren. Beim impliziten Verfahren steht der Funktionswert $\mathbf{x}(k+1)$. Somit muss zusätzlich ein Gleichungssystem gelöst werden, um den Funktionswert zu ermitteln. Welches der beiden Verfahren für einen Anwendungsfall besser geeignet ist, hängt von der Funktion f ab, die die Dynamik der Differentialgleichung bestimmt. Damit das explizite Verfahren konvergiert und eine konsistente Lösung angibt, muss die Funktion während des Zeitintervalls näherungsweise konstant sein, da sonst zu große Abweichungen von der tatsächlichen Lösung zustande kommen.
Besitzt ein System sehr hohe Eigenfrequenzen, muss eine sehr kleine Schrittweite T_S gewählt werden, um eine stabile Berechnung der Lösung zu gewährleisten. Der Nachteil ist jedoch, dass die Berechnung dadurch sehr lange dauern kann. Das implizite Verfahren erlaubt jedoch die Verwendung von größeren Schrittweiten, obwohl die Eigenfrequenzen sehr hoch sind. Als Nachteil steht dem

gegenüber, dass ein Gleichungssystem bei jedem Integrationsschritt gelöst werden muss (siehe Gleichung 2.6).
Zur Vollständigkeit wird noch auf die Eigenschaften von Mehr-Schritt-Verfahren eingegangen. Diese verwenden zur Approximation des Integrals aus Gleichung 2.5 mehrere Stützstellen, so dass größere Schrittweiten gewählt werden können. Zudem gibt es Verfahren mit Schrittweiten-Steuerung, so dass die Integrationsschrittweite immer wieder angepasst werden kann [SWP12]. Weitere numerische Schwierigkeiten entstehen durch Unstetigkeiten in den Modellgleichungen. Um diese zu berücksichtigen, werden spezielle numerische Verfahren benötigt [Dro03]. Diese Form der numerischen Schwierigkeit tritt auch bei der Modellierung von Stufenautomaten auf und wird im nächsten Kapitel näher beleuchtet.

Zum Abschluss dieses Abschnitts wird auf die Definition von echtzeitfähigen Modellen eingegangen. Nach [Dro03] ist ein Modell echtzeitfähig, wenn die berechneten Simulationsergebnisse innerhalb eines fest vorgegeben Zeitintervalls vorliegen. Dabei ist zu beachten, dass dieses vorgegebene Zeitintervall kleiner als die echte Zeit ist, damit die Echtzeitfähigkeit erreicht werden kann. Um dies zu gewährleisten, muss ein Simulationsverfahren mit fester Schrittweite verwendet werden. Der Rechenaufwand zu einem Simulationszeitschritt lässt sich dabei durch die Integrationsvorschrift aus Gleichung 2.6 ermitteln. Um die Echtzeitfähigkeit zu erreichen, muss entsprechend die Simulationsschrittweite angepasst werden. Allerdings kommt es bei zu großen Schrittweiten zu Instabilitäten bei der Berechnung, weshalb diese nicht beliebig groß gewählt werden können, um die Echtzeitfähigkeit sicherzustellen. Somit hängt es immer von den vorgegeben Ressourcen und den numerischen Eigenschaften ab, ob ein Modell aus numerischer Sicht echtzeitfähig ist.

2.2.4 Verfahren zur Modellvereinfachung

Ziel der Modellvereinfachung ist es, die numerischen Eigenschaften des abgeleiteten Modells des dynamischen Systems durch Modellvereinfachung so anzupassen, dass die Zeit, die zur Simulation benötigt wird, verkürzt wird. Im Folgenden werden verschiedene Ansätze vorgestellt, mit denen dieses Vorhaben erreicht werden kann. Die Vorstellung der Verfahren orientiert sich dabei im Wesentlichen an [Nil09], wonach die folgenden drei grundsätzlich unterschiedlichen Möglichkeiten zur Modellvereinfachung verwendet werden können:

- Mathematische Vereinfachung der Modellgleichungen
- Verwendung experimenteller Modellbildung
- Verwendung von Wissen über das dynamische System

Diese werden nun vorgestellt und es wird gezeigt, unter welchen Voraussetzungen sie sich anwenden lassen.

Mathematische Vereinfachung der Modellgleichungen

Die Verfahren zur Modellvereinfachung orientieren sich daran, ob die Modellgleichungen ein lineares oder nichtlineares dynamisches System beschreiben. Der Grundgedanke der meisten Verfahren ist dabei, die Anzahl der Modellgleichungen zu reduzieren. Dies wird durch eine Reduktion der Zustände erreicht, weshalb die Verfahren in der Literatur auch als Modell-Ordnungsreduktion-Verfahren zu finden sind [Ant05; Sch08]. Dabei sind die Verfahren zur Modellvereinfachung im linearen Fall gut erforscht und kommen besonders bei der Modellierung von Systemen mit verteilten Parametern zum Einsatz [Ant05]. Dies wird durch modale Reduktion oder balanciertes Abschneiden erreicht. Dabei wird der Einfluss der Gleichungen auf das Ausgangsverhalten untersucht, und ab einer vorgegebenen Schwelle werden die Gleichungen vernachlässigt.

Eine genaue mathematische Beschreibung der Verfahren findet sich beispielsweise in [Ant05; Sch08].
Ein numerisch robustes Vorgehen ist der Einsatz von Krylov-Unterraum-Verfahren, bei denen das vorliegende Gleichungssystem auf eine neue Basis projiziert wird. Dabei wird ebenso die Anzahl der Gleichungen reduziert. Der Nachteil dieser Verfahren ist, dass in der Anwendung sehr häufig Modelle nichtlinearer dynamischer Systeme vorliegen. Im nichtlinearen Fall ist das Vorgehen deutlich erschwert und ist Teil aktueller Grundlagenforschung. Der Grundgedanke der Verfahren ist dabei, bekannte Verfahren vom linearen auf den nichtlinearen Fall zu erweitern. Somit bilden die oben genannten Verfahren eine Basis für den nichtlinearen Fall.
Eine Möglichkeit zur Modellvereinfachung in diesem Fall ist die Linearisierung um einen bestimmten Arbeitspunkt mit anschließender Anwendung eines der oben genannten Verfahren für den linearen Fall. Der Nachteil ist hier, dass das abgeleitete Modell nur um den Arbeitspunkt herum gültig ist. Eine weitere Möglichkeit ist die Erweiterung des balancierten Abschneidens für nichtlineare Modellgleichungen. Allerdings müssen die Modellgleichungen aus Gleichung 2.1 in folgender Form geschrieben werden können

$$\begin{aligned} \dot{\mathbf{x}}(t) &= f(\mathbf{x}(t)) + g(\mathbf{x}(t)) \cdot \mathbf{u}(t) \\ \mathbf{y}(t) &= h(\mathbf{x}(t)). \end{aligned} \tag{2.7}$$

Ebenso kann die Methode des modalen Abschneidens erweitert werden, allerdings werden dazu häufig zusätzliche Simulationsdaten benötigt. Dies ist ein wesentlicher Unterschied zu vorher, da im linearen Fall die Modellvereinfachung ohne Simulation durchgeführt werden kann. Deshalb werden die Verfahren im nichtlinearen Fall unterschieden in simulationsbasierte und simulationsfreie [BBF14]. Dabei sind besonders simulationsfreie Verfahren im Fokus aktueller Forschungsarbeiten.

Da in dieser Arbeit die Anwendung im Vordergrund steht und die aktuell bekannten simulationsfreien Verfahren nur in Spezialfällen funktionieren, werden die folgenden zwei simulationsbasierten Verfahren vorgestellt, die aktuell am häufigsten zur numerischen Vereinfachung verwendet werden:

- Proper orthogonal decomposition:
  Für dieses Verfahren werden Simulationsdaten verwendet, um einen Raum mit niedriger Dimension zu finden, in den die gegebenen Zustandsgleichungen projiziert werden. Dazu wird eine Hauptkomponentenanalyse der gegeben Simulationsdaten durchgeführt, um die Zustände zu finden, die den Modellausgang nur gering beeinflussen. In [Ast04] wird dieses Verfahren an verschiedenen Gleichungen von Systemen mit verteilten Parametern vorgestellt. Die Ergebnisse hängen dabei stark von den vorgegebenen Simulationsdaten ab. So kann es passieren, dass bestimmte Zustände fälschlicherweise als unwichtig angesehen werden.
- Trajectory piecewise-linear model reduction:
  Bei diesem Verfahren ist die Idee, die Gleichungen aus Gleichung 2.1 entlang einer vorgegeben Trajektorie zu simulieren und zu linearisieren. Zusätzlich wird für die erstellten Modellgleichungen eine Modellvereinfachung mit einem der oben genannten Verfahren durchgeführt. Die Herausforderung bei diesem Verfahren ist die Vorgabe von Arbeitspunkten, um die linearisiert werden soll, und dass sich die Zustandsgleichungen auch an den gegebenen Arbeitspunkten linearisieren lassen.

Verwendung experimenteller Modellbildung

Eine weitere Möglichkeit zur Modellvereinfachung ist die Verwendung der experimentellen Modellbildung. Ziel ist es dabei, mit Hilfe von aufgezeichneten Simulationsdaten ein numerisch vereinfachtes Ersatzmodell zu erstellen. Dies ist dann hilfreich, wenn die Modellgleichungen nicht zugänglich sind oder

eine Form haben, für die die oben genannten Verfahren zur Vereinfachung der Modellgleichungen nicht angewendet werden können [Nil09]. Somit geht bei diesem Verfahren die Möglichkeit zur physikalischen Interpretation des Modells verloren. Bei der Verwendung experimenteller Modellbildung werden Simulationsdaten benötigt wie für die meisten zuvor vorgestellten Verfahren zur Vereinfachung von nichtlinearen Modellgleichungen. Dabei hängt die Modellgüte auch hier von der Qualität der vorgegebenen Daten ab. Die Hauptschwierigkeit liegt bei diesem Vorgehen darin, eine geeignete Modellstruktur zu finden, mit der das Eingangs- und Ausgangsverhalten abgebildet werden kann. Eine Modellvereinfachung kann dann erzielt werden, wenn der numerische Aufwand des erstellten Modells geringer ist. Ist es zusätzlich möglich, eine ausreichend hohe Simulationsschrittweite zu verwenden, so dass die Ergebnisse der Simulation innerhalb der gegebenen Zeit berechnet werden können, wird dabei auch ein echtzeitfähiges Modell erstellt.

Verwendung von Wissen über das dynamische System

Die Verwendung von Wissen über das dynamische System kann dazu beitragen, bestimmte physikalische Effekte zu vernachlässigen, von denen bekannt ist, dass diese das Verhalten nur schwach beeinflussen. Ein Beispiel ist die Trennung von schneller und langsamer Dynamik mit anschließender Approximation der schnellen Dynamik durch ein statisches System[5]. Eine weitere Möglichkeit ist das Zusammenfassen mehrerer Effekte zu einem Ersatz-Effekt. Ein Beispiel dafür ist die Verwendung von Ersatzwiderständen in Schaltungssimulationen [Dro03]. Die Verwendung von Wissen benötigt ein gewisses Maß an Erfahrung und somit nicht unmittelbar anwendbar. Sind diese Voraussetzungen erfüllt, sind die Ergebnisse ist den meisten Fällen sehr gut. Zudem führt

[5] Dies hat eine gewisse Ähnlichkeit zum Vorgehen beim oben genannten modalen Abschneiden, bei dem hohe Eigenfrequenzen identifiziert und abgeschnitten werden.

dieses Vorgehen zu einem größeren Verständnis, da wichtige Effekte bei einem bestimmten dynamischen System von unwichtigen getrennt werden. Allerdings wird hier nicht garantiert, dass eine numerische Vereinfachung eintritt [Nil09].

2.3 Modellierungs- und Simulationswerkzeuge

In diesem Abschnitt wird ein Überblick über graphische Modellierungswerkzeuge gegeben, die in der Funktionsentwicklung verwendet werden, um die Modellierung der physikalischen Umgebung des Steuergeräts zu erleichtern. Zudem sind in die meisten Werkzeuge auch verschiedene Simulationsverfahren integriert, so dass gleichzeitig Simulationen durchgeführt werden. Diese können in signalflussorientierte und energieflussorientierte Modellierungswerkzeuge eingeteilt werden.

2.3.1 Signalflussorientierte Modellierungswerkzeuge

In signalflussorientierten Modellierungswerkzeugen werden gewöhnliche Differentialgleichungen, lineare Übertragungsfunktionen, zeitdiskrete Gleichungen oder statische Zusammenhänge verwendet, um die physikalische Umgebung eines Steuergeräts oder mehrerer Steuergeräte zu modellieren. Die Modellgleichungen müssen durch Überlegungen abgeleitet werden oder sind bekannt. Dazu kann entweder theoretische oder experimentelle Modellbildung verwendet werden bzw. eine Kombination der beiden Verfahren. Bei detaillierten Modellen kann es jedoch passieren, dass eine hohe Anzahl von Signalen vorliegt und es somit eine Herausforderung ist, Teilmodelle richtig miteinander zu kombinieren. Dies erschwert die Anwendung der theoretischen Modellbildung in diesen Werkzeugen. Deshalb eignen sich signalflussorientierte Modelle eher für die Analyse der Wirkzusammenhänge der Komponenten eines mechatronischen Systems. Ein Beispiel für ein signalflussorientiertes Modellierungswerkzeug

ist Matlab/Simulink [Mat]. In Abbildung 2.6 ist ein harmonischer Oszillator mit externer Kraftanregung als signalflussorientiertes Modell zu sehen (siehe Gleichung A.3).

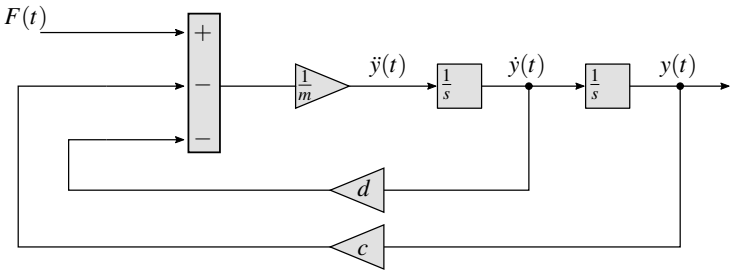


Abbildung 2.6: Harmonischer Oszillator als signalflussorientiertes Modell

Dieses Beispiel lässt erahnen, wie bei einer detaillierten Modellierung einer komplexen physikalischen Umgebung die Anzahl der Signale sehr schnell anwächst. Teilweise bieten jedoch entsprechende Modellierungswerkzeuge eine umfangreiche Modellbibliothek an, in der standardisierte Schnittstellen verwendet werden. Dadurch können auf anwenderfreundliche Weise detaillierte Teilmodelle miteinander verbunden werden, so dass der Aufwand bei der Modellerstellung sinkt [Güh02; Güh03].

2.3.2 Energieflussbasierte Modellierungswerkzeuge

Energieflussbasierte Modellierungswerkzeuge erlauben eine komfortablere Erstellung detaillierter Modelle einer komplexen physikalischen Umgebung. Dazu werden in diesen Modellierungswerkzeugen statt gerichteter Signale zwischen Teilmodellen ungerichtete Signale verwendet. Außerdem werden Flussvariablen und Potentialvariablen eingeführt, durch die die Energieerhaltung berücksichtigt wird (siehe Tabelle 2.1).

System	Flussvariable	Potentialvariable
Elektrisch	Elektrischer Strom	Elektrische Spannung
Mechanisch (Translation)	Geschwindigkeit	Kraft
Mechanisch (Rotation)	Winkelgeschwindigkeit	Drehmoment
Hydraulisch	Volumenstrom	Druckdifferenz
Thermodynamisch	Wärmestrom	Temperatur

Tabelle 2.1: Flussvariablen und Potentialvariablen der verschiedenen physikalischen Disziplinen

Dadurch ist eine einheitliche Modellierung unterschiedlicher physikalischer Domänen möglich. Die Modellgleichungen für die Simulation werden anschließend automatisiert aus dem erstellten Modell abgeleitet [Dro03; Ise08a]. Die Vereinheitlichung besteht darin, dass gezeigt wurde, dass jede physikalische Domäne als ein Netzwerk aus Kapazitäten, Induktivitäten und Widerständen modelliert werden kann. Weiter gibt es Quellen und Senken und es kann nichtlineares Verhalten einzelner Komponenten berücksichtigt werden. Mehr Details dazu finden sich in [Ise08a; Wel79]. Zudem kann diese Art der Modellbildung in der Theorie als Erweiterung der Lagrange-Mechanik angesehen werden, wie in [Jan09] beschrieben ist. Ein weiterer Vorteil energieflussbasierter Modellierungswerkzeuge ist die hohe Wiederverwendbarkeit der Komponentenmodelle. So kann zum Beispiel dasselbe Modell der Masse für viele unterschiedliche Simulationsfragestellungen genutzt werden.
Ein Beispiel für ein energieflussbasiertes Modellierungswerkzeug ist Dymola [Dym], das die Modellierungssprache Modelica [Mod] verwendet, die in [Til01; OHK09] sehr ausführlich beschrieben ist. Weitere Modellierungswerkzeuge, die ebenfalls diese Modellierungssprache verwenden, sind SimulationX [Sim] und MapleSim [Map].

In Abbildung 2.7 ist der harmonische Oszillator mit externer Kraftanregung als energieflussbasiertes Modell zu sehen. Dort ist zu sehen, wie das Modell einer Feder, einer Masse und einer Dämpfung miteinander verbunden werden.

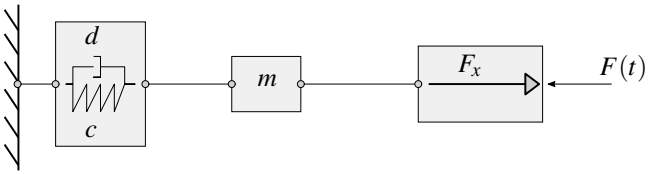


Abbildung 2.7: Harmonischer Oszillator als energieflussbasiertes Modell

Die dazugehörige gewöhnliche Differentialgleichung, wie sie in Abbildung 2.6 als signalflussorientiertes Modell umgesetzt zu sehen ist, wird automatisch abgeleitet. Bei komplexeren Modellen geschieht dies ebenso automatisiert, was eine große Erleichterung bedeutet. Diese Eigenschaft erlaubt eine komfortable Erstellung detaillierter Modelle der physikalischen Umgebung. Die so erstellten Modelle können dann verwendet werden, um verschiedene Konzepte zur Funktionsrealisierung zu betrachten.
Zum Abschluss dieses Abschnitts wird noch auf einen wichtigen Aspekt bei der Verwendung energieflussbasierter Modellierungswerkzeuge eingegangen. Da ein energieflussbasiertes Modell bidirektionale Verbindungen besitzt, ist keine Kausalität gegeben, wie sie für eine Simulation benötigt wird [Wel79; Til01; OHK09]. Deshalb müssen bei jedem energieflussbasierten Modell signalflussorientierte Anregungsgrößen eingefügt werden, so dass ein kausales Simulationsmodell erstellt werden kann. Beim Beispiel des harmonischen Oszillators ist das eine externe Kraft. Alternativ könnte zum Beispiel auch die Auslenkung vorgegeben werden.

2.3.3 Kopplung von Modellierungswerkzeugen

Das interdisziplinäre Zusammenspiel verschiedener Domänen bei mechatronischen Systemen trägt dazu bei, dass es eine Reihe unterschiedlicher Modellierungswerkzeuge gibt, die sich auf eine Domäne spezialisiert haben. Um dennoch das Zusammenspiel verschiedener Domänen in einer Simulation zu untersuchen, gibt es nach [Dro03; GKL06] vier verschiedene Möglichkeiten, Simulationsprogramme miteinander zu koppeln. Dadurch ist es möglich, von der Spezialisierung einzelner Modellierungswerkzeuge zu profitieren und dennoch das Zusammenspiel der einzelnen Domänen zu untersuchen. Dabei wird nach der Anzahl der verwendeten numerischen Integrationsverfahren und der Anzahl der verwendeten Modellierungswerkzeuge unterschieden.

Die klassische Simulation ist eine Möglichkeit und zeichnet sich durch Verwendung eines Modellierungswerkzeugs mit einem numerischen Integrationsverfahren aus.

Eine weitere Möglichkeit zur Verwendung von Modellen aus unterschiedlichen Modellierungswerkzeugen ist die Zusammenführung der Gleichungssysteme aus den jeweiligen Modellierungswerkzeugen und die Verwendung eines numerischen Integrationsverfahrens zur Lösung des entstehenden Gleichungssystems. Da bei manchen Modellierungswerkzeugen auf die Domäne angepasste numerische Integrationsverfahren verwendet werden, kann bei dieser Art der Kopplung von Modellierungswerkzeugen dieser Vorteil nicht ausgenutzt werden.

Deshalb gibt es eine andere Möglichkeit der Kopplung. So können nicht nur Modelle aus den Modellierungswerkzeugen verwendet werden, sondern auch die jeweiligen numerischen Verfahren. Dabei werden Schnittstellen definiert, über die bestimmte Größen ausgetauscht werden. Das Vorgehen dazu ist in [Dro03] genau beschrieben. Diese Möglichkeit zur Kopplung von Modellen wird in [Dro03;

GKL06] als Co-Simulation bezeichnet. In manchen Anwendungsfällen wird dadurch eine Echtzeitsimulation ermöglicht.

Die letzte Möglichkeit zur Kopplung von Modellen, die in den genannten Arbeiten beschrieben wird, ist die Modellteilung innerhalb eines Modellierungswerkzeugs und die Verwendung von angepassten numerischen Integrationsverfahren für die dabei entstehenden Teilmodelle. So kann zum Beispiel für ein Teilmodell eine andere Simulationszeitschrittweite gewählt werden als für ein weiteres Teilmodell, wodurch der numerische Aufwand bei der Simulation sinken kann.

2.4 Modellbildung des Fahrzeugs

Weiter oben wurde gezeigt, dass ein Fahrzeug aus verschiedenen mechatronischen Systemen besteht, die in Wechselwirkung miteinander treten. Die Wechselwirkungen finden dabei einerseits auf physikalischer und andererseits auf informationstechnischer Ebene statt. Außerdem wirkt der Fahrer auf das Verhalten des Fahrzeugs ein, indem er Lenkradwinkel, Gaspedalstellung, Bremspedalstellung oder Schalthebel betätigt. Zusätzlich muss in einer Simulation die Kraftübertragung auf die Straße berücksichtigt werden. Es gibt zahlreiche Simulationsumgebungen, in denen Teilmodelle in unterschiedlichen Detaillierungsgraden aus einer Bibliothek ausgewählt werden können [Güh02; Güh03; Rie+04; Sch13; Mat13]. Ziel dieser Bibliotheken ist es, dass Teilmodelle fachbereichsübergreifend ausgetauscht werden können. So können angepasste Modelle zur Beantwortung verschiedener Simulationsfragestellungen erstellt werden.
Im Folgenden wird der Aufbau eines Gesamtfahrzeugmodells genauer beschrieben, wie er in verschiedenen Modellierungswerkzeugen verwendet wird. In Abbildung 2.8 ist dieses Gesamtmodell zu sehen. Dabei wird zwischen den Modellen von Steuergeräten bzw. realen Steuergeräten und Modellen der physikalischen Umgebung unterschieden. Zudem sind die mechatronischen Kompo-

nenten als energieflussbasiertes Modell und als signalflussorientiertes Modell zu sehen.

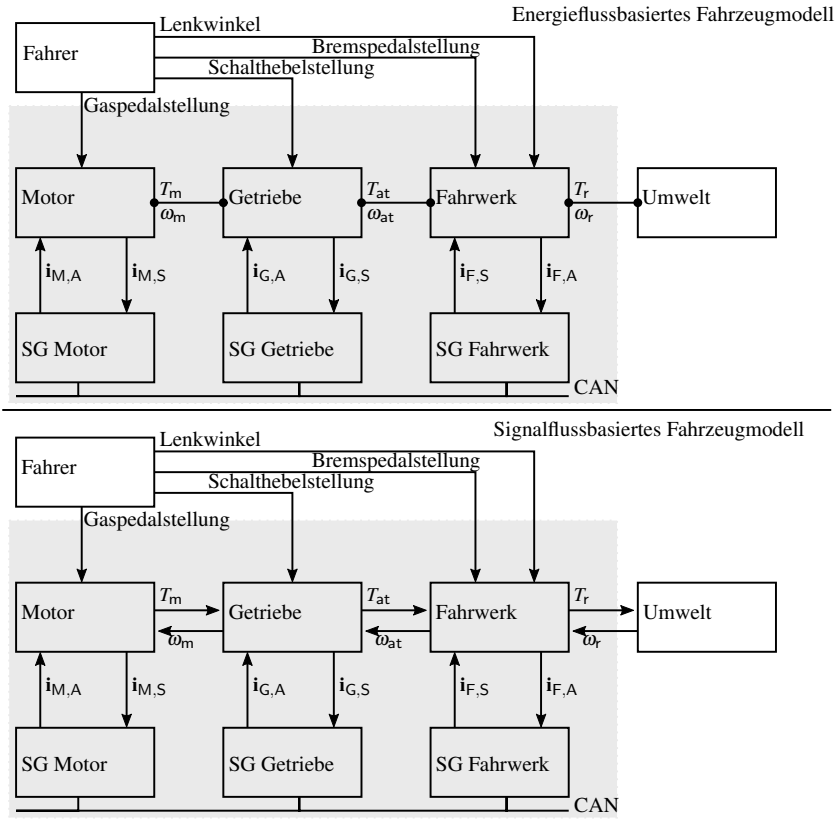


Abbildung 2.8: Teilmodelle eines Fahrzeugs in signalflussorientierter und in energieflussbasierter Darstellung

Solch ein Gesamtfahrzeugmodell besteht dabei aus folgenden Teilmodellen:

- Umweltmodell
- Fahrermodell
- Fahrzeugmodell:
  - Motormodell
  - Getriebemodell
  - Fahrwerksmodell
- Modelle von Steuergeräten bzw. reale Steuergeräte:
  - Modell des Motorsteuergeräts oder reales Motorsteuergerät
  - Modell des Getriebesteuergeräts oder reales Getriebesteuergerät
  - Modelle der Steuergeräte des Fahrwerks

Die aufgelisteten Teilmodelle werden im Folgenden näher beschrieben. Dabei wird auch darauf eingegangen, welche unterschiedlichen Detaillierungsgrade der Modelle für welche Simulationsfragestellungen verwendet werden. Die Beschreibung der Teilmodelle orientiert sich dabei an folgenden Arbeiten: [Amm97; HH03; Mat13].

2.4.1 Fahrermodell und Umweltmodell

Das Fahrer- und das Umweltmodell geben dem verwendeten Modell des Fahrzeugs einen Gesamtrahmen, durch den eine Untersuchung unterschiedlicher Simulationsfragestellungen möglich ist. So enthält das Umweltmodell den Verlauf der Straße, der zur Untersuchung von längs- und querdynamischen Eigenschaften des Fahrzeugs dient. Dazu werden verschiedene Kurvenverläufe, Reibwerte und Steigungsprofile vorgegeben. In den meisten Simulationsfragestellungen wird ein einfaches Fahrermodell verwendet, das durch vorgegebene Lenkradwinkel, Schalthebel, Gaspedal- und Bremspedalstellung auf das Fahrzeug einwirkt, um so eine fest definierte Auswahl von Fahrszenarien zu untersuchen.

Für spezielle Simulationsfragestellungen wird jedoch ein Fahrermodell benötigt, bei dem die Reaktion des Fahrers vom Verlauf der Straße und einem zusätzlich verwendeten Verkehr abhängt. Dies kann beispielsweise verwendet werden, um eine Funktion wie den Abstandsregelautomaten zu untersuchen, dessen Verhalten auch von der Reaktion des Fahrers abhängt. Diese Art der Simulation ist jedoch nicht im Fokus dieser Arbeit, und es werden nur vorher fest definierte Fahrszenarien berücksichtigt. Deshalb ist in Abbildung 2.8 das Fahrermodell vom Umweltmodell entkoppelt, und es besteht nur eine Verbindung zwischen Fahrzeug- und Umweltmodell.

2.4.2 Motormodell und Getriebemodell

Das Motormodell stellt das Drehmoment dar, das zum Beschleunigen des Fahrzeugs zur Verfügung steht. Dieses wird durch die Stellung des Gaspedals beeinflusst. Um das resultierende Drehmoment zu berechnen, gibt es unterschiedliche Detaillierungsgrade. Für die meisten Simulationsfragestellungen reicht ein einfaches, kennfeldbasiertes Modell, das einen statischen Zusammenhang zwischen Motordrehmoment, Motorwinkelgeschwindigkeit und Öffnung der Drosselklappe beschreibt. Um das dynamische Verhalten des Motors zu berücksichtigen, das bei einem Lastwechsel auftritt, wird dieses häufig als Verhalten erster Ordnung approximiert, und das kennfeldbasierte Modell wird entsprechend erweitert. Diese Modelle können dann effizient als signalflussorientiertes Modell umgesetzt werden.
Zur Bestimmung der Parameter dieser Modelle kommt die experimentelle Modellbildung zum Einsatz. Die dafür benötigten Daten werden entweder am Prüfstand oder durch Simulation detaillierter Motormodelle gewonnen. Die dazu notwendigen detaillierten Modelle werden in den entsprechenden Fachabteilungen erstellt. Diese Modelle berücksichtigen dabei weitere physikalische Effekte

zur Berechnung des dynamischen Drehmoments, wie Verbrennungseffekte oder Strömungseffekte, wobei Modelle mit verteilten Parametern entstehen.
In [Nil09] ist beschrieben, wie durch Verwendung der oben genannten Methoden der Modellvereinfachung ein energieflussbasiertes Modell numerisch vereinfacht werden kann. Die Modellierung des Getriebes ermöglicht es, Schaltvorgänge und statische Übersetzungen von Winkelgeschwindigkeit und Drehmoment zu berücksichtigen. In Kapitel 3.2 wird die Modellierung von Stufenautomaten und die numerische Vereinfachung von Getriebemodellen ausführlich behandelt.

2.4.3 Fahrwerkmodell

Das Fahrwerkmodell in Abbildung 2.8 ist eine Zusammenfassung mehrerer Teilmodelle und besteht aus einem Reifenmodell, einem Modell der Bremse und einem Modell des Aufbaus des Fahrwerks. In diesem Abschnitt werden diese drei Teilmodelle des Fahrwerkmodells näher beschrieben.
Der Aufbau des Fahrwerks wird je nach Simulationsfragestellung durch ein einfaches Punkt-Masse-Modell, ein Ein-Spur-Modell oder ein Zwei-Spur-Modell beschrieben. Während das einfache Punkt-Masse-Modell zur Untersuchung der Längsdynamik verwendet wird, können das Ein-Spur-Modell und das Zwei-Spur-Modell zusätzlich zur Untersuchung querdynamischer Effekte verwendet werden [Amm97].
Diese Modelle lassen sich ebenfalls gut als signalflussorientiertes Modell umsetzen. Je nach Detaillierungsgrad werden dabei auch die Nick- und Wankdynamik berücksichtigt. Für höhere Detaillierungsgrade wird der Aufbau des Fahrwerks durch ein Mehrkörpermodell beschrieben, das durch energieflussbasierte Modellierungswerkzeuge erstellt werden kann.
Das Reifenmodell modelliert die Kraftübertragung auf die Straße. Im einfachsten Fall wird diese durch einen starren Reifen dargestellt, der einem idealen

Kraftüberträger entspricht. Für fahrdynamische Untersuchungen findet das Pacejka-Reifen-Modell eine breite Verwendung, durch das eine dynamische Kraftübertragung berücksichtigt wird.
Weiter wird zur Simulation von fahrdynamischen Szenarien ein Modell der Bremse benötigt. Im einfachsten Fall ist dies ein dem Antriebsmoment entgegengesetzt wirkendes Drehmoment. Detailliertere Modelle berücksichtigen zusätzlich dynamische Effekte der elektrohydraulischen Ansteuerung. Diese sind notwendig, um eine Bremsregelung zu untersuchen bzw. zu optimieren, wie es in [Amm97; Hal+99; HH03] beschrieben ist.

2.4.4 Steuergeräte und Modelle von Steuergeräten

Bisher wurde nur auf die Modellierung der physikalischen Umgebung der Steuergeräte eingegangen. Um das Verhalten eines mechatronischen Systems zu untersuchen, werden auch Modelle der Steuergeräte benötigt, wie weiter oben gezeigt wurde. So wird je nach Simulationsfragestellung ein Motorsteuergerät oder ein Modell des Motorsteuergeräts verwendet. Durch dieses Modell kann eine Regelung des Drehmoments des Motors berücksichtigt werden. Besitzt das Fahrzeugmodell ein Automatikgetriebe, wird ein Getriebesteuergerät oder ein Modell des Getriebesteuergeräts benötigt. Es werden dann zum Beispiel die Schaltzeitpunkte und die Durchführung von Schaltungen berücksichtigt, die durch die Funktionssoftware gegeben sind. Näheres dazu findet sich in Kapitel 3.
Weiter gibt es eine Reihe von Steuergeräten, die im Fahrwerk verbaut sein können. Ein Beispiel dafür ist das elektronische Stabilitätsprogramm, das über eine elektrohydraulische Ansteuerung die Bremskraft regulieren kann. Für die Entwicklung einer Funktion, die durch Vernetzung von bestimmten, bestehenden Steuergeräten entstehen soll, werden entsprechende Modelle der Steuergeräte benötigt, damit das Verhalten in einer Simulation untersucht werden kann.

2.4.5 Klassifizierung von Simulationsfragestellungen

Die genannten Beispiele zeigen, dass die Simulationsfragestellungen aufgeteilt werden können in Fragestellungen, die für die Fachabteilungen relevant sind, und in Fragestellungen, die über mehrere Fachabteilungen hinweg von Bedeutung sind. Beispiele für Fachabteilungen sind Abteilungen, die für die Entwicklung von Getrieben oder von Motoren zuständig sind. In diesen werden detaillierte Modelle verwendet, die eine fundierte Analyse des dynamischen Verhaltens ermöglichen und somit zur Beantwortung der Fragestellungen verwendet werden, die hauptsächlich für die Fachabteilung relevant ist. So können auch Grenzfälle untersucht oder verschiedene Konzepte zur Ansteuerung bewertet werden. Auf diese Weise entstehen Modelle, die das Verhalten des von der Fachabteilung entwickelten mechatronischen Systems valide beschreiben.
Um Simulationsfragestellungen zu beantworten, die über mehrere Fachabteilungen hinweg von Bedeutung sind, stehen die Untersuchung des Zusammenspiels der verschiedenen mechatronischen Systeme und die Untersuchung auf Gesamtfahrzeugebene im Vordergrund. Beispiele für diese Art von Simulationsfragestellungen kommen bei der Systemintegration vor (siehe Abbildung 2.4) und sind in [Güh02; Güh03; LG03; Rie+04; Sch13] ausführlich beschrieben. Um die Teilmodelle für eine Simulation auf Fahrzeugebene zu gewinnen, können die oben genannten Methoden zur Modellvereinfachung zum Einsatz kommen, wie zum Beispiel bei der Erstellung von Motormodellen. Eine andere Möglichkeit ist die weiter oben beschriebene Co-Simulation bzw. die Verwendung der erstellten Teilmodelle der Fachabteilungen in einem kompatiblen Modellierungswerkzeug.
Häufig werden jedoch angepasste Teilmodelle für die Simulation auf Gesamtfahrzeugebene erstellt, wodurch zusätzlicher Aufwand für die Modellierung entsteht. Dabei besteht jedoch die Gefahr, dass für das Verhalten der vernetzten mechatronischen Systeme wichtige Effekte nicht ausreichend genau berücksich-

tigt werden. Dies zeigt, dass ein genereller Forschungsbedarf besteht, die Methoden zur numerischen Vereinfachung bzw. die Methoden zum Austausch von Modellen und zur Kopplung von Simulationswerkzeugen weiterzuentwickeln. Im nächsten Kapitel wird darauf eingegangen, inwiefern dieser Forschungsbedarf bei der Modellierung von Stufenautomaten besteht.

2.5 Fazit

In diesem Kapitel wurde gezeigt, wie Modellbildung und Simulation verwendet werden können, um eine frühzeitige Bewertung von Konzepten zur Funktionsrealisierung durchzuführen bzw. eine frühzeitige Erkennung von Fehlern in der Softwareentwicklung zu ermöglichen. Dabei kommen Modellbibliotheken zum Einsatz, die einen Austausch von Modellen zwischen den Entwicklungsabteilungen erlauben. In den Fachabteilungen werden zur Erstellung detaillierter Modelle häufig energieflussbasierte Modellierungswerkzeuge verwendet, da diese eine komfortable Erstellung ermöglichen. Mit diesen Modellen ist anschließend eine fundierte Untersuchung verschiedener Wechselwirkungen zwischen Funktion und physikalischer Umgebung möglich und somit auch eine Bewertung verschiedener Konzepte zur Funktionsrealisierung. Nachteilig an diesen Modellen ist jedoch, dass sie meist numerisch anspruchsvoll sind, so dass für die anschließende Systemintegration vereinfachte Modelle benötigt werden. Außerdem werden häufig signalflussorientierte Werkzeuge zur Systemintegration verwendet. Dadurch ist auch die direkte Verwendung detaillierter Modelle der jeweiligen Fachabteilungen meist nicht ohne weiteres möglich. So kann es vorkommen, dass in manchen Anwendungsfällen auch durch die Verwendung einer Co-Simulation keine numerische Vereinfachung erfolgt. Für solche Fälle ist es deshalb wichtig, dass Methoden zur numerischen Vereinfachung zur Verfügung stehen, wie sie oben beschrieben wurden. Dabei gilt es jedoch zu beachten, dass es zur numerischen Vereinfachung nichtlinearer dynamischer Systeme kein

Standardvorgehen gibt, so dass im Einzellfall überprüft werden muss, welche der Methoden zur numerischen Vereinfachung angewendet werden kann.

3 Aufbau und Modellierung von Stufenautomaten

In diesem Kapitel wird zunächst der aktuelle Stand der Wissenschaft zum Aufbau und zur Entwicklung von Stufenautomaten zusammengefasst. Da Stufenautomaten zu den automatischen Lastschaltgetrieben gehören, wird dabei das Prinzip von Lastschaltungen thematisiert. Anschließend wird auf den aktuellen Stand der Forschung zur Modellbildung von Stufenautomaten eingegangen. Genauer wird gezeigt, welche unterschiedlich detaillierten Modelle von Stufenautomaten erstellt werden können. Dabei wird thematisiert, für welche Simulationsfragestellungen die jeweiligen Modelle verwendet werden, und welche Phasen der Entwicklung damit unterstützt werden können. Weiter werden die numerischen Eigenschaften der verschiedenen Modelle beschrieben, und es wird darauf eingegangen, wie diese einen fachbereichsübergreifenden Austausch von Getriebemodellen erschweren.
Deshalb werden verschiedene Lösungsansätze zur numerischen Vereinfachung beschrieben, wobei gezeigt wird, dass eine automatische numerische Vereinfachung bei Getriebemodellen bisher nur teilweise existiert und es einen Bedarf zur Weiterentwicklung von Methoden zur numerischen Vereinfachung gibt.

3.1 Grundlagen automatischer Lastschaltgetriebe

Neben den Stufenautomaten zählt auch das Doppelkupplungsgetriebe zu den automatischen Lastschaltgetrieben. Diese Getriebe zeichnen sich dadurch aus,

dass keine Zugkraftunterbrechung nötig ist, um Schaltvorgänge beim Anfahren durchzuführen [Fis+12], wodurch der Fahrkomfort erhöht wird. Zudem ist es durch ein entsprechendes Schaltprogramm und entsprechende Funktionsrealisierung möglich, den Kraftstoffverbrauch zu senken.

3.1.1 Das Prinzip von Lastschaltungen

Das Prinzip von Lastschaltvorgängen beruht auf einem Getriebe mit zwei Gängen mit den beiden Übersetzungen R_1 und R_2. Die Übersetzung R beschreibt das Verhältnis der Winkelgeschwindigkeit des Abtriebs und des Motors und ist gegeben durch

$$R = \frac{\omega_{at}}{\omega_m}. \tag{3.1}$$

Die Kupplungen verbinden Motorwelle und Antriebswelle des Fahrwerks. Durch Kraftschluss werden die Kupplungen so geschlossen, dass entweder der erste Gang oder der zweite Gang eingelegt ist. Um diesen Kraftschluss herzustellen, werden die Kupplungen so stark zusammengepresst, dass das dabei entstehende Reibmoment dafür ausreicht. In Abbildung 3.1 ist das Schema eines Lastschaltgetriebes zu sehen.

Um den Gang zu wechseln, muss eine Kupplung geöffnet und die andere geschlossen werden. Je nach Fahrsituation können Schaltungsart und Schaltungsdurchführung klassifiziert werden. Dazu werden folgende drei Kriterien verwendet [Fis+12; Koc01]:

1. Betrachtung, ob die Übersetzung des Zielgangs kleiner oder größer ist als die des aktuellen Gangs
2. Betrachtung, ob das Motormoment größer oder kleiner als null ist
3. Vergleich der Richtung der Änderung der Winkelgeschwindigkeit des Motors und der gewünschten Änderung der Winkelgeschwindigkeit des Gangwechsels

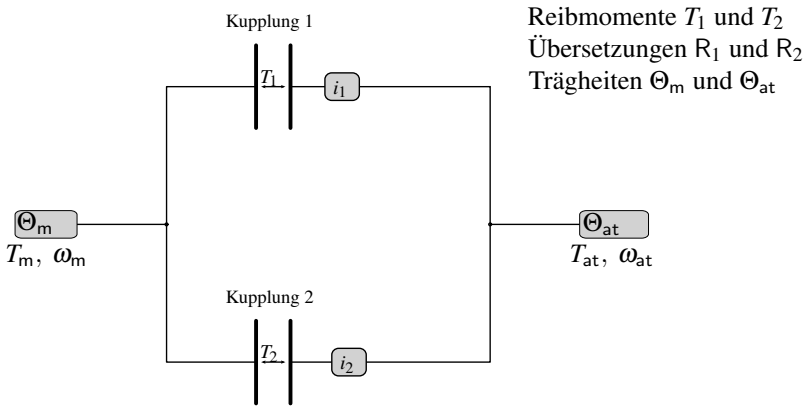


Abbildung 3.1: Schema eines Lastschaltschaltgetriebes mit zwei Gängen

Die daraus resultierenden Schaltungsarten und die dazugehörige Schaltungsdurchführung sind in Tabelle 3.1 zusammengefasst und werden im Folgenden näher beschrieben.

	$T_\mathsf{m} > 0$	$T_\mathsf{m} \leq 0$
$\mathsf{R}_\mathsf{neu} < \mathsf{R}_\mathsf{alt}$	Zughochschaltung	Schubhochschaltung
Durchführung	Überhöhungsschaltung	Freigabeschaltung
$\mathsf{R}_\mathsf{neu} > \mathsf{R}_\mathsf{alt}$	Zugrückschaltung	Schubrückschaltung
Durchführung	Freigabeschaltung	Überhöhungsschaltung

Tabelle 3.1: Klassifizierung von Schaltvorgängen im Automatikgetriebe

Das letzte der oben genannten Kriterien wird zur Unterscheidung zwischen einer Freigabeschaltung und einer Überhöhungsschaltung verwendet. Bei einer Freigabeschaltung erfolgt die Änderung der beiden Winkelgeschwindigkeiten

in die gleiche Richtung. Während des Schaltvorgangs wird der Motor vom Getriebe getrennt und die Zielwinkelgeschwindigkeit stellt sich von alleine ein. Bei einer Überhöhungsschaltung erfolgt dagegen die Änderung der beiden Winkelgeschwindigkeiten in unterschiedliche Richtungen. Es wird ein zusätzliches Moment benötigt, um die Winkelgeschwindigkeit des Motors an die Zielwinkelgeschwindigkeit anzupassen. Die ersten beiden Kriterien werden dazu verwendet, die Schaltungsart zu charakterisieren. Damit ergeben sich folgende vier Schaltungen [Koc01]:

1. Zughochschaltung:
   Die Zughochschaltung wird als **Überhöhungsschaltung** durchgeführt. Dies trifft auf Beschleunigungsvorgänge zu, bei denen das Motormoment größer als null ist.
2. Schubhochschaltung:
   Die Schubhochschalung wird als **Freigabeschaltung** durchgeführt. Dies trifft auf das Ende von Beschleunigungsvorgängen oder Bergabfahrten zu, bei denen das Motormoment niedrig bzw. kleiner als null ist.
3. Zugrückschaltung:
   Die Zugrückschaltung wird als **Freigabeschaltung** durchgeführt. Dies tritt bei der Einleitung eines Beschleunigungsvorgangs, wobei sich Winkelgeschwindigkeit des Motors erhöht, um die geforderte Leistung bereitzustellen.
4. Schubrückschaltung:
   Die Schubrückschaltung wird als **Überhöhungsschaltung** durchgeführt. Diese Situation tritt ein, wenn das Fahrzeug abgebremst wird.

In Abbildung 3.2 sind die Verläufe der Reibmomente in den Kupplungen, der Winkelgeschwindigkeiten und der Drehmomente bei einer Zugrückschaltung und einer Zughochschaltung dargestellt.

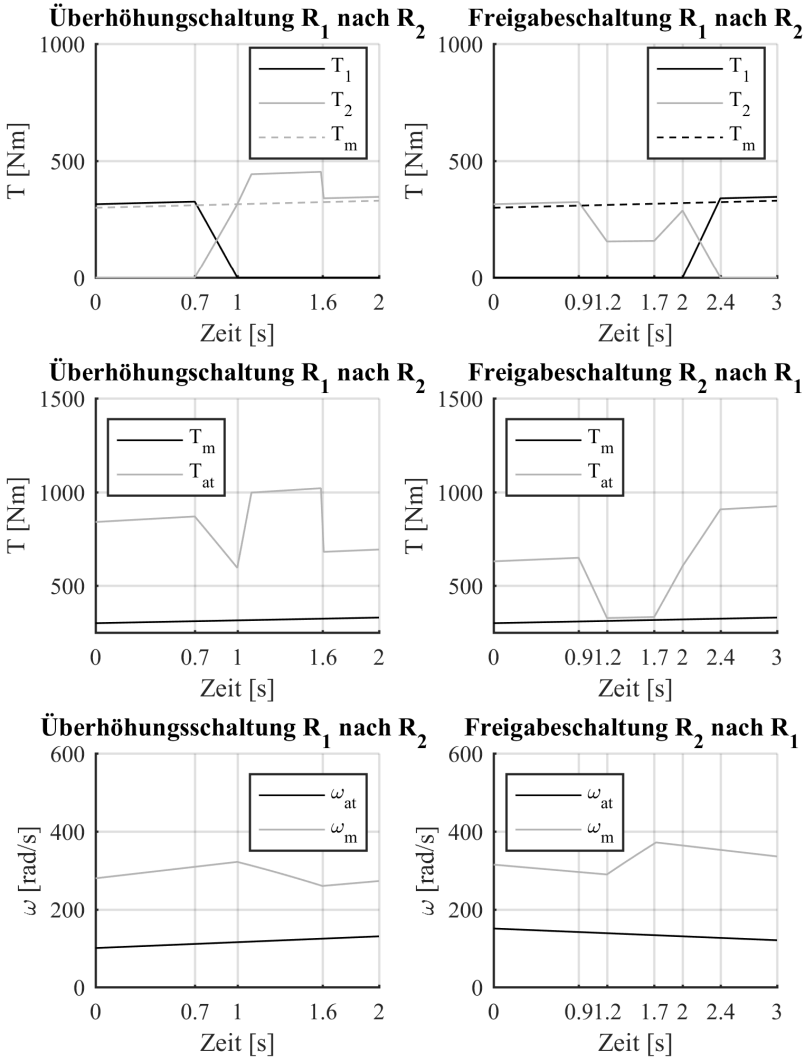


Abbildung 3.2: Schematischer Verlauf der Reibmomente, Winkelgeschwindigkeiten und Drehmomente bei Überhöhungs- und Freigabeschaltungen

Bei der Zughochschaltung ist das Überhöhungsmoment zu sehen, das zur Anpassung der Winkelgeschwindigkeit des Motors benötigt wird. Außerdem ist zu erkennen, wie die Reibmomente in den Kupplungen den Schaltvorgang beeinflussen. Die Anfangsphase dient dazu, dass das Motormoment von der Kupplung, die hinzugeschaltet wird, übertragen wird. Das Überhöhungsmoment wird dabei durch das Reibmoment der zuschaltenden Kupplung zur Verfügung gestellt. Die Höhe des Überhöhungsmoments beeinflusst dabei die Zeit, die zur Anpassung der Winkelgeschwindigkeit des Motors benötigt wird. Bei der Zugrückschaltung wird deutlich, wie die Winkelgeschwindigkeit des Motors sich an die neue Winkelgeschwindigkeit anpasst. Dazu wird das Reibmoment der aktiven Kupplung abgesenkt, um diese Anpassung zu ermöglichen. Am Ende der Schaltung wird die zuvor inaktive Kupplung hinzugeschaltet und die zuvor aktive Kupplung abgeschaltet.

3.1.2 Aufbau von Stufenautomaten

Stufenautomaten bestehen im Allgemeinen aus mehr als zwei Gangstufen, jedoch ist die Bauweise so, dass das Prinzip der eben vorgestellten Lastschaltung weiterhin seine Gültigkeit besitzt [Fis+12]. Die Komponenten des Stufenautomaten sind ein hydrodynamischer Wandler, die Getriebemechanik, die elektrohydraulische Ansteuerung und ein Getriebesteuergerät. In Abbildung 3.3 ist eine Innensicht des ZF-8HP-Automatikgetriebes zu sehen. Dort ist zu sehen, wie die Planetenstufen und die Schaltelemente verbaut sind. Außerdem sind der Wandler mit Überbrückungskupplung am Getriebeeingang und das Getriebesteuergerät mit elektrohydraulischer Ansteuerung zu sehen.
Die Aufgaben und die Funktionsweise dieser Komponenten von Stufenautomaten werden im Folgenden genauer beschrieben.

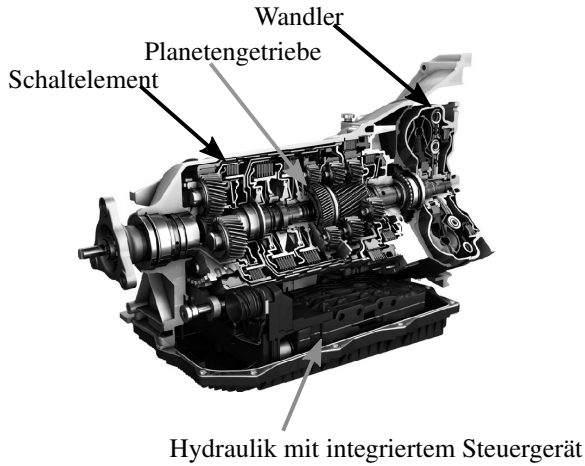


Abbildung 3.3: Innenansicht des ZF-8HP-Getriebes (aus [Fis+12])

Hydrodynamischer Wandler

Der Wandler dient als Anfahrelement und bietet hohe Komforteigenschaften. Zur Übertragung des Motordrehmoments auf das Getriebe werden eine Pumpe und eine Turbine verwendet. Die Turbine ist dabei mit der Antriebswelle des Motor verbunden und die Turbine mit dem Getriebe. Die Übertragung des Motordrehmoments auf das Getriebe findet durch eine Differenz der Winkelgeschwindigkeiten statt [Fis+12]. Zusätzlich wird ein Leitrad verwendet, dass mit einem Freilauf ausgestattet ist. Dies sorgt dafür, dass der Wandler bei Anfahrvorgängen für eine zusätzliche Übersetzung des Motordrehmoments sorgt und damit sehr effizient Anfahrvorgänge ermöglicht.
Ist ein Anfahrvorgang abgeschlossen, haben sich die Winkelgeschwindigkeiten von Turbine und Pumpe angepasst, wodurch die Effizienz des Wandlers sinkt. Um dem entgegenzuwirken, wird zusätzlich eine Wandlerüberbrückungskupp-

lung[1] (WÜK) verwendet, um den Motor starr mit dem Getriebe verbinden zu können. Näheres zum Ablauf der dazu benötigten Regelung der WÜK findet sich in [Gru10; Mat13].

Getriebemechanik

Die Getriebemechanik besteht aus Planetenstufen, die über Wellen und Schaltelemente miteinander verbunden sind. Als Schaltelemente werden Kupplungen und Bremsen verwendet. Mit diesen ist es möglich, die Planetenstufen auf unterschiedliche Art und Weise über Kupplungen zu verbinden bzw. am Getriebegehäuse durch Bremsen abzustützen. So kann durch Schließen bestimmter Kupplungen bzw. Bremsen eine bestimmte Übersetzung i eingestellt werden. Die Getriebemechanik wird häufig mit einem Radsatz schematisch dargestellt. Dies wird im Folgenden an einem Beispiel illustriert.
Um besser zu verstehen, wie die Getriebemechanik funktioniert, ist in Abbildung 3.4 der Radsatz des oben erwähnten ZF-8HP-Getriebes zu sehen. Dieser beschreibt schematisch, wie die Wellen der Planetenstufen mit Kupplungen verbunden werden, und welche Wellen durch Bremsen festgehalten werden müssen, um die verschiedenen Gangstufen zu realisieren. Daraus ergibt sich dann die Schaltmatrix, aus der abgelesen werden kann, welche Schaltelemente für eine bestimmte Gangstufe geschlossen werden müssen [Fis+12; Kir07; Kuc+10]. Für das hier gezeigte Beispiel ist die zugehörige Schaltmatrix in Tabelle 3.2[2] zu sehen. Zudem gibt sie an, welche Schaltelemente bei einem Gangwechsel angesteuert werden müssen. Soll beispielsweise vom zweiten in den dritten Gang geschaltet werden, müssen die Bremse B_{A} abgeschaltet und die Kupplung K_{C} hinzugeschaltet werden.

[1] Wandlerüberbrückungskupplung ...WÜK
[2] Dabei wird folgende Schreibweise verwendet: Elemente B_{I} Bremsen, Elemente K_{I} Kupplungen

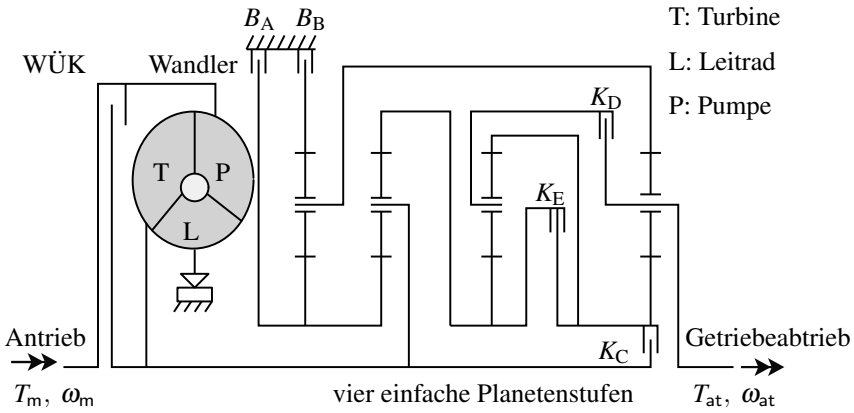


Abbildung 3.4: Radsatz des ZF-8HP-Getriebes (aus [Fis+12])

Gang	B_A	B_B	K_C	K_D	K_E	Übersetzung
1	1	1	1	0	0	4,70
2	1	1	0	0	1	3,13
3	0	1	1	0	1	2,10
4	0	1	0	1	1	1,67
5	0	1	1	1	0	1,29
6	0	0	1	1	1	1,00
7	1	0	1	1	0	0,84
8	1	0	0	1	1	0,67
R	1	1	0	1	0	-

Tabelle 3.2: Schaltmatrix des ZF-8HP-Getriebes (aus [Fis+12])

Elektrohydraulische Ansteuerung

Durch die elektrohydraulische Ansteuerung können die Schaltelemente gezielt angesteuert werden, um Schaltungen durchführen zu können. Diese dient also als Schnittstelle zwischen Getriebesteuergerät und Getriebemechanik. Dazu wird die elektrohydraulische Ansteuerung verwendet, die ein gezieltes Öffnen und Schließen der Schaltelemente ermöglicht.
In [Rei10] findet sich eine genaue Beschreibung der elektrohydraulischen Ansteuerung. Zur Versorgung des hydraulischen Kreislaufs mit Getriebeöl wird eine Pumpe verwendet, die über die Getriebeeingangswelle angetrieben wird. Zusätzlich kann eine elektrische Pumpe hinzugeschaltet werden, um auch im Stillstand die Hydraulik mit Öl zu versorgen. Zur Vollständigkeit wird an dieser Stelle erwähnt, dass die Hydraulik die Schmierung und Kühlung des Getriebes übernimmt [Fis+12].

Funktionen des Getriebesteuergeräts

Die Hauptaufgaben des Getriebesteuergeräts sind die Erkennung von Schaltzeitpunkten und die Ansteuerung der Schaltelemente zur Durchführung von Lastschaltungen. Die Schaltzeitpunkte sind dazu in Kennfeldern hinterlegt. Diese werden verändert, wenn eine Berg- bzw. Talfahrt durchfahren wird. Um die Schaltzeitpunkte zu beeinflussen, stehen dem Fahrer meist mehrere Schaltprogramme zur Verfügung. So kann er beispielsweise zwischen einer sportlichen oder einer sparsamen Auslegung der Schaltzeitpunkte entscheiden. Die Funktionen der Software des Getriebesteuergeräts sind folgende:

- Bestimmung von Schaltzeitpunkten:
  Die Bestimmung von Schaltzeitpunkten erfolgt in enger Abstimmung mit der Charakteristik des Verbrennungsmotors. Für die Wahl der Schaltzeitpunkte wird unterschieden, je nachdem ob eher eine sportliche oder eine

ökonomische Ausrichtung gewählt wurde. Weiter werden für Berg- und Talfahrten jeweils andere Schaltzeitpunkte verwendet. Die Schaltzeitpunkte werden im Getriebesteuergerät in Form von Schaltkennlinien abgelegt. Wann eine Schaltung durchgeführt werden soll, hängt von der Gaspedalstellung und der Geschwindigkeit des Fahrzeugs ab. Mehr Details zur Bestimmung von Schaltzeitpunkten finden sich in [Fis+12].

- Durchführung von Lastschaltungen:
  Welche Schaltelemente zur Durchführung eines bestimmten Schaltvorgangs zu betätigen sind, ist der Schaltmatrix zu entnehmen. Durch Ansteuerung des Stufenautomaten über die elektrohydraulische Aktuatorik werden die Lastschaltungen zum größten Teil gesteuert durchgeführt. Dazu werden Kennlinien zur Ansteuerung im Steuergerät hinterlegt. Zusätzlich gibt es verschiedene Kriterien, die in die Auswahl eines bestimmten Verlaufs eingehen. Dies kann zum Beispiel die Dauer für den Schaltvorgang sein. Zusätzlich gibt es Ausführungen, bei denen der Druck in der Getriebehydraulik während des Schaltvorgangs geregelt wird. Zudem wird die Ansteuerung der WÜK übernommen.
- Diagnosefunktionen:
  Damit soll sichergestellt werden, dass alle Bauteile des Getriebes noch korrekt funktionieren. Somit können Sicherheitsanforderungen erfüllt werden. Wird ein Ausfall eines Bauteils bemerkt, dann wird dies gemeldet.
- Treiber und Signalverarbeitung:
  Im Steuergerät werden die ankommenden Sensorsignale und die Ansteuerströme zur Ansteuerung der Schaltelemente aufbereitet. Dies stellt den sogenannten hardwarenahen Anteil der Software dar [Rei10].

Die ersten beiden Punkte werden durch die Funktionssoftware realisiert. Dabei ist die Entwicklung der Durchführung von Lastschaltungen wesentlich aufwändiger als die Festlegung von Schaltzeitpunkten. In [Fis+12] ist beschrieben,

welche weiteren Aspekte bei der Entwicklung beachtet werden müssen. Für das Ziel dieser Arbeit, nämlich die Erarbeitung einer Methode zur numerischen Vereinfachung von Getriebemodellen, werden diese Aspekte jedoch nicht im Detail benötigt, weshalb an dieser Stelle darauf nicht weiter eingegangen wird. Allerdings gibt es typische Ablaufschemata bei der Durchführung von Lastschaltungen, die im Folgenden beschrieben werden.

3.1.3 Steuerung und Regelung von Stufenautomaten

Zum Abschluss wird auf die Steuerung und Regelung von Lastschaltungen eingegangen. Dabei werden typische Sequenzen der Ansteuerströme durchlaufen, um Lastschaltungen durchzuführen. Diese Sequenzen werden im Folgenden Schaltablauf genannt. Außerdem werden dabei die typischen Zeitkonstanten betrachtet, da diese für die Modellbildung von Stufenautomaten und die anschließende Simulation besonders wichtig sind.

Ein typischer Schaltablauf kann in folgende drei Phasen gegliedert werden.

Vorfüllphase

Ziel der Vorfüllphase ist es, den Kolben der zuschaltenden Kupplung so vorzubereiten, dass diese das zu übertragende Drehmoment übernehmen kann. Dazu wird der Volumenstrom im Kupplungskolben der zuschaltenden Kupplung für kurze Zeit stark erhöht, so dass der Kolben gefüllt ist und die Lamellenpakete der Kupplung anliegen. Ist dies erreicht, wird der Volumenstrom so weit abgesenkt, dass die Lamellenpakete weiterhin anliegen. Die Vorfüllphase stellt für die spätere Modellierung durch neuronale Netze eine Herausforderung dar, da sich hier die Ansteuerströme stark ändern, dies jedoch noch keine Auswirkung auf das Schaltverhalten hat und entsprechend in den neuronalen Netzen

berücksichtigt werden muss. Ein typischer Wert für die Dauer der Vorfüllphase ist etwa $0,1[\mathrm{s}]$ [Mat13].

Übergabe des Drehmoments

Nachdem die Vorfüllphase beendet ist, beginnt die Übergabe des Drehmoments von der abschaltenden auf die zuschaltende Kupplung. Dazu wird der Druck im Kupplungskolben abgesenkt bzw. aufgebaut. Wird das Drehmoment vollständig von der zuschaltenden Kupplung getragen, beginnt die Phase zur Anpassung der Winkelgeschwindigkeit. Während dieses Vorgangs kann zum Beispiel auch das oben beschriebene Überhöhungsmoment beobachtet werden. Somit tritt etwa $0,1[\mathrm{s}]$ nach Einleitung des Schaltvorgangs eine Veränderung am Drehmoment der Antriebswelle ein.

Anpassung der Winkelgeschwindigkeit

Dazu wird der Druck im Kolben der zuschaltenden Kupplung so erhöht, dass das für die Schaltung benötigte Reibmoment zur Verfügung gestellt werden kann. Dieser Vorgang wird Synchronisierung genannt [Mat13; Fis+12]. Für das Schaltverhalten bedeutet dies, dass sich die Auswirkung sich ändernder Ansteuerströme auf die Winkelgeschwindigkeit etwas später zeigt als die Auswirkungen auf das Drehmoment. Die Winkelgeschwindigkeit ändert sich dabei etwa $0,3[\mathrm{s}]$, nachdem der Schaltvorgang eingeleitet wurde. Die Schaltung ist beendet, wenn die Winkelgeschwindigkeit angepasst ist. Typische Werte für die gesamte Dauer einer Schaltung liegen zwischen $0,5[\mathrm{s}]$ und $1[\mathrm{s}]$, je nach Auslegung der Schaltvorgänge [Mat13; Fis+12].

3.2 Gruppierung von Simulationsfragestellungen im Entwicklungsprozess

In diesem Abschnitt wird auf verschiedene Ansätze zur Modellierung des Stufenautomaten eingegangen und auf die jeweiligen Simulationsfragestellungen der simulationsgestützten Entwicklung. Dazu wird zunächst auf die Simulationsfragestellungen im Entwicklungsprozess eingegangen. Anschließend wird gezeigt, welche Modelle dabei verwendet werden.
Die Simulationsfragestellungen können nach der Phase, wann diese im Entwicklungsprozess beantwortet werden sollen, einer der folgenden vier Kategorien zugeordnet werden, wobei das V-Modell aus Kapitel 2 zu Grunde gelegt wird:

1. Simulationsfragestellung zur Funktionsentwicklung
2. Simulationsfragestellung zur funktionalen Absicherung
3. Simulationsfragestellung zur Entwicklung und Absicherung des Steuergeräts
4. Simulationsfragestellung zur Absicherung weiterer Funktionen bei der Systemintegration

Diese Kategorien werden nun näher beschrieben.

3.2.1 Simulationsfragestellung zur Funktionsentwicklung

Hier werden die Entwicklung der Algorithmen zur Ansteuerung bzw. Regelung des Stufenautomaten und die Bestimmung der Schaltzeitpunkte verschiedener Fahrstrategien unterstützt. Dabei spielt auch die Abstimmung mit dem Motor eine große Rolle. Für diese Simulationsfragestellungen wird daher ein detailliertes Modell der elektrohydraulischen Ansteuerung des Getriebes benötigt. Dieses Modell ermöglicht eine Untersuchung verschiedener Algorithmen zur

Ansteuerung des Stufenautomaten, wodurch eine Optimierung simulationsgestützt durchgeführt werden kann. Die verwendeten Simulationsverfahren sind MiL am Anfang der Entwicklung und gegen Ende SiL, wenn die Softwareentwicklung nahezu abgeschlossen ist.

3.2.2 Simulationsfragestellung zur funktionalen Absicherung

Nachdem die Optimierung der Ansteuerung abgeschlossen ist, folgt die funktionale Absicherung. Das bedeutet, dass die Funktionssoftware mit einer virtuellen Umgebung in einer SiL-Simulation abgesichert wird. Um diese Tests zu beschleunigen, ist es vorteilhaft, wenn der numerische Aufwand der Modelle nicht zu groß ist. Jedoch werden keine echtzeitfähigen Modelle benötigt, und somit ist die Auswahl der ursprünglich zur Verfügung stehenden numerischen Verfahren nicht eingeschränkt. Die simulationsgestützte Absicherung ermöglicht eine frühzeitige Erkennung von Softwarefehlern. Durch steigende Funktionsumfänge heutiger Stufenautomaten ist diese Absicherung besonders wichtig [Wie+10].

3.2.3 Simulationsfragestellung zur Entwicklung und Absicherung des Steuergeräts

Nachdem die Entwicklung der weiteren Softwareanteile und das Zusammenführen mit der Funktionssoftware auf dem Getriebesteuergerät abgeschlossen sind, muss die Funktionsweise abgesichert werden. Dazu kann zum Beispiel eine HiL-Simulation verwendet werden. Für diese Simulationsfragestellungen wird ein numerisch vereinfachtes Modell benötigt, das jedoch weiterhin die Getriebeansteuerung berücksichtigt. Die Absicherung von Getriebesteuergeräten ist wichtig, da es sich meist um integrierte Steuergeräte handelt und es fast unmöglich ist, ein detailliertes Modell des gesamten Getriebesteuergeräts zu erhalten.

Die HiL-Simulation kann somit auch Fehler, die das Steuergerät betreffen, aufzeigen und ist daher ein wichtiger Teil des Entwicklungsprozesses [Ram+99; KG00; Güh03; Rie+04; Abe+11; Cav+12].

3.2.4 Simulationsfragestellung zur Absicherung weiterer Funktionen bei der Systemintegration

Für weitere Simulationsfragestellungen des Entwicklungsprozesses wird ein Getriebemodell benötigt, um das Zusammenspiel mehrerer mechatronischer Systeme auf Gesamtfahrzeugebene durch eine Simulation absichern zu können Dies kann zum Beispiel die Absicherung des Motorsteuergeräts sein oder die Absicherung weiterer Funktionen bzw. Steuergeräte des Fahrzeugs. Für diese Simulationsfragestellungen reichen meist stark vereinfachte Getriebemodelle. Oft wird bei diesen Modellen auf die Modellierung der elektrohydraulischen Ansteuerung ganz verzichtet, wodurch das dynamische Schaltverhalten stark vereinfacht dargestellt ist.

3.3 Modellbildung von Stufenautomaten und Simulation von Schaltvorgängen

Im Folgenden wird näher auf unterschiedliche Ansätze zur Modellierung von Stufenautomaten eingegangen. Dabei wird zunächst die detaillierte Modellierung der Schaltdynamik vorgestellt, und es werden die numerischen Eigenschaften diskutiert. In diesem Zusammenhang wird gezeigt, dass eine numerische Vereinfachung nötig ist, um die erstellten Modelle der Komponentenentwicklung zur Absicherung des Steuergeräts in einer HiL-Simulation verwenden zu können. Dazu werden verschiedene Ansätze vorgestellt.
Um den Entwicklungsprozess möglichst effizient zu gestalten, soll dabei die numerische Vereinfachung so automatisch wie möglich erfolgen. Deshalb wird

diskutiert, inwiefern die vorhandenen Ansätze dazu ausreichen bzw. weiter entwickelt werden müssen. Die Motivation ist dabei, dass ein schon vorhandenes Modell der Komponentenentwicklung automatisiert numerisch vereinfacht werden kann, so dass eine aufwändige zusätzliche Modellbildung eines echtzeitfähigen Getriebemodells entfallen kann.

3.3.1 Detaillierte Modellierung von Stufenautomaten

Ziel dieser Modellierung ist, eine möglichst exakte Abbildung der dynamischen Effekte der elektrohydraulischen Ansteuerung und der Getriebemechanik zu erreichen. Um ein detailliertes Modell der Ansteuerung zu erhalten, werden die Modellgleichungen durch theoretische Modellbildung abgeleitet. In [WS03; TZC11] wird gezeigt, dass die Berücksichtigung des dynamischen Übertragungsverhaltens der Getriebehydraulik zur Bewertung von Konzepten zur Durchführung von Lastschaltungen unverzichtbar ist. Bei der Modellbildung werden das in Abbildung 3.5 dargestellte Prinzip der elektrohydraulischen Ansteuerung und die Mechanik des Stufenautomaten berücksichtigt.

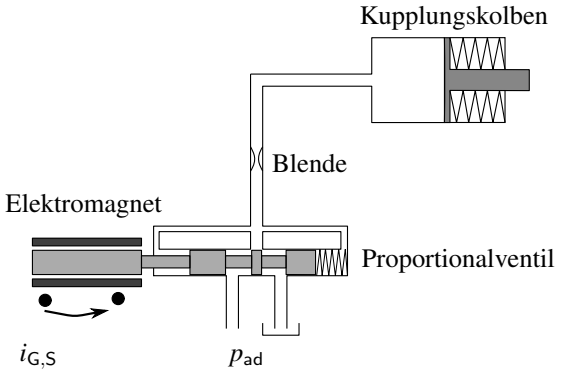


Abbildung 3.5: Elektrohydraulische Ansteuerung von Stufenautomaten

Die elektrohydraulische Ansteuerung bildet dabei die elektronische Schnittstelle, so dass durch Vorgabe elektrischer Ströme Lastschaltungen in der Simulation untersucht werden können.
Zusätzlich kommt ein Modell des Wandlers hinzu. Kann dieser durch eine WÜK überbrückt werden, wird ein zusätzliches Schaltelement modelliert. Bei manchen Ausführungen von Stufenautomaten kann zusätzlich der Arbeitsdruck der Getriebehydraulik an verschiedene Fahrsituationen angepasst werden [Fis+12; Dür+14]. Dabei entsteht ein komplexes System von Modellgleichungen, die das dynamische Verhalten der Getriebehydraulik und Getriebemechanik beschreiben. Der Wandler wird jedoch häufig durch ein Kennfeld modelliert.
In [Koc01; WS03; TZC11; Pfe08b] wird beschrieben, wie diese Gleichungen von Hand aufgestellt werden können. In [JK03; Pfe08a] wird sehr detailliert auf die Modellierung der Getriebehydraulik eingegangen.
Da diese Ableitung der Modellgleichungen bei komplexeren Stufenautomaten sehr aufwändig werden kann, können zur Reduktion des Modellaufwands energieflussbasierte Modellierungswerkzeuge verwendet werden. Dazu können aus einer Modellbibliothek Modelle von Planetengetrieben, Schaltelementen und Bauteilen der elektrohydraulischen Ansteuerung ausgewählt und miteinander verbunden werden. Dies ist beispielhaft in Abbildung 3.6 dargestellt, in welcher die Teilmodelle des Stufenautomaten zu sehen sind.
Die Ableitung der Modellgleichungen findet dabei automatisiert vor dem Start der Simulation statt. Weitere Details zur Modellbildung von Stufenautomaten mit energieflussbasierten Modellierungswerkzeugen sind in [SBB02; SO05; OHK09] zu finden.

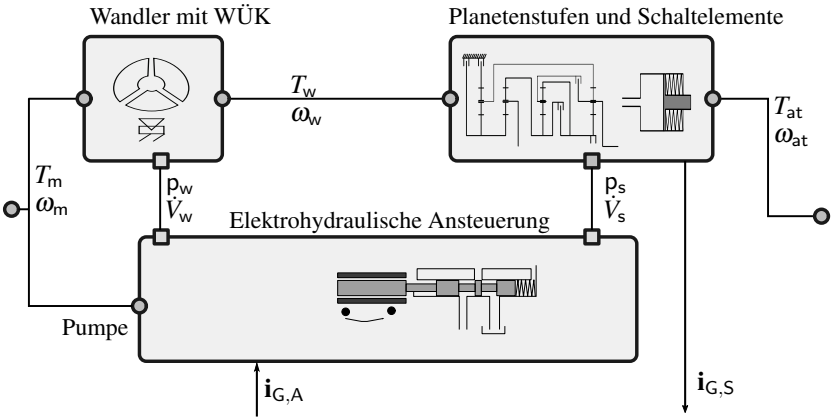


Abbildung 3.6: Teilmodelle eines detaillierten Streckenmodells eines Stufenautomaten

3.3.2 Simulation von Schaltvorgängen mit detaillierten Getriebemodellen

Für diese Simulation wird das erstellte energieflussbasierte Modell eines Stufenautomaten als Teilmodell eines energieflussbasierten Fahrzeugmodells verwendet, wie es in Abbildung 3.7 dargestellt ist [3].

Zur Simulation von Lastschaltungen werden verschiedene Fahrmanöver untersucht. Ziel ist dabei die Untersuchung der Dynamik von Lastschaltungen, die sich bei den Winkelgeschwindigkeiten ω_m des Motors und ω_{at} der Antriebswelle und beim Drehmoment T_{at} der Antriebswelle und T_m des Motors zeigt. Dazu wird je nach Simulationsfragestellung der Detaillierungsgrad des Fahrwerks gewählt, um die Auswirkungen der durchgeführten Lastschaltungen unterschiedlich genau zu untersuchen.

[3] Abbildung 3.7 ist eine erweiterte Darstellung der zweiten Hierarchieebene des Modells aus Abbildung 2.8.

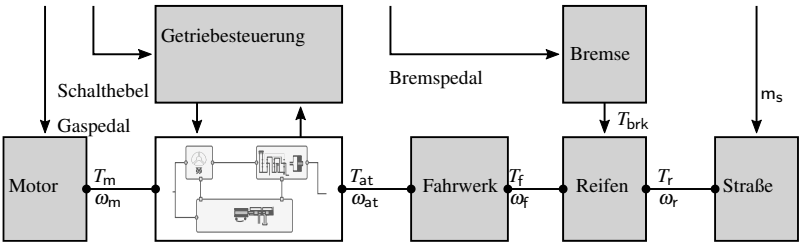


Abbildung 3.7: Modell des Antriebsstrangs zur Simulation von Schaltvorgängen

Um einen Eindruck davon zu gewinnen, wie eine Untersuchung des Schaltverhaltens abläuft, sind typische Größen von Schaltvorgängen eines Stufenautomaten in Abbildung 3.8 als Ausschnitt aus einer Simulation zu sehen. Im Detail sind die Verläufe der Ansteuerströme $\mathbf{i}_{G,A}$ der Schaltelemente, der WÜK und der Regelung des Arbeitsdrucks zu sehen. Die Ansteuerströme können dabei in zwei Gruppen unterteilt werden. Die erste Gruppe $\mathbf{i}_{G,A,1}$ besteht aus den Ansteuerströmen der Schaltelemente, die zweite Gruppe $\mathbf{i}_{G,A,2}$ besteht aus den restlichen Ansteuerströmen. Dabei zeigt sich der charakteristischen treppenförmige Verlauf der Ansteuerströme, um Lastschaltvorgänge durchzuführen. Weiter ist der Verlauf der Drehmomente T_{at} der Antriebswelle, T_m des Motors und T_{brk} der Bremse dargestellt. Außerdem ist auch der Verlauf der Winkelgeschwindigkeiten ω_m des Motors und ω_{at} der Antriebswelle zu sehen. Dabei ist zu sehen, dass die Ansteuerströme dynamischer sind im Vergleich zu den Drehmomente und den Winkelgeschwindigkeiten.
In den gezeigten Signalverläufen ist die Vorfüllphase daran zu erkennen, dass sich die Änderung der Ansteuerströme verzögert auswirkt. Dabei wird deutlich, dass sich eine Änderung der Ansteuerströme zunächst auf das übersetzte Drehmoment an der Antriebswelle auswirkt und anschließend auf die übersetzte Winkelgeschwindigkeit des Motors. Dabei decken sich die genannten Größenordnungen der Zeitkonstanten mit den zuvor genannten.

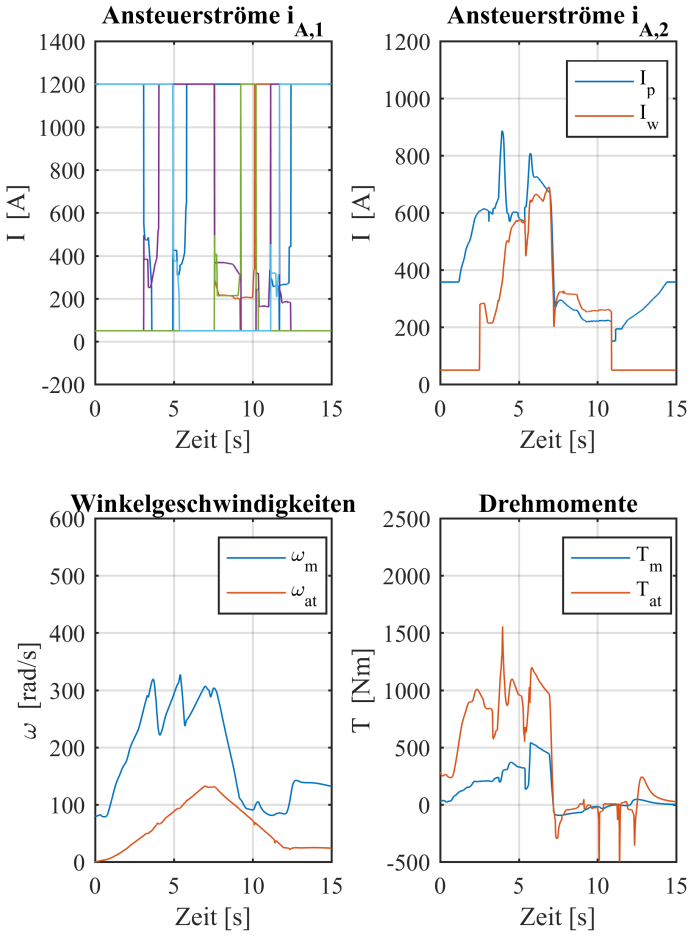


Abbildung 3.8: Beispiel zur Simulation von Schaltvorgängen

In der Literatur finden sich folgende Beispiele, in denen detaillierte Getriebemodelle zum Einsatz kommen. Diese werden zusätzlich den oben genannten Kategorien von Simulationsfragestellungen zugeordnet:

1. Unter Verwendung eines Vierspurmodells des Fahrwerks wird eine Optimierung der Steuerung von Lastschaltvorgängen oder eine Bewertung der Fahrbarkeit durch eine Simulation durchgeführt [Alv08; Mat13] (**Kategorie 1**). Weiter wird in [Alv08] beschrieben, wie diese Modelle zur virtuellen Abstimmung von Motor und Getriebe verwendet werden können. In [Bag+08; Kah13] wird solch ein Modellaufbau zur Optimierung der Ansteuerung von Stufenautomaten verwendet (**Kategorie 1**).
2. In [Koc01; Pfe08b] wird ein detailliertes Modell eines Stufenautomaten dazu verwendet, verschiedene Regelungen zur Durchführung von Schaltvorgängen zu analysieren. Allerdings wird nur die Längsdynamik betrachtet, jedoch besitzt der Antriebsstrang einen Schwingungsfreiheitsgrad, so dass hier auch – bis zu einem gewissen Grad – die Anregung von Schwingungen im Antriebsstrang untersucht werden kann (**Kategorie 1**).
3. Wird die Fahrzeugdynamik auf die Längsdynamik eines Massenpunkts reduziert, kann die Funktionssoftware des Getriebesteuergeräts in einer SiL-Simulation abgesichert werden [RCT09; Chr+11] (**Kategorie 2**).

Hieraus wird ersichtlich, dass detaillierte Getriebemodelle hauptsächlich für Simulationsfragestellungen der ersten beiden Kategorien verwendet werden und sehr selten in Simulationsfragestellungen der Kategorie 3 und 4.
Der Hauptgrund ist in den numerischen Eigenschaften detaillierter Getriebemodelle zu finden, auf die zum Abschluss dieses Abschnitts eingegangen wird.

Bei der Verwendung der vorgestellten detaillierten Getriebemodelle in der Simulation können folgende numerische Schwierigkeiten entstehen:
Durch die Kopplung mechanischer und hydraulischer Teilmodelle entsteht

wegen der stark unterschiedlichen Eigenfrequenzen ein steifes Differentialgleichungssystem. Um dieses zu lösen, müssen angepasste Simulationsverfahren verwendet werden wie zum Beispiel ein implizites Verfahren (siehe Kapitel 2). Auch das in den Schaltelementen verwendete Reibungsmodell bereitet numerische Schwierigkeiten, da dieses zwischen den Zuständen offen, geschlossen und schleifend umschalten kann, wodurch Unstetigkeiten vorhanden sind. Sollen diese Übergänge in einer Simulation korrekt aufgelöst werden, sind kurzzeitig sehr kleine Schrittweiten nötig. Diese numerischen Schwierigkeiten schränken die Auswahlmöglichkeiten von Simulationsverfahren weiter ein.
Die numerischen Schwierigkeiten einer Simulation eines Modells mit hydraulischen und mechanischen Anteilen, sowie die Probleme von Unstetigkeiten in den Modellgleichungen sind detailliert in [Dro03] beschrieben. Diese beiden numerischen Eigenschaften erschweren eine Verwendung eines derartigen Modells in Echtzeitsimulationen, in denen das reale Getriebesteuergerät abgesichert (**Kategorie 3**) oder eine andere Simulationsfragestellung zur Unterstützung der Systemintegration (**Kategorie 4**) untersucht werden soll.
Eine Möglichkeit, die vorhandenen detaillierten Getriebemodelle in diesen Simulationsfragestellungen zu verwenden, ist der Modellaustausch. Dazu werden die Modellgleichungen des detaillierten Getriebemodells in ein Modellierungswerkzeug exportiert, das bei der Systemintegration verwendet wird. Dabei handelt es sich in den meisten Fällen um ein signalflussbasiertes Modellierungswerkzeug, dessen Verfahren zur numerischen Integration ausgewählt werden können. Dabei kann es jedoch passieren, dass die Schaltzustände der Kupplungen nicht richtig berechnet werden und es zu inkonsistentem Schaltverhalten in der Simulation kommt. Das passiert dann, wenn die zur Verfügung stehenden numerischen Verfahren des Modellierungswerkzeugs nicht geeignet sind, um die Schaltdynamik zu berechnen.
Eine weitere Möglichkeit ist durch die Co-Simulation gegeben, bei der das Modellierungswerkzeug, in dem das detaillierte Getriebemodell erstellt wurde, und

das Modellierungswerkeug der Systemintegration zusammen genutzt werden. Zur numerischen Integration können die jeweiligen spezialisierten Verfahren verwendet werden. Weiter müssen Schnittstellen definiert werden, so dass physikalische Größen zwischen den beiden Modellierungswerkzeugen ausgetauscht werden können. Wie das genau geschieht, ist in [Dro03] beschrieben. In der Praxis ist jedoch eine entsprechende Unterstützung der Modellierungswerkzeuge notwendig. Die in der Getriebemodellierung verwendeten Modellierungswerkzeuge ermöglichen zwar prinzipiell solch eine Co-Simulation, allerdings stehen dann die numerischen Verfahren nur eingeschränkt zur Verfügung. So kann es auch hier passieren, dass kein geeignetes numerisches Verfahren für das detaillierte Getriebemodell ausgewählt werden kann. Da die Modellierung von Hydraulik und Mechanik innerhalb eines Werkzeugs stattfindet, ist es zudem sehr schwer, Schnittstellen für den Austausch von Größen zu definieren. Deshalb wird im nächsten Abschnitt auf verschiedene Lösungsansätze eingegangen, wie detaillierte Modelle numerisch vereinfacht werden können, um sie in einer Echtzeitsimulation oder in Simulationsfragestellungen der Systemintegration verwenden zu können.

3.3.3 Ansätze zur numerischen Vereinfachung von Getriebemodellen

Nun wird auf Ansätze zur numerischen Vereinfachung von Getriebemodellen eingegangen. Diese werden benötigt, um im Entwicklungsprozess das erstellte Getriebemodell für weitere Simulationsfragestellungen verwenden zu können. In Kapitel 2.2 wurden dazu die folgenden grundsätzlich verschiedenen Ansätze: Verwendung von Wissen zur Vereinfachung der Modellgleichungen, Mathematische Vereinfachung der Modellgleichungen (durch Reduktion der Anzahl der Zustände) oder Ableitung eines Ersatzmodells durch experimentelle Modellbildung.

Im Folgenden wird diskutiert, wie diese zur numerischen Vereinfachung verwendet werden können.

Verwendung von Wissen zur Vereinfachung der Modellgleichungen

Die numerischen Schwierigkeiten, die bei der Verwendung hydraulischer und mechanischer Teilmodelle entstehen, sind ein bekanntes Problem [Dro03; JK03]. Um den numerischen Aufwand zu senken, gibt es folgende Möglichkeiten:

1. Netzwerkanalyse:
   Das Ziel ist dabei eine Entfernung von Teilmodellen mit hohen Eigenfrequenzen, die das dynamische Verhalten wenig beeinflussen. Dadurch werden die Differentialgleichungen weniger steif, und es können größere Simulationsschrittweiten verwendet werden, ohne dass die Modellgenauigkeit zu stark sinkt oder die Berechnung instabil wird [TZC11; Abe+11; BYA12]. In [Abe+11] wird beschrieben, wie eine HiL-Simulation für das aus technischer Sicht verwandte Doppelkupplungsgetriebe durchgeführt werden kann.
2. Vereinfachung der Hydraulik:
   Es wird eine Ersatzkompressibilität verwendet oder die Trägheit der Ventilkolben erhöht, um hohe Eigenfrequenzen zu vermeiden [Cav+12; Cav+13].
3. Vereinfachung des Reibungsmodells:
   Eine weitere Möglichkeit ist, die numerisch aufwändigen Modellgleichungen der Reibungsmodelle zu modifizieren. Dabei ist das Ziel, dass die durch die Unstetigkeiten in den Reibungsmodellen verursachten numerischen Schwierigkeiten aufgelöst werden, so dass numerische Verfahren mit fester Schrittweite und mit erträglichem Rechenaufwand verwendet werden können [Mat13; BSH14] und eine Echtzeitsimulation ermöglicht wird.

4. Abstraktion der elektrohydraulischen Ansteuerung:
   In [Güh03] werden der mechanische Anteil des Getriebes detailliert modelliert und die dynamischen Effekte der elektrohydraulischen Ansteuerung als quasi-statisch dargestellt. Das Reibungsmodell bleibt unberührt und ist weiterhin zustandsbasiert.

Die ersten drei genannten Möglichkeiten zielen auf eine Anwendung in einer Echtzeitsimulation ab (**Kategorie 3**), die letzte auf eine Anwendung in Simulationsfragestellungen der Systemintegration (**Kategorie 4**), aber auch zur Verwendung in einer Echtzeitsimulation (**Kategorie 3**). Die Vereinfachung der Modellgleichungen erfolgt in den genannten Beispielen teilweise rechnergestützt, allerdings läuft diese nicht automatisch ab.

Mathematische Vereinfachung der Modellgleichungen

Durch eine mathematische Vereinfachung der Modellgleichungen kann die Vereinfachung automatisch erfolgen. Allerdings können nicht alle Verfahren, die in Kapitel 2 vorgestellt wurden, verwendet werden. Die Dynamik von Schaltvorgängen wird durch ein nichtlineares System von Differentialgleichungen beschrieben. Daher kommen nur folgende zwei Verfahren als Lösungsansatz in Betracht:

1. Proper orthogornal decomposition
2. Trajectory piecewise-linear model reduction

Das erste Verfahren liefert gute Ergebnisse hinsichtlich der numerischen Vereinfachung, wenn ein nichtlineares System mit vielen Zuständen vorliegt, wie es bei der Simulation eines Systems mit verteilten Parametern der Fall ist [Ast04]. Dabei werden deutlich über 10000 Zustände verwendet.
Die Modellgleichungen des detaillierten Getriebemodells haben jedoch deutlich weniger Zustände, so dass hier durch Anwendung eines solchen Ansatzes keine

merkliche numerische Vereinfachung möglich ist, da die Anzahl der Zustände kaum weiter reduziert werden kann, um eine Auswirkung zu merken.
Um das zweite Verfahren anwenden zu können, müssen die Modellgleichungen linearisierbar sein. Durch die Verwendung eines zustandsbasierten Reibungsmodells ist dies jedoch nicht möglich, da keine stetigen Ableitungen entstehen, die jedoch eine Voraussetzung für eine Linearisierung sind. Die Unstetigkeiten im Reibungsmodell stellen im Allgemeinen die mathematischen Verfahren vor Herausforderungen. Methoden zur numerischen Vereinfachung nichtlinearer dynamischer Systeme mit Unstetigkeiten sind Teil aktueller Grundlagenforschung [BBF14].
In manchen Fällen sind die Modellgleichungen nicht zugänglich, da zum Beispiel Know-How geschützt wird oder diese als fertige Software vorliegen. In solchen Fällen können diese Verfahren nicht verwendet werden, da dazu die Modellgleichungen explizit vorliegen müssen.
Ein Beispiel, das zu dieser Art von numerischer Vereinfachung passt, ist die Inline-Integration. Hier wird das Ziel verfolgt, ein implizites numerisches Verfahren zu verwenden. Dabei steigt jedoch zunächst der Rechenaufwand für die Lösung des nichtlinearen Gleichungssystems. Um diesen zu senken, werden die Modellgleichungen vor der Simulation durch vorgelagerte Integration vereinfacht [EMO02; Elm+04; SBB02]. Allerdings ist ein Export dieses Modells und somit eine Verwendung in anderen Simulationswerkzeugen kaum möglich, was die Verwendung in Simulationsfragestellungen der Systemintegration einschränkt.

Ableitung eines Ersatzmodells durch experimentelle Modellbildung

Bei der Ableitung eines Ersatzmodells für das detaillierte Getriebemodell durch experimentelle Modellbildung wird eine Modellstruktur verwendet, die nicht die oben genannten numerischen Schwierigkeiten bereitet. Allerdings besitzt das

dabei entstehende Modell im Allgemeinen keine physikalisch interpretierbaren Modellgleichungen. Außerdem bieten die Methoden der experimentellen Modellbildung kein Standardvorgehen zur Modellierung nichtlinearer dynamischer Systeme. Somit muss jedes System individuell bei der Modellierung betrachtet werden.
Ein Beispiel, bei dem ein System mit Reibung durch experimentelle Modellbildung numerisch vereinfacht wird, ist die mechatronische Ansteuerung einer Drosselklappe eines Verbrennungsmotors. In [Ren12; RKS13] wird beschrieben, wie dynamische, stückweise lineare Modelle von mechatronischen Komponenten mit Reibung erstellt und anschließend in Echtzeitsimulation zur Absicherung von Steuergeräten mit einer virtuellen Umgebung verwendet werden können. Dieses Modell besitzt jedoch deutlich weniger Eingangssignale als ein Stufenautomat, weshalb eine Übertragung nicht ohne weiteres möglich ist.
In [CNG13; CNG14] sind verschiedene Ansätze beschrieben, wie das Schaltverhalten von Stufenautomaten durch neuronale Netze beschrieben und dadurch eine numerische Vereinfachung erzielt werden kann.
Ein weiterer Ansatz, der zur Absicherung weiterer Funktionen zum Einsatz kommt, ist die strukturell vereinfachte Modellierung von Stufenautomaten (vgl. [Amm97; Mat13]). Dabei wird auf eine detaillierte Modellierung der Ansteuerung verzichtet und im Wesentlichen nur die Schaltlogik des Stufenautomaten berücksichtigt. Die so erstellten Modelle werden hauptsächlich für Simulationsfragestellungen der Kategorie 4 verwendet.
Die Winkelgeschwindigkeiten und Drehmomente werden vom Stufenautomaten folgendermaßen berechnet

$$\begin{aligned} T_{\mathsf{m}} &= \mathsf{R}(\mathbf{i}_{\mathsf{G,A}})T_{\mathsf{at}} \\ \omega_{\mathsf{at}} &= \mathsf{R}(\mathbf{i}_{\mathsf{G,A}})\omega_{\mathsf{m}}. \end{aligned} \tag{3.2}$$

Dabei steht R für die aktuelle Übersetzung des Getriebes, die durch Ansteuersignale angepasst werden kann.

Diese Berechnungsvorschrift wird dann im Gesamtmodell des Fahrzeugs verwendet, das in Abbildung 2.8 zu sehen ist. Die Übersetzungen R werden dabei durch die Schaltlogik bestimmt und in einem entsprechenden Kennfeld abgelegt. Dazu wird ein vereinfachtes Funktionsmodell verwendet, das über die Steuersignale $\mathbf{i}_{\mathsf{G,A}}$ die optimale Übersetzung vorgeben kann. Je nach angefordertem Gang werden diese dann ausgewählt. Die Schaltübergänge werden durch ein einfaches dynamisches Modell berücksichtigt, so dass diese nicht plötzlich stattfinden. Das kann beispielsweise durch eine Überblendung der Übersetzungen oder ein lineares Verhalten erster Ordnung berücksichtigt werden. Dabei wird eine für Schaltvorgänge typische Schaltdauer von etwa $0,5[\mathrm{s}]$ zu Grunde gelegt [Mat13]. Der Nachteil dieses Modells ist, dass als Funktionsmodell nur die Getriebelogik verwendet werden kann, da es kein einfaches Modell der hydraulischen Ansteuerung gibt. Für Simulationsfragestellungen, bei denen eine vollständige Funktionssoftware oder ein reales Getriebesteuergerät verwendet werden soll, kann solch ein Modell also nicht verwendet werden.
Zudem werden bei dieser vereinfachten Modellierung des Getriebes die Verluste, die durch die Schaltelemente in der Getriebemechanik entstehen, nicht berücksichtigt. Diese wirken sich hauptsächlich auf das Drehmoment T_{at} aus. Zudem wird der Einfluss durch die rotatorischen Trägheiten der Wellen in der Getriebemechanik auf das Drehmoment T_{at} nicht berücksichtigt [Güh02; Güh03; Mat13]. Um diese Effekte valide in das Modell einzubringen, sind weitere Kennfelder nötig, die durch experimentelle Modellbildung gewonnen werden können. Diese Modelle werden für die Simulationsfragestellungen der Kategorie 3 und 4 verwendet.

3.3.4 Bedarf zur automatischen Durchführung der numerischen Vereinfachung von Getriebemodellen

Im letzten Abschnitt wird diskutiert, welchen Bedarf es zur Entwicklung einer Methode gibt, die zur automatischen numerischen Vereinfachung von Getriebemodellen verwendet werden kann. Die vorgestellten Ansätze zur numerischen Vereinfachung zeigen, dass es einen Bedarf im Hinblick auf eine automatische Durchführung der numerischen Vereinfachung gibt. Dies wird im Folgenden näher erläutert, wobei auf die oben genannten Ansätze zur numerischen Vereinfachung eingegangen wird.

Wissensbasierte Vereinfachung der Modellgleichungen

Hier gibt es teilweise Unterstützung durch entsprechende Modellierungswerkzeuge, jedoch muss die Vereinfachung manuell durchgeführt werden. Dabei werden gute Resultate erzielt und die numerischen Eigenschaften so vereinfacht, dass die Modelle in Echtzeitanwendungen verwendet werden können. Eine voll automatisierte Vereinfachung dieser Art erscheint jedoch nach aktuellem Stand der Forschung schwer umsetzbar, da es keinen systematischen Ansatz gibt, eine wissensbasierte Vereinfachung durchzuführen.

Mathematische Vereinfachung der Modellgleichungen

Die Entwicklung dieser Verfahren zielt darauf ab, Modelle automatisch numerisch zu vereinfachen, und ist Teil aktueller Forschung. Für lineare dynamische Systeme sind die Verfahren sehr ausgereift, allerdings ergeben sich Schwierigkeiten in der Anwendung bei nichtlinearen dynamischen Systemen. Dabei ist es stark abhängig vom Anwendungsfall, ob diese verwendet werden können. Viele

der Verfahren für nichtlineare Systeme sind datenbasiert und haben damit Ähnlichkeiten mit der experimentellen Modellbildung. Verfahren, bei denen keine Daten verwendet werden müssen, funktionieren nur in Spezialfällen [BBF14].

Numerische Vereinfachung durch experimentelle Modellbildung

In diesem Fall wird durch experimentelle Modellbildung das Schaltverhalten nachgebildet. Für eine Automatisierung der Modellerstellung müsste jedoch bekannt sein, welche Modellstruktur sich eignet, mit welchem Ansatz die Dynamik modelliert wird und welches Verfahren sich zur Parameteroptimierung eignet. Da für nichtlineare dynamische Systeme dazu kein einheitliches Vorgehen bekannt ist, muss eine spezielle Lösung zur Modellierung der Schaltdynamik gefunden werden.

Als Startpunkt zur Weiterentwicklung kommen die mathematische Vereinfachung der Modellgleichungen und die experimentelle Modellbildung in Frage. Gegen eine Verwendung der Methoden der mathematischen Vereinfachung spricht, dass häufig die Modellgleichungen nicht vorliegen oder weitergegeben werden können. Wenn diese vorliegen, zeigt die obige Diskussion, dass eine wissensbasierte Vereinfachung zielführender ist, da kaum eine weitere Reduktion der Anzahl der Zustände detaillierter Getriebemodelle möglich ist. Zudem kann nicht damit gerechnet werden, dass durch eine mathematische Vereinfachung das Modell der Reibung vereinfacht wird.
Somit ist die Ableitung eines Ersatzmodells durch experimentelle Modellbildung ein guter Ansatz, um eine Methode zur automatischen numerischen Vereinfachung von Getriebemodellen zu entwickeln.
Dazu wurde bereits in [CNG13; CNG14] gezeigt, dass neuronale Netze verwendet werden können, um die Schaltdynamik zu modellieren. Jedoch ist noch ungeklärt, wie die Erstellung der neuronalen Netze automatisch erfolgen kann, damit die numerische Vereinfachung auch automatisch durchführbar ist.

3.4 Fazit

In diesem Kapitel wurde gezeigt, wie detaillierte Getriebemodelle zur Untersuchung von Lastschaltvorgängen verwendet werden können, und wie diese den Entwicklungsprozess unterstützen. Dazu wurde gezeigt, wie sich diese folgenden den vier Kategorien von Simulationsfragestellungen zur Funktionsentwicklung, funktionalen Absicherung, Entwicklung und Absicherung des Steuergeräts und Absicherung weiterer Funktionen bei der Systemintegration zuordnen lassen. Es wurde darauf eingegangen, dass detaillierte Modelle vor allem bei Simulationsfragestellungen zum Einsatz kommen, die die Entwicklung und Absicherung der Funktionssoftware des Getriebesteuergeräts unterstützen. Anschließend wurde gezeigt, dass zur Entwicklung und Absicherung des Steuergeräts bzw. zur Absicherung weiterer Funktionen bei der Systemintegration numerisch vereinfachte Modelle benötigt werden. Ursachen für die numerischen Schwierigkeiten sind dabei die zustandsbasierte Modellierung der Reibung in den Schaltelementen und die gemeinsame Simulation von elektrohydraulischer Ansteuerung und Getriebemechanik. Dadurch wird der Modellierungsaufwand erhöht. Um diesen zu senken, wurde beschrieben, dass es einen Bedarf zur automatischen numerischen Vereinfachung detaillierter Getriebemodelle gibt. Dazu wurden verschiedene Ansätze zur numerischen Vereinfachung diskutiert, wobei erklärt wurde, dass dazu die Verwendung neuronaler Netze geeignet ist.

4 Neuronale Netze zur numerischen Vereinfachung von Getriebemodellen

In diesem Kapitel wird darauf eingegangen, wie eine automatische Erstellung neuronaler Netze dazu verwendet werden kann, detaillierte Getriebemodelle numerisch zu vereinfachen. Im Allgemeinen ist damit zu rechnen, dass der Zusammenhang zwischen den Eingangssignalen und den Ausgangssignalen nichtlinear ist, wobei eine hohe Anzahl von Eingangssignalen vorliegt. Deshalb wird im ersten Teil dieses Kapitels darauf eingegangen, weshalb sich neuronale Netze für eine solche Modellierung eignen, und wie diese zur Modellierung der Schaltdynamik verwendet werden können.
Im zweiten Teil dieses Kapitels wird allgemein auf die Erstellung neuronaler Netze eingegangen, und es wird diskutiert, wie sich die Schritte zur Erstellung neuronaler Netze automatisieren lassen.
Dabei spielt die Strukturoptimierung neuronaler Netze eine wesentliche Rolle. Deshalb werden im dritten und letzten Teil dieses Kapitels verschiedene Ansätze zur statischen und dynamischen Modellierung vorgestellt, und es wird dabei diskutiert, welcher dieser Ansätze sich eignet, die Schaltdynamik darzustellen. Dabei spielt auch der Aufwand zur Erstellung der neuronalen Netze eine Rolle.

4.1 Neuronale Netze zur Modellierung der Schaltdynamik von Stufenautomaten

Neuronale Netze sind in der Lage, einen unbekannten funktionalen Zusammenhang zwischen Eingangssignalen und einem Ausgangssignal eines beliebigen MISO-Systems zu modellieren. Dabei findet eine Komprimierung des Eingangsraums durch die Verwendung von Basisfunktionen statt. Dadurch können theoretisch beliebige nichtlineare dynamische Systeme modelliert werden, da neuronale Netze mathematisch universelle Funktionsapproximatoren sind, mit denen beliebige funktionale Zusammenhänge beliebig genau approximiert werden können, wozu es ein entsprechendes Theorem gibt [Sjö+95; Jud+95; Lju98; Nel01; Nel06b; Sch10; Har14].
Diese Eigenschaft motiviert die Verwendung der neuronalen Netze zur Modellierung der Schaltdynamik von Stufenautomaten, da diese im Allgemeinen eine große Anzahl von Eingangssignalen besitzen und die Schaltdynamik nichtlinear ist. Allerdings beinhaltet dieses Theorem keine Aussage darüber, wie genau die neuronalen Netze dazu aufgebaut sein müssen. Im Folgenden wird deshalb auf den Aufbau neuronaler Netze eingegangen.

4.1.1 Aufbau neuronaler Netze

Neuronale Netze bestehen im Allgemeinen aus einer Eingangsschicht, einer Ausgangsschicht und einer oder mehreren verdeckten Schicht(en). In den verdeckten Schichten werden Basisfunktionen Φ_i des gleichen Typs verwendet. Eine Basisfunktion bildet den Vektor $\mathbf{u}$ der Eingangssignale nichtlinear auf einen skalaren Wert ab, wodurch eine Komprimierung der Dimension des Eingangsraums stattfindet. Wie die Komprimierung durchgeführt wird, bestimmen die Parameter $\mathbf{w}_{1,i}$ der Basisfunktion Φ_i und der Typ der einheitlich verwendeten Basisfunktion der verdeckten Schicht. Diese bilden gleichzeitig den Teil

der Modellparameter, die optimiert werden müssen. Der Modellausgang $\hat{y}$ des neuronalen Netzes ergibt sich durch Überlagerung der M Basisfunktionen Φ, die mit den Parametern $\mathbf{w}_2$ der Ausgangsschicht gewichtet werden (siehe Abbildung 4.1).

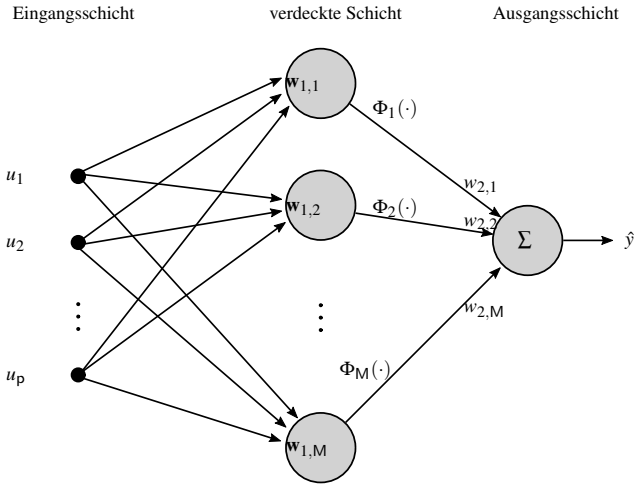


Abbildung 4.1: Neuronales Netz mit einer verdeckten Schicht

Zusätzlich kann noch eine globale Verschiebung verwendet werden, die durch den Parameter w_0 gegeben ist. Somit lässt sich der Modellausgang $\hat{y}$ schreiben als

$$\hat{y}(\mathbf{u}) = \underbrace{w_0 + \sum_{i=1}^{M} w_{2,i}\Phi(\mathbf{u}, \mathbf{w}_{1,i})}_{\mathsf{NN}(\mathbf{u})}. \tag{4.1}$$

Dabei wird mit $\hat{y}$ gekennzeichnet, dass es sich um ein approximiertes Ausgangssignal handelt.

Später wird in diesem Kapitel auf Optimierungsverfahren, die zur Bestimmung der Parameter **w** zum Einsatz kommen, genauer eingegangen.
Soll nun ein neuronales Netz zur Modellierung eines unbekannten funktionalen Zusammenhangs verwendet werden, gewährleistet zwar das oben genannte Theorem, dass dies möglich ist, allerdings wird nicht gesagt, wie viele Basisfunktionen in der verdeckten Schicht verwendet werden müssen, um diesen beschreiben zu können.
Somit muss neben der Optimierung der Parameter der Basisfunktionen auch eine Strukturoptimierung des neuronalen Netzes erfolgen, durch die die optimale Anzahl von Basisfunktionen bestimmt wird. Diese strukturbeschreibenden Parameter werden in der Arbeit als Hyperparameter bezeichnet. Dies wird später in diesem Kapitel weiter thematisiert.

Im Folgenden wird näher auf den oben genannten Konstruktionsmechanismus zur Komprimierung eingegangen. Die Komprimierung der Dimension des Eingangsraums geschieht durch Abbildung des hochdimensionalen Vektors **u** auf eine skalare Größe durch folgende zwei Schritte (siehe Abbildung 4.2):

1. Unter Vorgabe der Parameter $\mathbf{w}_{1,i}$ wird die Aktivierung x_i bestimmt.
2. Anschließend wird die Aktivierung x_i durch eine nichtlineare Funktion transformiert.

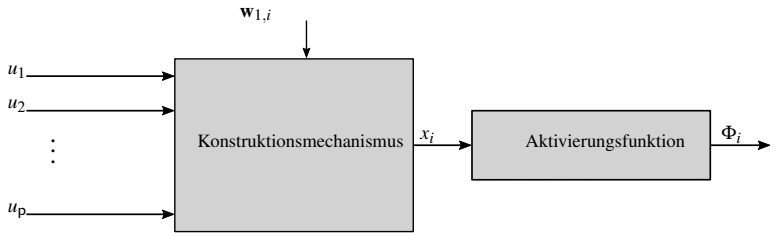


Abbildung 4.2: Prinzip der nichtlinearen Transformation

Zur Berechnung der Aktivierung gibt es folgende zwei Möglichkeiten [Nel06b; Har14]:

1. **Verwendung eines Skalarprodukts**:
   Die Aktivierung x_i wird aus dem Skalarprodukt des Eingangsvektors $\mathbf{u}$ und des Parametervektors $\mathbf{w}_{1,i}$ bestimmt

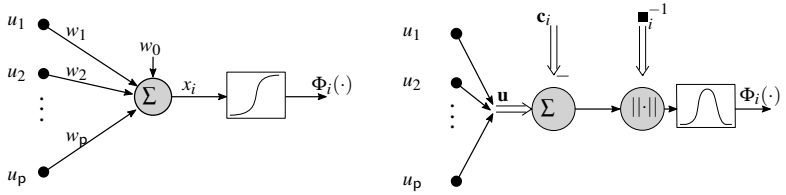
   $$x_i = w_{1,i,0} + \sum_{j=1}^{\mathrm{p}} w_{1,i,j} u_j. \tag{4.2}$$

   Die Aktivierung erfolgt entlang des Parametervektors $\mathbf{w}_{1,i}$. Durch die Bestimmung des Parametervektors wird die Richtung des nichtlinearen Verhaltens bestimmt. Die nach dieser Vorschrift erzeugte Aktivierung heißt Perceptron. Sie ist in Abbildung 4.3a dargestellt und hat eine globale Wirkung.
2. **Verwendung einer Abstandsnorm**:
   Die Aktivierung wird über den Abstand zum Zentrum $\mathbf{c}_i$ bestimmt, das die gleiche Dimension wie der Eingangsvektor $\mathbf{u}$ besitzt. Zur Berechnung wird im Allgemeinen die Mahalanobis-Norm [Nel06b] verwendet, die gegeben ist durch

   $$x_i = ||\mathbf{u} - \mathbf{c}_i||_\Sigma = \sqrt{(\mathbf{u} - \mathbf{c}_i)^T \, \Sigma^{-1} \, (\mathbf{u} - \mathbf{c}_i)}. \tag{4.3}$$

   Dabei wird das Inverse der Kovarianzmatrix Σ verwendet, mit der sich die geometrische Form der resultierenden Aktivierung x_i im Raum drehen und skalieren lässt. Dadurch kann zwar eine sehr flexible Anpassung stattfinden, jedoch muss eine hohe Anzahl an Parametern ermittelt werden. Deshalb ist es sinnvoll, entweder nur die Achsen zu skalieren, was zu einer diagonalen Kovarianzmatrix führt, oder den radialsymmetrischen Fall zu verwenden, so dass alle Werte auf der Diagonalen gleich sind. Letzterer Fall entspricht der Verwendung der euklidischen Norm.

Die nach dieser Vorschrift erzeugte Aktivierung heißt RBF-Neuron und hat durch Verwendung einer Abstandsnorm eine lokale Wirkung (siehe Abbildung 4.3b).



(a) Perceptron (b) RBF-Neuron

Abbildung 4.3: (a) Perceptron (b) RBF-Neuron

Diese beiden unterschiedlichen Konstruktionsmechanismen führen zu folgenden neuronalen Netzen, die eine breite Verwendung finden:

- Multi-Layer-Perceptron-Netz (MLP-Netz), bei dem ein Skalarprodukt verwendet wird
- Lokales Modellnetz (LM-Netz), das eine Erweiterung eines Radial-Basis-Function-Netzes (RBF-Netz) ist

Im Folgenden wird erklärt, welche Ziele durch die Modellierung mit neuronalen Netzen zur Beschreibung der Schaltdynamik erreicht werden sollen.

4.1.2 Ziele der Modellierung mit neuronalen Netzen

In Kapitel 3.2 wurde gezeigt, dass eine automatische numerische Vereinfachung von Getriebemodellen den Entwicklungsprozess dadurch unterstützen würde, dass valide Getriebemodelle aus der Entwicklung für weitere Simulationsfragestellungen verwendet werden könnten.

Das zu erstellende neuronale Netz soll dabei folgende Eigenschaften besitzen:

- Das signalflussorientierte Modell soll Ansteuerströme zur Untersuchung von Schaltvorgängen berücksichtigen. Dabei sollen die dynamischen Effekte, die bei den Schaltvorgängen im detaillierten Modell enthalten sind, approximiert werden. Beispiele für die dynamischen Effekte sind das Überhöhungsmoment oder der Synchronisierungsvorgang zur Anpassung der Winkelgeschwindigkeit.
- Die Modellerstellung soll dabei so automatisiert wie möglich erfolgen. Im Idealfall bedeutet dies eine Knopfdrucklösung. Zusätzlich soll eine Bewertung der erreichten Modellgüte möglich sein, damit auch für andere Fachbereiche deutlich wird, wie valide das Modell ist.
- Um ein numerisch weniger aufwändiges Modell zu erhalten, soll ein zeitdiskretes signalflussorientiertes Modell mit einer festen Abtastzeit T_S entstehen. Die numerische Vereinfachung besteht darin, dass das erstellte Ersatzmodell deutlich weniger Ressourcen bei der Berechnung pro Simulationszeitschritt braucht.
- Die für die experimentelle Modellbildung notwendigen Daten sollen durch Simulation des detaillierten Getriebemodells erzeugt werden. Dabei wird sichergestellt, dass Daten verwendet werden, die das Schaltverhalten valide beschreiben. Außerdem können durch die Simulation beliebig viele Daten erzeugt werden, was sich positiv auf die experimentelle Modellbildung auswirkt. Dabei ist darauf zu achten, dass möglichst alle möglichen Betriebsmodi durchfahren werden, um einen aussagekräftigen Datensatz für die Modellerstellung zu erhalten.

Die Eingangs- und Ausgangssignale des signalflussorientierten Modells eines Stufenautomaten werden so gewählt, dass das Modell möglichst vielseitig in einem Gesamtfahrzeugmodell verwendet werden kann, wie es in Abbildung 2.8 zu sehen ist.

Der Grundgedanke der Modellierung des Schaltverhaltens durch ein signalflussorientiertes neuronales Netz besteht in folgender Erweiterung des in Abschnitt 3.2 beschriebenen funktionalen Modells von Stufenautomaten:
Es sollen nicht nur die statischen Übersetzungen der Drehmomente und Winkelgeschwindigkeiten modelliert, sondern auch die Effekte, die während eines Schaltvorgangs auftreten, durch eine geeignete Modellstruktur beschrieben werden. Diese Modellstruktur besitzt dabei folgende Eingangssignale **u**:

- Elektrische Ansteuerströme $\mathbf{i}_{\mathsf{G,A}}$:
  Durch diese wird die Ansteuerung von Stufenautomaten zur Durchführung von Schaltvorgängen ermöglicht. Diese beeinflussen maßgeblich das Schaltverhalten. Weitere Ansteuersignale können zum Beispiel die Ansteuerung der WÜK oder eine Ansteuerung des Arbeitsdrucks sein (siehe Abschnitt 3.1).
- Drehmoment T_{m} des Motors:
  Das Drehmoment des Motors wird durch den Stufenautomaten für verschiedene Fahrszenarien übersetzt und angepasst, so dass zum Beispiel eine optimale Beschleunigung des Fahrzeugs erfolgen kann.
- Winkelgeschwindigkeit ω_{at} der Antriebswelle:
  Die Winkelgeschwindigkeit der Antriebswelle ist proportional zur Fahrzeuggeschwindigkeit und wird durch den Verlauf verschiedener Größen bzw. Signale beeinflusst, wie zum Beispiel das Drehmoment des Motors oder die elektrischen Ansteuerströme.
- Drehmoment T_{brk} der Bremse:
  Dadurch wird eine Rückwirkung auf eine sich verändernde Last berücksichtigt, die das Schaltverhalten beeinflussen kann.
- Steigung der Straße $\mathsf{m_s}$:
  Die Steigung der Straße bedeutet ebenso eine Lastpunktverschiebung und kann damit auch eine Auswirkung auf das Schaltverhalten haben.

Die Eingangssignale des signalflussorientierten Modells eines Stufenautomaten können in folgendem Vektor zusammengefasst werden

$$\mathbf{u}(k) = \begin{pmatrix} \mathbf{i}_{\mathsf{G,A}}(k) & T_{\mathsf{m}}(k) & \omega_{\mathsf{at}}(k) & T_{\mathsf{brk}}(k) & \mathsf{m_s}(k) \end{pmatrix}. \tag{4.4}$$

Die Ausgangssignale $\mathbf{y}(k)$ des Modells sind folgende:

- Sensorsignale $\mathbf{i}_{\mathsf{G,S}}$:
  Dies sind verschiedene Signale, die Auskunft über physikalische Größen geben, wie zum Beispiel die Winkelgeschwindigkeiten des Motors oder die Winkelgeschwindigkeit des Wandlers.
- Drehmoment T_{at}:
  Das ist das durch das Getriebe übersetzte Drehmoment an der Antriebswelle.
- Winkelgeschwindigkeit ω_{m}:
  Dies ist die durch das Getriebe übersetzte Winkelgeschwindigkeit am Motor.

Die Ausgangssignale können analog in folgendem Vektor zusammengefasst werden

$$\mathbf{y}(k) = \begin{pmatrix} \mathbf{i}_{\mathsf{G,S}}(k), T_{\mathsf{at}}(k), \omega_{\mathsf{m}}(k) \end{pmatrix}. \tag{4.5}$$

Die Verwendung dieser signalflussorientierten Modellstruktur erlaubt eine breite Einbindung in generische, signalflussorientierte Fahrzeugmodelle, wie es in Abbildung 2.8 zu sehen ist. In Abbildung 4.4 ist der Verlauf der numerischen Vereinfachung schematisch abgebildet. Dort ist zu sehen, wie aus dem energieflussbasierten Getriebemodell ein signalflussorientiertes Getriebemodell abgeleitet werden soll. Eine automatische numerische Vereinfachung von Getriebemodellen zu erzielen ist also gleichbedeutend mit der automatischen Erstellung eines neuronales Netzes.

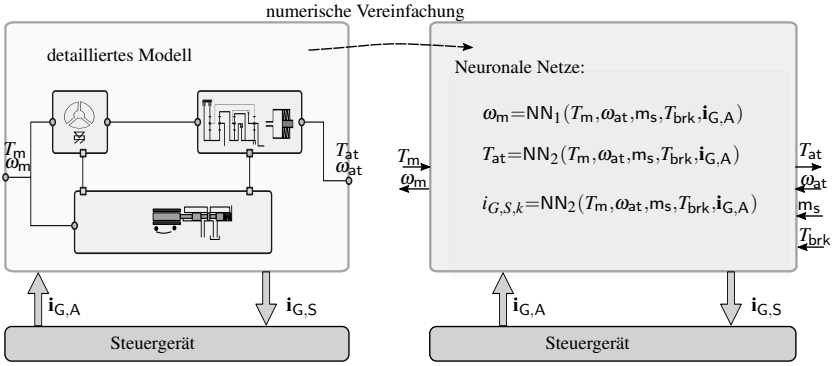


Abbildung 4.4: Numerische Vereinfachung von Getriebemodellen

Dazu werden die Eingangs- und Ausgangssignale aus Gleichung 4.4 und Gleichung 4.5 verwendet, und es wird pro Ausgangssignal ein neuronales Netz erstellt. Um zu zeigen, wie eine automatische Erstellung neuronaler Netze erfolgen kann, wird nun näher auf die oben angeschnittene Struktur- und Parameteroptimierung neuronaler Netze eingegangen.

4.2 Struktur- und Parameteroptimierung neuronaler Netze

In diesem Abschnitt findet eine allgemeine Betrachtung der Struktur- und Parameteroptimierung statt. Dazu wird die Modellerstellung für ein nichtlineares System $f(\mathbf{u})$ mit mehreren Eingängen und einem Ausgang betrachtet. Weiter wird bei der Erstellung neuronaler Netze zwischen Modellparametern $\mathbf{w}$ und K Hyperparametern P unterschieden. Durch die Hyperparameter wird dabei die Architektur des verwendeten neuronalen Netzes bestimmt, wie zum Beispiel die Anzahl der Basisfunktionen oder die Anzahl der verdeckten Schichten. Werden dynamische neuronale Netze verwendet, kommen zusätzliche Hyperparameter

hinzu, durch die die Ordnung der Dynamik beschrieben werden kann. Auf diese wird später näher eingegangen.
Um diese Hyperparameter bei der Modellierung zu berücksichtigen, kann das oben beschriebene neuronale Netz umgeschrieben werden und ist gegeben durch

$$\hat{y} = \mathsf{NN}(\mathbf{u}, \mathbf{w}, \mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_\mathsf{K}). \tag{4.6}$$

Die Bestimmung der Modellparameter $\mathbf{w}$ und der Hyperparameter P_j erfolgt getrennt. Dazu werden die Hyperparameter festgehalten, und es werden die Modellparameter für diese Wahl der Hyperparameter bestimmt. Soll zusätzlich eine Optimierung der Hyperparameter durchgeführt werden, müssen unterschiedliche Kombinationsmöglichkeiten von Hyperparametern untersucht werden, wobei überprüft wird, welche der untersuchten Kombinationsmöglichkeiten sich am besten zur Modellierung eignet. Die automatische Erstellung neuronaler Netze besteht daher aus folgenden zwei Schritte:

1. Parameteroptimierung:
   Mit einem geeigneten Verfahren können die Modellparameter $\mathbf{w}$ des neuronalen Netzes bestimmt werden. Optimierungsverfahren haben häufig zusätzliche Einstellparameter, die den Optimierungsverlauf beeinflussen. Diese Anzahl sollte möglichst gering sein, damit eine automatisierte Modellerstellung möglich ist.
2. Validierung und Strukturoptimierung:
   Ziel ist es, dass das neuronale Netz auch zu Eingangssignalen, die nicht zur Parameteroptimierung verwendet wurden, valide Werte für das Ausgangssignal ausgibt. Für eine automatisierte Modellerstellung müssen zusätzlich die Hyperparameter bestimmt werden.

Im Folgenden werden diese beiden Schritte näher erläutert.

4.2.1 Parameteroptimierung

Die Verfahren zur Parameteroptimierung neuronaler Netze, die in dieser Arbeit betrachtet werden, werden auch als überwachtes Training bezeichnet [Nel01][1]. Ziel dieser Verfahren ist es, die Parameter eines gegebenen neuronalen Netzes so anzupassen, dass die gegebenen Ausgangssignale unter Verwendung der gegebenen Eingangssignale möglichst gut approximiert werden. Die Verfahren zum überwachten Training können folgendermaßen weiter unterschieden werden [Nel01; Sch10]:

- lineare Optimierung
- nichtlineare lokale Optimierung
- nichtlineare globale Optimierung

Für die Parameteroptimierung wird ein Datensatz, im Folgenden der Trainingsdatensatz genannt, verwendet, der das Verhalten des Systems beschreibt. Dazu werden die Eingangssignale $\mathbf{u}$ und das Ausgangssignal y zu N äquidistanten Zeitpunkten abgetastet. Ziel der Verfahren ist die Minimierung der Summe der Fehlerquadrate

$$\min_{\mathbf{w}} \sum_{k=1}^{\mathsf{N}} \left(\underbrace{y(k) - \mathsf{NN}(\mathbf{u}, \mathbf{w}, \mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_\mathsf{K})}_{\varepsilon} \right)^2 . \tag{4.7}$$

Die dazugehörige Optimierung der Parameter $\mathbf{w}$ läuft dabei ab, wie es in Abbildung 4.5 zu sehen ist.

[1] In [Nel01] ist eine sehr anschauliche Beschreibung der Funktionsweise der Algorithmen zur Parameteroptimierung zu finden.

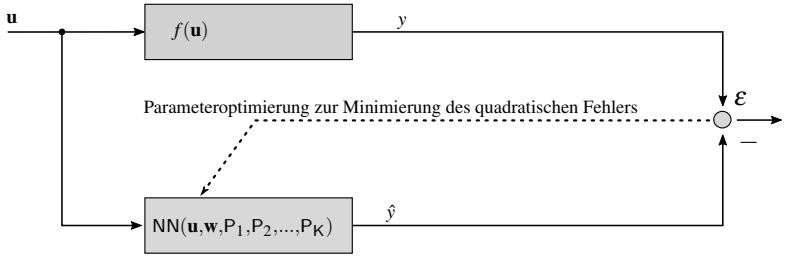


Abbildung 4.5: Allgemeines Vorgehen bei der Parameteroptimierung neuronaler Netze

Lineare Optimierung

Verfahren zur linearen Optimierung können verwendet werden, wenn das zu Grunde liegende mathematische Modell linear in den Parametern **w** ist. Dies ist durch folgenden Zusammenhang erfüllt

$$\hat{y} = \mathbf{w} \cdot \Phi(\mathbf{u}). \tag{4.8}$$

Zur Berechnung der Modellparameter, durch die die Summe der Fehlerquadrate minimal wird, wird folgende Messmatrix aufgestellt

$$\mathbf{X} = \begin{pmatrix} \Phi_1(\mathbf{u}(1)) & \Phi_2(\mathbf{u}(1)) & \dots & \Phi_M(\mathbf{u}(1)) \\ \Phi_1(\mathbf{u}(2)) & \Phi_2(\mathbf{u}(2)) & \dots & \Phi_M(\mathbf{u}(2)) \\ \vdots & & & \vdots \\ \Phi_1(\mathbf{u}(N)) & \Phi_2(\mathbf{u}(N)) & \dots & \Phi_M(\mathbf{u}(N)) \end{pmatrix}. \tag{4.9}$$

Die Parameter **w** des Modells, die zum globalen Minimum der Fehlerquadrate aus Gleichung (4.7) führen, sind gegeben durch

$$\mathbf{w} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T y. \tag{4.10}$$

Dieses Verfahren wird auch als Least-Squares bezeichnet [Nel01]. Es gibt noch weitere Verfahren, mit denen die Modellparameter bestimmt werden können, wie zum Beispiel das rekursive Least-Squares-Verfahren. Im Gegensatz zum vorgestellten Least-Squares-Verfahren werden dort die Messdaten nacheinander verwendet und nicht wie zuvor alle auf einmal. Dies findet vor allem dann Anwendung, wenn während der Messung direkt ein Modell erstellt werden soll [Sch10]. Einen Überblick über weitere Verfahren zur linearen Optimierung findet sich in [Nel01].

Nichtlineare lokale Optimierung

Die Optimierung der Parameter erfolgt in diesem Fall iterativ [Nel01]. Dazu müssen die Modellparameter $\mathbf{w}$ initial vorgegeben werden. Ein weit verbreitetes Verfahren zur Minimierung der Kostenfunktion ist das Gradienten-Verfahren, bei dem der Gradient der Kostenfunktion ε bezüglich der Modellparameter $\mathbf{w}$ verwendet wird. Dieses ist gegeben durch

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \gamma \frac{\partial \varepsilon_i}{\partial \mathbf{w}_i}. \tag{4.11}$$

Dabei wird das Konvergenzverhalten im Wesentlichen von der Schrittweite γ beeinflusst. Bei der Verwendung dieses Verfahrens zeigen sich folgende Schwierigkeiten [Nel01; Sch10]:

- Langsame Konvergenz bei zu kleiner Schrittweite
- Oszillatorisches Verhalten bei zu großer Schrittweite
- Es wird nicht das globale, sondern nur ein lokales Minimum gefunden

Bei zahlreichen Anwendungen mit einer quadratischen Fehlerfunktion hat sich der Levenberg-Marquardt-Algorithmus [Nel01; Sch10] durchgesetzt. In der Praxis zeigt er robuste Konvergenz und ist gegeben durch

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \gamma_k \left(\mathbf{J}_k^T \mathbf{J}_k + \lambda_k \mathsf{I}\right)^{-1} \mathbf{J}_k^T \mathbf{f}_k. \tag{4.12}$$

Dabei wird die Jacobi-Matrix $\mathbf{J}$ der Kostenfunktion verwendet, die gegeben ist durch

$$\mathbf{J} = \begin{pmatrix} \frac{\partial \mathbf{f}(1)}{\partial w_1} & \frac{\partial \mathbf{f}(1)}{\partial w_2} & \cdots & \frac{\partial \mathbf{f}(1)}{\partial w_\mathsf{M}} \\ \frac{\partial \mathbf{f}(2)}{\partial w_2} & \frac{\partial \mathbf{f}(2)}{\partial w_2} & \cdots & \frac{\partial \mathbf{f}(2)}{\partial w_\mathsf{M}} \\ \vdots & \vdots & & \vdots \\ \frac{\partial \mathbf{f}(\mathsf{N})}{\partial w_1} & \frac{\partial \mathbf{f}(\mathsf{N})}{\partial w_2} & \cdots & \frac{\partial \mathbf{f}(\mathsf{N})}{\partial w_\mathsf{M}} \end{pmatrix}. \tag{4.13}$$

Bei diesem Verfahren wird zwar ein höherer Rechenaufwand benötigt, um die Jacobi-Matrix zu bestimmen. Das Verfahren besteht dabei aus einer Kombination des Gauß-Newton-Verfahrens, das in Verbindung mit einer Regularisierungstechnik verwendet wird, das absteigende Funktionswerte erzwingt [Nel01]. Dadurch entfällt die Vorgabe der Lernschrittweite bei der Parameteroptimierung, und das Konvergenzverhalten wird robuster, wodurch eine automatisierte Modellerstellung erleichtert wird. Dies geschieht nach jedem Iterationsschritt, bei dem betrachtet wird, ob die Summe der Fehlerquadrate weiter minimiert werden konnte. Eine genaue Beschreibung der einzelnen Schritte findet sich in [Sch10]. Bei diesem Verfahren besteht jedoch weiterhin die Gefahr, dass ein lokales Minimum gefunden wird.

Nichtlineare globale Optimierung

Bei der globalen Optimierung werden stochastische Elemente zur Minimierung der Summe der Fehlerquadrate eingebaut, mit dem Ziel, das globale Minimum zu finden. Dafür gibt es verschiedene Strategien, wie zum Beispiel das Simulated Annealing, das Particle Swarm und evolutionäre bzw. genetische Verfahren [Nel01; Sch10]. Charakteristisch ist dabei, dass alle Parameter gleichzeitig optimiert werden. Bei einer Vielzahl von Modellparametern ist jedoch mit einem hohen Zeitaufwand bei der Optimierung zu rechnen.
Der Verlauf der Optimierung hängt dabei außerdem von den gewählten zusätzlichen Einstellparametern der Verfahren ab, und die Konvergenz ist im Allgemeinen langsam, vor allem, wenn es sich um viele zu optimierende Modellparameter handelt, was bei der Modellbildung von Stufenautomaten zu erwarten ist. So wird eine automatisierte Modellerstellung erschwert, weshalb in dieser Arbeit diese Verfahren nicht weiter betrachtet werden. Jedoch können diese Verfahren zur Bestimmung der Hyperparameter verwendet werden. Im nächsten Kapitel wird auf diese Möglichkeit näher eingegangen.

4.2.2 Validierung und Strukturoptimierung

Nachdem die optimalen Modellparameter eines neuronalen Netzes bestimmt wurden, gibt es nach [Ren12] zur Validierung folgende Möglichkeiten:

- Bestimmung der Abweichungen durch die Wurzel der mittleren quadratischen Abweichungen (RMSE), in manchen Fällen auch ohne Verwendung der Wurzel (MSE).
- Bestimmung der kleinsten und größten Abweichung
- Vergleich des vorgegebenen Ausgangssignals y und des Modellausgangssignals $\hat{y}$ im zeitlichen Verlauf
- Histogramm der Modellabweichungen

Dadurch kann ein Eindruck davon gewonnen werden, wie gut die gegebenen Daten durch das neuronale Netz modelliert werden. Allerdings gilt das nur für eine feste Wahl von Hyperparametern, die die Struktur des neuronalen Netzes beschreiben. Durch eine andere Wahl von Hyperparametern kann die Anzahl der zu optimierenden Modellparameter erhöht oder reduziert werden.
Das Vorgehen zur Bestimmung optimaler Hyperparametern wird in dieser Arbeit als Strukturoptimierung bezeichnet. Dabei wirkt sich im Allgemeinen eine Änderung der Hyperparameter auch auf die Anzahl der Modellparameter aus. Durch eine steigende Anzahl von Modellparametern steigt die Flexibilität des neuronalen Netzes. Dadurch kann das durch die Daten gegebene Verhalten besser modelliert werden. Dabei ist es jedoch wichtig, eine Überanpassung zu vermeiden [Nel01; Sch10]. Unter Überanpassung wird dabei verstanden, dass das neuronale Netz die zur Parameteroptimierung vorgegebenen Daten sehr gut modellieren kann. Sobald aber Daten als Eingangssignale verwendet werden, die nicht zur Parameteroptimierung verwendet wurden, können sehr große Modellfehler entstehen. In der Literatur ist dies als Bias-Variance-Dilemma bekannt [Nel01], das sich folgendermaßen ergibt. Einerseits soll erreicht werden, dass die Trainingsdaten sehr gut beschrieben werden. Andererseits soll jedoch auch eine Überanpassung vermieden werden, da auch vom Trainingsdatensatz abweichende Daten verwendet werden sollen, ohne dass Divergenzen auftreten. Die letztere Eigenschaft wird auch als Generalisierungfähigkeit eines neuronalen Netzes bezeichnet [Nel01].
Um eine Überanpassung zu vermeiden, gibt es verschiedene Möglichkeiten, die ausführlich in [Nel01; Har14] beschrieben sind, wie zum Beispiel die Kreuzvalidierung, die bei eingeschränktem Datenzugang verwendet wird.
In dieser Arbeit wird, da durch die Verwendung von Simulationsdaten keine Einschränkungen bei der Menge an Daten zu erwarten sind, folgendes Vorgehen zur Strukturoptimierung verwendet:

Die Simulationsdaten werden in drei Anteile aufgeteilt. Der erste Teil wird zur Parameteroptimierung, der zweite Teil für die Validierung und der dritte Teil für den Test verwendet. Damit kann untersucht werden, ob eine Überanpassung bzw. Unteranpassung vorliegt. Mit steigender Anzahl der Modellparameter wird folgender Verlauf der verschiedenen Modellfehler erwartet:
Der Fehler der Trainingsdaten sinkt mit steigender Modellkomplexität, während der Validierungsfehler nur zunächst sinkt und ab einer gewissen Modellkomplexität wieder ansteigt. Wenn der Validierungsfehler den kleinsten Wert annimmt, ist die optimale Modellkomplexität gefunden. Eine zu niedrige Modellkomplexität bedeutet Unteranpassung und eine zu hohe Modellkomplexität Überanpassung. Dieses Verhalten ist schematisch in Abbildung 4.6 zu sehen. Zur Berechnung des Modellfehlers wird in dieser Arbeit die Wurzel des Mittelwerts der quadratischen Abweichungen verwendet (root mean square error), der gegeben ist durch

$$\mathrm{RMSE} = \sqrt{\frac{1}{\mathsf{N}} \sum_{i=1}^{\mathsf{N}} (y(i) - \hat{y}(i))^2}. \tag{4.14}$$

Zur Strukturoptimierung werden die jeweiligen Datensätze verwendet, um den Trainingsfehler $\mathrm{RMSE}_{\mathsf{T}}$ und den Validierungsfehler $\mathrm{RMSE}_{\mathsf{V}}$ zu berechnen und anschließend zu entscheiden, welche Modellkomplexität optimal ist. Um eine abschließende Bewertung des erstellten Modells durchzuführen, wird der Testdatensatz verwendet, der nicht bei der Struktur- und Parameteroptimierung berücksichtigt wird. Dazu wird der Testdatenfehler $\mathrm{RMSE}_{\mathsf{TS}}$ betrachtet und überprüft, ob sich dieser in der gleichen Größenordnung wie der Validierungsfehler befindet.

4.2.3 Automatische Erstellung neuronaler Netze

Für eine automatische Erstellung neuronaler Netze werden zunächst ein Trainings-, ein Validierungs- und ein Testdatensatz benötigt. Die Trainingsdaten

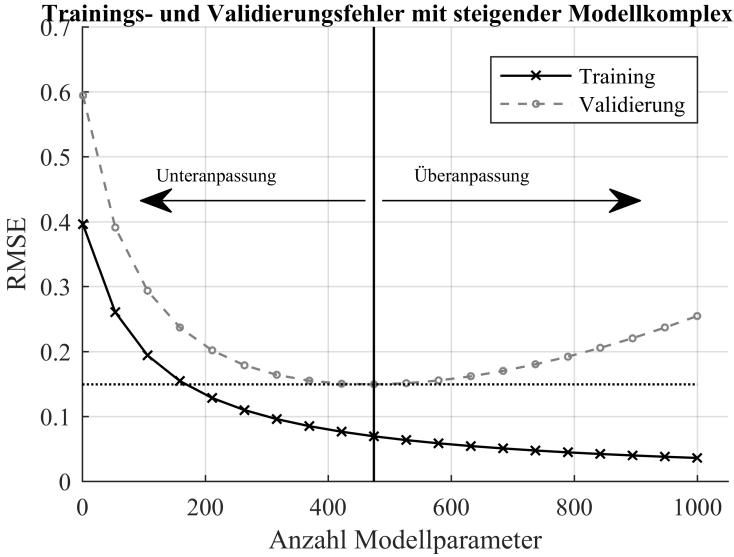


Abbildung 4.6: Verlauf von Trainingsfehler und Validierungsfehler mit steigender Anzahl von Modellparametern und Einteilung in die Bereiche Unter- und Überanpassung

werden dazu verwendet, die Modellparameter zu bestimmen. Zur Bestimmung der Modellparameter wird eines der oben vorgestellten Verfahren zur Optimierung verwendet. Die Auswahl hängt davon ab, ob ein nichtlineares oder lineares Optimierungsproblem vorliegt. Durch Verwendung der Validierungsdaten wird die Strukturoptimierung durchgeführt, was gleichbedeutend ist mit der Optimierung der Hyperparameter. Anschließend wird durch Verwendung des Testdatensatzes eine abschließende Validierung des erstellten Netzes durchgeführt, bei der der Testdatenfehler in der Größenordnung des Trainingsdaten- und Validierungsdatenfehlers liegen sollte.

Im Folgenden wird näher auf die Modellbildung mit neuronalen Netzen eingegangen, wobei zwischen statischer und dynamischer Modellbildung unterschie-

den wird. Dabei wird zusätzlich gezeigt, welche zu optimierenden Hyperparameter durch eine Strukturoptimierung bestimmt werden müssen.

4.3 Statische Modellbildung mit neuronalen Netzen

Zur statischen Modellbildung werden in diesem Abschnitt die Eigenschaften der gebräuchlichsten neuronalen Netze zusammengefasst und diskutiert, wie diese zur Modellierung der Schaltdynamik verwendet werden können:
Multi-Layer-Perceptron-Netze (MLP-Netze) und lokale Modellnetze (LM-Netze).

4.3.1 Multi-Layer-Perceptron-Netze

Die nichtlineare Transformation der M Basisfunktionen (Neuronen) in der verdeckten Schicht besteht aus Aktivierungsfunktionen g_i, in denen das Skalarprodukt aus Eingangssignalen und dem jeweiligen Parametervektor $w_{1,j}$ des Neurons k weiterverwendet wird. Dabei findet eine starke Kompression des Eingangsraums statt, weshalb sich dieser Netztyp besonders für Modellierungsaufgaben mit einer hohen Dimension des Eingangsraums eignet. Der Parametervektor hat dabei die gleiche Dimension wie die Eingangssignale $\mathbf{u} = \left(u_1, u_2, \ldots, u_{\mathsf{p}}\right)$. In der Ausgangsschicht wird die gewichtete Summe aller Aktivierungen gebildet. Diese besteht aus dem Produkt der Aktivierung des Neurons k und den Parametern $w_{2,k}$. Für die Aktivierung wird in diesem Beispiel der Tangens Hyperbolicus verwendet. Es gibt eine Reihe von weiteren Möglichkeiten, jedoch werden sehr häufig mit dieser Aktivierungsfunktion gute Ergebnisse erzielt [Nel01; Sch10]. Das MLP-Netz[2] ist gegeben durch

[2] MLP-Netz ... Multi-Layer-Perceptron-Netz

$$\hat{y} = \sum_{k=1}^{M} \mathbf{w}_{2,k} \cdot \tanh\left(\sum_{j=1}^{p} \mathbf{w}_{1,j} u_j\right). \tag{4.15}$$

Der Modellausgang entsteht durch gewichtete Überlagerung von Basisfunktionen (siehe auch Abbildung 4.1). Die Modellparameter $\mathbf{w}$ von MLP-Netzen werden üblicherweise alle gleichzeitig nichtlinear optimiert. Dazu werden lokale, nichtlineare Optimierungsverfahren wie zum Beispiel das Gradienten-Verfahren oder der Levenberg-Marquardt-Algorithmus verwendet [Nel01]. Für viele Klassifikationsaufgaben, wie zum Beispiel in der Bildverarbeitung, sind MLP-Netze der Standard, da diese sich besonders gut für hoch-dimensionale Eingangsräume eignen. Allerdings können diese auch zur Modellierung statischer Systeme mit mehreren Eingangssignalen verwendet werden. Ein möglicher Nachteil von MLP-Netzen ist dabei, dass durch die global wirkenden Basisfunktionen eine Interpretation des Modells erschwert wird. Vor allem bei der Verwendung mehrerer verdeckter Schichten ist es schwer nachzuvollziehen, wie genau die Berechnung bestimmter Modellausgänge zustande kommt [Nel01].
Im Folgenden wird auf Einzelheiten beim Ablauf der Parameteroptimierung eingegangen. Am Anfang der Parameteroptimierung werden die Parameter des MLP-Netzes für die Optimierung zufällig initialisiert. Das hat zur Folge, dass bei jeder Optimierung unterschiedliche Modellparameter ermittelt werden, da die Optimierungsverfahren unterschiedliche lokale Minima finden. Deshalb wird in der Praxis die Optimierung mehrmals wiederholt, wobei die erstellten Modelle miteinander verglichen werden, und anschließend das neuronale Netz mit dem kleinsten Validierungsfehler ausgewählt wird [Nel01; Hay09]. In manchen Anwendungen werden MLP-Netze mit mehr als einer verdeckten Schicht verwendet (siehe Abbildung 4.7), wodurch nichtlineare, statische Zusammenhänge besser modelliert werden können. Dadurch steigt jedoch auch der Aufwand der Parameteroptimierung durch eine höhere Anzahl von Modellparametern und verdeckten Schichten. Deshalb werden in den meisten Anwendungen nicht mehr als zwei Schichten verwendet [Nel01; Hay09].

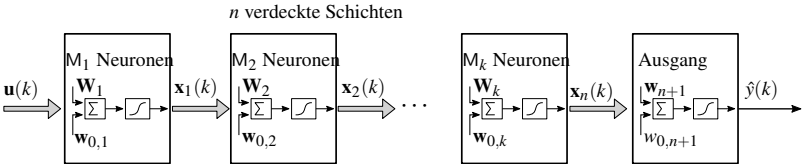


Abbildung 4.7: Graphische Darstellung eines MLP-Netzes mit k verdeckten Schichten

Für die Strukturoptimierung muss eine geeignete Anzahl von Neuronen M pro verdeckter Schicht und eine geeignete Anzahl n von verdeckten Schichten gefunden werden. Somit stellen diese beiden Parameter Hyperparameter dar. Die optimale Struktur eines MLP-Netzes kann also durch Vorgabe unterschiedlicher Architekturen erfolgen, wobei anschließend die mit dem kleinsten Validierungsfehler ausgewählt wird. Um MLP-Netze zu erstellen, gibt es zahlreiche vorhandene Implementierungen, so dass diese auch gut in der Praxis verwendet werden können. Ein Beispiel für eine Implementierung ist Matlab in Kombination mit der Neural Network Toolbox [Mat].

4.3.2 Lokale Modellnetze

Lokale Modellnetze stellen eine Erweiterung von Radialen-Basis-Funktionen-Netzen dar, die als Konstruktionsmechanismus eine Abstandsnorm verwenden. Somit werden immer nur wenige Basisfunktionen durch einen bestimmten Eingangsvektor aktiviert [TS85; Mur94; Nel01; Sch10]. Als nichtlineare Funktionen im oben beschriebenen Konstruktionsmechanismus ist die Verwendung gaußscher Glockenfunktionen weit verbreitet, die gegeben sind durch

$$\mu_i(\mathbf{u},\Sigma) = \exp\left(\frac{1}{2}\left(\mathbf{u}-\mathbf{c}_i\right)^T \Sigma^{-1}\left(\mathbf{u}-\mathbf{c}_i\right)\right). \tag{4.16}$$

Dabei ist Σ die Kovarianzmatrix, durch die die Form der gaußschen Glockenkurve beeinflusst werden kann. Wie genau das geschieht, wurde weiter oben bereits beschrieben (siehe Gleichung 4.3). Bei lokalen Modellnetzen werden diese jedoch normiert als Basisfunktionen verwendet. Die normierten Gaußfunktionen sind gegeben durch

$$\Phi_i(\mathbf{u},\Sigma) = \frac{\mu_i(\mathbf{u},\Sigma)}{\sum_{k=1}^{\mathsf{M}} \mu_k(\mathbf{u},\Sigma)}. \tag{4.17}$$

Sie nehmen Werte zwischen null und eins an. Deshalb werden diese in dieser Arbeit auch Aktivierungsfunktionen genannt [3]. Nun wird darauf eingegangen, wie diese im lokalen Modellnetz verwendet werden, um eine statische Nichtlinearität zu modellieren:

Die Idee lokaler Modellnetze ist es, den Eingangsraum in M lokale gültige Modelle einzuteilen, um das globale nichtlineare Verhalten durch lokale, einfache Modelle zu approximieren. In den meisten Fällen werden lokal-lineare Modelle verwendet, weshalb dieser Netztyp in dieser Arbeit als lokales Modellnetz bezeichnet wird. Zusätzlich wird jedes lokal-lineare Modell jeweils mit einer Basisfunktion Φ_i gewichtet. Die Basisfunktionen verwenden zur Berechnung der Aktivierung einen radialen Konstruktionsmechanismus. Dabei sind die Basisfunktionen Φ_i so normiert, dass deren Summe eins ergibt. Dazu werden gaußsche Glockenkurven aus Gleichung (4.16) verwendet und normiert. Auf diese Weise können je nach Eingangsvektor die lokalen Modelle unterschiedlich stark gewichtet werden. Für den allgemeinen Fall ist ein lokales Modellnetz gegeben durch

$$\hat{y} = \sum_{i=1}^{\mathsf{M}} \left(w_{i,0} + \sum_{j=1}^{\mathsf{p}} w_{i,j} \cdot u_j \right) \Phi_i(\mathbf{u}). \tag{4.18}$$

[3] Eine weitere Bezeichnung für die Basisfunktionen lokaler Modellnetze sind Gültigkeitsfunktionen. Dabei werden die nicht normierten Gaußfunktionen auch als Zugehörigkeitsfunktionen bezeichnet [Nel01].

Dabei sind die $w_{i,j}$ die Parameter des lokalen Modellnetzes, und die Anzahl der Parameter beträgt $(\mathsf{p}+1)\cdot\mathsf{M}$.
Eine weitere Möglichkeit bei der Modellierung mit lokalen Modellnetzen ist die Verwendung von unterschiedlichen Regressoren, also eines **z**-Regressors und eines **x**-Regressors. Diese werden verwendet, um aus den Eingangssignalen **u** jeweils eine bestimmte Teilmenge eingehen zu lassen. Dies ist vorteilhaft, wenn bekannt ist, dass ein Eingangssignal einen linearen Einfluss hat. Dann kann dies berücksichtigt werden, indem dieses Eingangssignal nicht im **z**-Regressor verwendet und somit bei der Partitionierung nicht berücksichtigt wird. Eine weitere Anwendung ist, dass physikalisch motivierte lokale Modelle und entsprechende Signale im **x**-Regressor verwendet werden. So können zum Beispiel die lokalen Modelle einen elektrischen Widerstand beschreiben, der seinen Wert in Abhängigkeit von bestimmten Eingangssignalen ändert.
Die Unterscheidung in einen **x**-Regressor und einen **z**-Regressor ist gegeben durch

$$\hat{y} = \sum_{i=1}^{\mathsf{M}} \left(\underbrace{w_{i,0} + \sum_{j=1}^{\mathsf{p}\times} w_{i,j} \cdot x_j}_{\mathrm{LM}(\mathbf{x})} \right) \Phi_i(\mathbf{z}). \tag{4.19}$$

In Abbildung 4.8 ist eine graphische Darstellung der Gleichung (4.19) zu sehen.
Dort werden der **x**-Regressor und der **z**-Regressor als Eingangssignale verwendet und das lokale Modellnetz durch folgende drei Anteile dargestellt:

1. Die lokalen Modelle $\mathrm{LM}(\mathbf{x})$
2. Die Gültigkeitsfunktionen bzw. Aktivierungsfunktionen $\Phi(\mathbf{z})$
3. Die gewichtete Summe Σ von Φ_i und LM_i, die den Modellausgang $\hat{y}$ ergibt

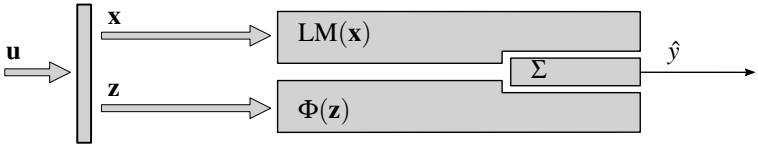


Abbildung 4.8: Vereinfachte graphische Darstellung eines lokalen Modellnetzes mit **x**-Regressor und **z**-Regressor

Die Parameteroptimierung lokaler Modellnetze kann wie bei MLP-Netzen durch Verwendung der Verfahren der lokalen, nichtlinearen Optimierung erfolgen. Dabei werden alle Parameter, d. h. die Zentren, die Standardabweichungen und die Parameter der lokalen Modelle des lokalen Modellnetzes gleichzeitig optimiert. Ein großer Vorteil lokaler Modellnetze ist jedoch, dass sogenannte heuristische Verfahren zur Parameteroptimierung verwendet werden können [Nel01]. Der Grundgedanke dieser Verfahren ist, die Parameteroptimierung in folgende zwei Teile aufzuteilen.

Im ersten Teil werden die Zentren c_i und Standardabweichungen σ_i der Aktivierungsfunktionen durch ein Partitionierungsverfahren[4] festgelegt.

In zweiten Teil werden dann die Parameter $\mathbf{w}$ der lokalen Modelle bestimmt. Dabei entsteht ein lineares Optimierungsproblem und die Parameter der lokalen Modelle können mit geringerem Aufwand ermittelt werden als bei gleichzeitiger nichtlinearer, lokaler Optimierung aller Modellparameter. Im Folgenden werden verschiedene Ansätze dazu vorgestellt.

[4] Durch das Partitionierungsverfahren findet die Teilung des Eingangsraums in lokale Modelle statt.

Verteilung von Zentren auf einem Gitter

Die gleichförmige Verteilung von Zentren auf einem Gitter eignet sich jedoch hauptsächlich für eine kleine Anzahl von Eingangssignalen ($p < 4$), da mit wachsender Anzahl von Eingangssignalen die Anzahl der Modellparameter exponentiell ansteigt (Fluch der Dimensionalität [Nel01]).

Clustering-Verfahren

Bei diesen Verfahren wird die Verteilung der vorliegenden Daten verwendet, um den Eingangsraum zu partitionieren. Ein Beispiel für solch ein Clustering-Verfahren ist der k-means-Algorithmus [Nel01]. Dieses Verfahren verwendet ein Nächster-Nachbar-Prinzip, um die Zentren zu setzen. Weitere Beispiele finden sich in [Pao89; NMG93; Nel01]. Der Vorteil dieser Verfahren ist, dass nur in Bereichen des Eingangsraums, in denen Daten vorhanden sind, lokale Modelle entstehen. Der Nachteil beim Clustering im Eingangsraum ist, dass die Komplexität des Prozesses beim Setzen der Zentren nicht berücksichtigt wird [Nel01; Har14]. Die Modellkomplexität kann berücksichtigt werden, indem zusätzlich die gemessenen Ausgangsdaten beim Clustering verwendet werden. In diesem Fall wird vom Clustering im Produktraum gesprochen. Beispiele hierfür sind der Gustafson-Kessel-Algorithmus und der Gath-Geva-Algorithmus, mit denen eine besonders flexible Anpassung der Gültigkeitsfunktionen möglich ist, da eine vollbesetzte Kovarianzmatrix ermittelt wird. Dies führt dazu, dass die daraus resultierenden Gültigkeitsfunktionen im Raum gedreht werden können. Somit ist eine besonders flexible Anpassung möglich. Weitere Details sind in [BV96; ABS02] beschrieben. Nachteil dieser Verfahren ist, dass keine Aufteilung in einen $\mathbf{x}$-Regressor und einen $\mathbf{z}$-Regressor durchgeführt werden kann. Außerdem führt Ermittlung der Kovarianzmatrix häufig zu numerischen Schwierigkeiten [Mur94; Nel01].

Inkrementelle Clustering-Verfahren

Da die Gültigkeitsfunktionen über die Normierung zusammenhängen, ist die Ermittlung der Zentren an Hand der Daten mit anschließender Normierung nicht ganz unproblematisch. Durch die Normierung können Seiteneffekte entstehen [MJ97; Nel01; Har14], wodurch unerwünschtes Verhalten bei der Modellierung auftritt. Die Problematik kann entschärft werden, indem durch ein inkrementelles Setzen der Zentren neben der Ausprägung der Nichtlinearität auch die Auswirkung der Normierung berücksichtigt wird. Als Kriterium, an welchen Stellen lokale Modelle erzeugt werden sollen, wird der lokale Fehler betrachtet. In Bereichen, in denen Verbesserungsbedarf besteht, werden neue lokale Modelle hinzugefügt. Das Hinzufügen lokaler Modelle wird beendet, wenn beispielsweise eine maximale Anzahl lokaler Modelle oder ein vorgegebenes Fehlermaß erreicht ist. Beispiele für inkrementelle Clustering-Verfahren sind in [Mur94; JK06] zu finden.

Heuristische Baumkonstruktionsverfahren

Diese Verfahren laufen ähnlich ab wie die zuvor beschriebenen inkrementellen Clustering-Verfahren. Jedoch wird hier der Eingangsraum bei jedem Schritt geteilt, wodurch sich ebenfalls die Anzahl der lokalen Modelle nach jedem Schritt erhöht. Die Richtung, in die geteilt wird, orientiert sich dabei am Prozessverhalten und erzeugt lokale Modelle in den Bereichen, in denen sie benötigt werden [Nel01; Har14]. Bei den Verfahren wird zwischen einer achsenorthogonalen und einer achsenschrägen Partitionierung unterschieden.

- Achsenorthogonale Partitionierung:
  Ein bekannter Algorithmus zur achsenorthogonalen Partitionierung des Eingangsraums heißt LOLIMOT [Nel01; Har14]. Für eine achsenorthogonale Partitionierung werden Gaußfunktionen mit einer diagonalen Kovarianzmatrix verwendet.
- Achsenschräge Partitionierung:
  Für eine achsenschräge Partitionierung des Eingangsraums werden Sigmoid-Funktionen verwendet, die eine gewisse Ähnlichkeit mit den Basisfunktionen eines MLP-Netzes aufweisen. Um den optimalen Schnitt bei der achsenschrägen Partitionierung zu finden, ist jedoch ein nichtlineares, lokales Optimierungsverfahren notwendig. Der Algorithmus zur achsenschrägen Partitionierung zur Partitionierung heißt HILOMOT [Nel06a; Har+12; Har14]. Eine achsenschräge Partitionierung bringt dann Vorteile, wenn die Nichtlinearität schräg im Raum liegt.

Anwendung von Partitionierungsverfahren zur automatischen Modellerstellung

Für die automatische Modellerstellung erscheint die Verwendung inkrementeller Clustering-Verfahren und heuristischer Baumkonstruktionsverfahren sinnvoll. Diese ermöglichen nämlich eine gleichzeitige Optimierung der Anzahl der Modellparameter durch Betrachtung des Fehlers, der bei Verwendung der Validierungsdaten entsteht. Steigt dieser wieder an, wird das Hinzufügen weiterer lokaler Modelle gestoppt, wodurch eine Strukturoptimierung stattfindet. Ein Beispiel für eine Implementierung ist die LMN-Toolbox der Universität Siegen [Har+12], in der die oben beschriebenen inkrementellen Baumkonstruktionsverfahren verwendet werden. Die inkrementellen Clustering-Verfahren führen auch eine Strukturoptimierung hinsichtlich der Anzahl der lokalen Modelle durch. Allerdings können je nach ausgewähltem Clustering-Verfahren weitere

Hyperparameter hinzukommen. Diese sind zum Beispiel die Einstellparameter des Clustering-Verfahrens, da durch diese die Einteilung des Eingangsraums beeinflusst wird, was einen Einfluss auf die Modellgüte hat.

4.3.3 Gegenüberstellung der Eigenschaften von MLP-Netzen und LM-Netzen

In Tabelle 4.1 sind die Vor- und Nachteile von MLP-Netzen und LM-Netzen gegenübergestellt. Dabei ist ein wichtiger Punkt, der in der Literatur auch als Fluch der Dimensionalität bezeichnet wird [Nel01], die Problematik, dass bei steigender Dimension des Eingangsraums ein starker Anstieg von Modellparametern zu verzeichnen ist. Das hat zur Folge, dass auch deutlich mehr Daten zur Modellerstellung benötigt werden, um ein neuronales Netz mit gewünschter Generalisierungsfähigkeit zu erhalten. Diese Problematik zeigt sich vor allem bei lokalen Modellnetzen. MLP-Netze sind dagegen sehr gut geeignet für hochdimensionale Eingangsräume und nach wie vor der Stand der Technik für solch eine Modellierung. Allerdings sind diese aufwändiger zu erstellen.
In den meisten Anwendungen kann auch die Verwendung von Wissen über den Prozess die beschriebene Problematik bei lokalen Modellnetzen entschärfen. Dazu bietet zum Beispiel die Unterteilung in einen $\mathbf{x}$-Regressor und einen $\mathbf{z}$-Regressor eine Möglichkeit oder die Verwendung lokaler Modelle mit quadratischer oder höherer Ordnung [Nel01; Har14].

4.3.4 Statische Modellbildung des Schaltverhaltens

In diesem Abschnitt wird beschrieben, inwiefern das Schaltverhalten von Stufenautomaten durch ein statisches neuronales Netz modelliert werden kann. Die Idee dabei ist, dass es einen zeitunabhängigen, nichtlinearen Zusammenhang zwischen den Eingangssignalen und den Ausgangssignalen gibt, der durch

	Vorteile	**Nachteile**
MLP-Netz	Auch für hochdimensionale Eingangsräume geeignet Weniger Rechenaufwand zur Bestimmung des Modellausgangs Sehr gute Interpolationseigenschaften	Rechenintensive Parameteroptimierung Aufwändigere Strukturoptimierung Schwer zu interpretierende Netzarchitektur Wissen über das nichtlineare System kann nur schwer eingebracht werden
LM-Netz	Einfachere Struktur- und Parameteroptimierung Wissen über das nichtlineare System kann einfach eingebracht werden	Einschränkungen bei hochdimensionalen Eingangsräumen Bei hochdimensionalen Eingangsräumen mehr Rechenaufwand, da starker Anstieg der Modellparameter

Tabelle 4.1: Vor- und Nachteile von MLP-Netzen und LM-Netzen

das neuronale Netz dargestellt werden soll. Mit anderen Worten, es gibt eine erweiterte Schaltmatrix, in der nicht nur statische, sondern auch dynamische Übersetzungen stehen. Allerdings zeigt eine Untersuchung von [CNG13], dass es dynamische Effekte gibt, die durch diesen Ansatz nicht berücksichtigt werden können. Dies zeigt sich dadurch, dass es keinen eindeutigen Zusammenhang zwischen den Ansteuerströmen während eines Schaltvorgangs und einer Übersetzung gibt. Ein statisches neuronales Netz ist also strukturell nicht in der Lage, die Schaltdynamik darzustellen. Diese nicht eindeutigen Zusammenhänge in den Eingangs- und Ausgangsdaten kann auch nicht die Verwendung von MLP-Netzen mit mehreren verdeckten Schichten auflösen, da diese zur Klassifikation verwendet werden, um auch nichtlinear separierbare Klassen bilden zu können [Hay09]. Als Ergebnis wird ein Modell erstellt, das zwar die statischen Übersetzungen sehr gut beschreibt, jedoch bei den Schaltübergängen große Abweichungen aufweist. Sollen also auch die Schaltübergänge besser beschrieben

werden, muss eine dynamische Modellbildung erfolgen, worauf im nächsten Abschnitt eingegangen wird.

4.4 Dynamische Modellbildung mit neuronalen Netzen

Zur dynamischen Modellbildung gibt es folgende Ansätze, die eine Erweiterung statischer neuronaler Netze darstellen [Nel01]:

- Statische neuronale Netze mit externer Dynamik
- Statische neuronale Netze mit interner Dynamik

Beide Ansätze haben gemeinsam, dass sie zeitdiskrete Gleichungen mit einer festen Abtastzeit T_S besitzen. Auf diese Weise bekommt das neuronale Netz ein Gedächtnis, und es können dynamische Effekte modelliert werden. In den meisten Fällen werden mehrere Zeitverzögerungsoperatoren pro Eingangssignal verwendet, wodurch die dynamische Ordnung des Modells erhöht werden kann. In der Literatur wird dieser Ansatz auch als Tapped-Delay-Line bezeichnet [NMG93; Nel01]. Die verwendeten Zeitverzögerungsoperatoren q^{-1} speichern den Eingangswert eines Abtastzeitschritts und geben diesen um einen Abtastzeitschritt verzögert aus. So wirkt dieser zum Beispiel auf das Eingangssignal $u(k)$ wie folgt

$$\begin{aligned} u(k) &= u(T_0 + k \cdot T_S) \\ u(k-1) &= q^{-1} u(k). \end{aligned} \tag{4.20}$$

Die Verwendung von Zeitverzögerungsoperatoren kann nun bei den Eingangssignalen, beim Ausgangssignal und den Neuronen in den verdeckten Schichten verwendet werden. Bei einem neuronalen Netz mit externer Dynamik werden die Zeitverzögerungsoperatoren bei den Eingangssignalen und dem Ausgangssignal verwendet, um die dynamische Ordnung darzustellen. Bei einem neu-

ronalen Netz mit interner Dynamik werden die Zeitverzögerungsoperatoren ausschließlich in den verdeckten Schichten verwendet [Nel01; Sch10].

4.4.1 Statische neuronale Netze mit externer Dynamik

In diesem Abschnitt wird auf die Erstellung statischer neuronaler Netze mit externer Dynamik eingegangen. Dazu ist in Abbildung 4.9 der allgemeine Aufbau zu sehen.

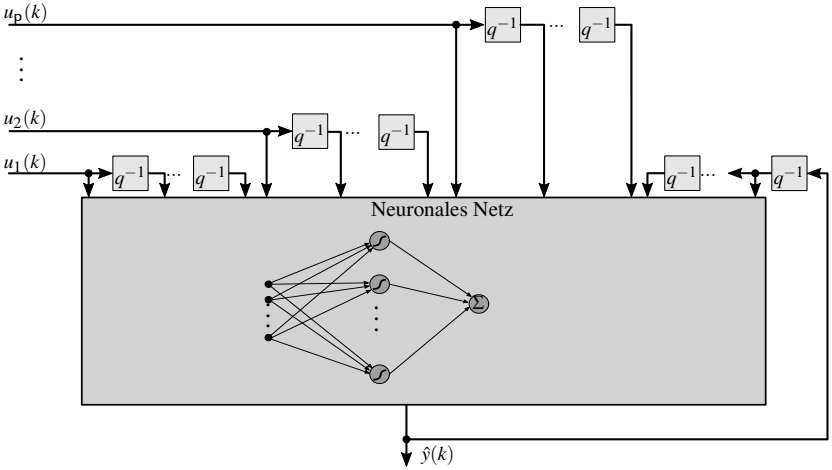


Abbildung 4.9: Neuronales Netz mit externer Dynamik

Das dabei entstehende statische neuronale Netz mit externer Dynamik ist gegeben durch

$$\begin{aligned} \mathbf{u}^k &= \begin{pmatrix} \mathbf{u}_1^k & \mathbf{u}_2^k & \dots & \mathbf{u}_{\mathsf{p}}^k \end{pmatrix} \\ \hat{\mathbf{y}}^k &= \begin{pmatrix} \hat{y}(k) & \hat{y}(k-1) & \dots & \hat{y}(k-n_y) \end{pmatrix} \\ \hat{y}(k+1) &= \mathsf{NN}(\mathbf{u}^k, \hat{\mathbf{y}}^k, \mathbf{w}, \mathsf{P}_1, \mathsf{P}_2, \dots, \mathsf{P}_{\mathsf{K}}). \end{aligned} \tag{4.21}$$

Dabei wird folgende Bezeichnung verwendet

$$\mathbf{u}_j^k = \begin{pmatrix} u_j(k) & u_j(k-1) & \dots & u_j(k-n_j) \end{pmatrix}. \tag{4.22}$$

Dabei werden mit $\mathbf{w}$ die Modellparameter bezeichnet. Weiter stellen die zu wählenden Ordnungen n_j zusätzliche Hyperparameter P dar, die bei der Strukturoptimierung berücksichtigt werden müssen. Dies entspricht der Optimierung der dynamischen Ordnung des Modells. Einerseits steigt zwar die dynamische Ordnung des Modells mit der Anzahl von Zeitverzögerungsoperatoren an, andererseits steigt durch die höhere Dimension des Eingangsraums auch die Anzahl der Modellparameter, wodurch die Modellflexibilität und die Gefahr für eine Überanpassung steigt. Eine weitere Herausforderung ist dabei, dass die Ordnungen für jedes Eingangssignal und das Ausgangssignal individuell gefunden werden müssen, wodurch der Aufwand der Strukturoptimierung weiter ansteigt. Für eine feste Wahl von Hyperparametern werden auch hier die Modellparameter so bestimmt, dass die gegebenen Eingangssignale und das gegebene Ausgangssignal durch das dynamische neuronale Netz mit dem kleinsten möglichen Fehler beschrieben werden. Zur Bestimmung der Modellparameter $\mathbf{w}$ des dynamischen neuronalen Netzes wird daher die Summe der Fehlerquadrate minimiert. Je nach Auswahl des statischen neuronalen Netzes ist dies ein lineares oder ein nichtlineares Optimierungsproblem [Nel01; Sch10]. Zudem steigt die Komplexität des Optimierungsproblems an, je mehr zeitverzögerte Signale verwendet werden (Fluch der Dimensionalität).
Bei der Verwendung externer Zeitverzögerungsoperatoren werden zudem folgende Fälle unterschieden:

- Modellierung mit zeitverzögerten Modellausgängen
- Modellierung ohne zeitverzögerte Modellausgänge

Im Folgenden werden die Auswirkungen dieser beiden Fälle auf die Modellerstellung beschrieben. Dazu wird eine Schreibweise eingeführt, mit der die

externen Zeitverzögerungsoperatoren kompakter dargestellt werden können. Für die Schreibweise wird ein beliebiges Eingangssignal u_j aus $\mathbf{u}$ des MISO-Systems betrachtet. Durch Verwendung von n_j Zeitverzögerungsoperatoren entsteht ein (n_j+1)-dimensionaler Signalvektor $\mathbf{u}_{\mathrm{TD},j}$ mit zeitverzögerten Signalen (siehe Abbildung 4.10), die als zusätzliche Eingangssignale im statischen neuronalen Netz verwendet werden. Die Erstellung dieses Signalvektors kann folgendermaßen geschrieben werden, wobei n_j die Anzahl der Zeitverzögerungsopertoren bzw. die Ordnung beschreibt

$$\mathrm{TD}(u_j, n_j) = \mathbf{u}_{j,\mathrm{TD}}(k). \tag{4.23}$$

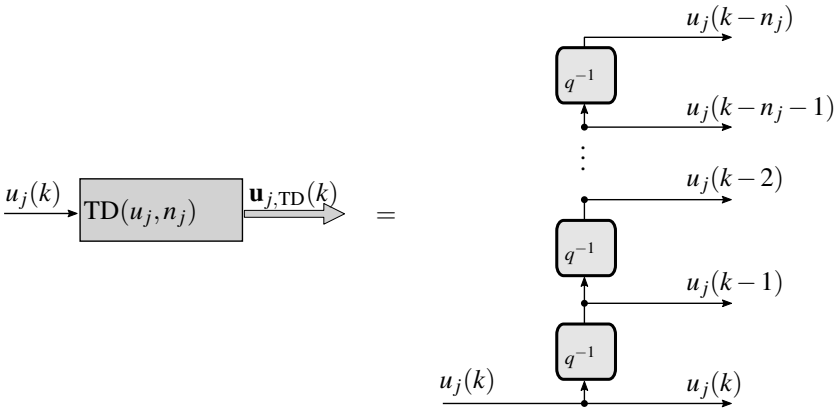


Abbildung 4.10: Symbol TD(u_j,n_j) zur Darstellung zeitverzögerter Signale

Modellierung mit zeitverzögerten Modellausgängen

In diesem Abschnitt wird auf die Modellierung mit zeitverzögerten Ausgangssignalen eingegangen. Dieser Modellierungsansatz hat den Vorteil, dass bereits

eine geringe Anzahl von zeitverzögerten Eingangssignalen und Ausgangssignalen ausreicht, um das dynamische Verhalten von Systemen modellieren zu können. Somit steigt die Anzahl der Modellparameter nicht übermäßig an. Dadurch ist auch die Durchführung der Strukturoptimierung erleichtert, da hier weniger Ordnungen berücksichtigt werden müssen. Die theoretische Begründung ist bei der Ähnlichkeit zu Filtern mit unendlich langen Impulsantworten (IIR-Filter[5]) zu finden. Dies bedeutet, dass durch eine geringe Anzahl von Zeitverzögerungsoperatoren eine beliebig lange Impulsantwort erzeugt werden kann, die zur Modellierung des dynamischen Systems verwendet werden kann. Der Nachteil dieses Ansatzes ist hingegen, dass es keine garantierte Stabilität bei der Verwendung in Simulationen gibt, da sich durch die Rückführung Fehler aufsummieren können. Die Modellgüte hängt zudem stark von der genauen Wahl der Anzahl der Zeitverzögerungen und der gewählten Abtastzeit ab. Im schlimmsten Fall können auch weitere Instabilitäten auftreten.
Die Parameteroptimierung ist nun deutlich aufwändiger, und die oben beschriebenen Verfahren können nicht ohne weiteres verwendet werden. Für die Verwendung der Modelle in einer Simulation muss auch der Fehler, der durch die Rückführung der Modellausgänge entsteht, berücksichtigt werden. Die Parameteroptimierung ist jedoch ein sehr wichtiger Aspekt besonders im Hinblick auf eine automatische Erstellung numerisch vereinfachter Getriebemodelle. Im Folgenden werden daher die Probleme näher beschrieben, die bei der Parameteroptimierung dynamischer neuronaler Netze mit Rückführung von Modellausgängen entstehen können. Dabei wird nach [Nel01] zwischen folgenden zwei Modellkonfigurationen unterschieden:

1. Serien-parallele Modellkonfiguration:
   Bei dieser Modellkonfiguration werden zeitverzögerte Ausgangssignale des dynamischen Systems verwendet.

[5] IIR-Filter ... engl. Infinite Impulse Response

2. Parallele Modellkonfiguration:
   Bei dieser Modellkonfiguration werden zeitverzögerte Modellausgänge verwendet.

Der Unterschied ist in Abbildung 4.11 graphisch dargestellt.

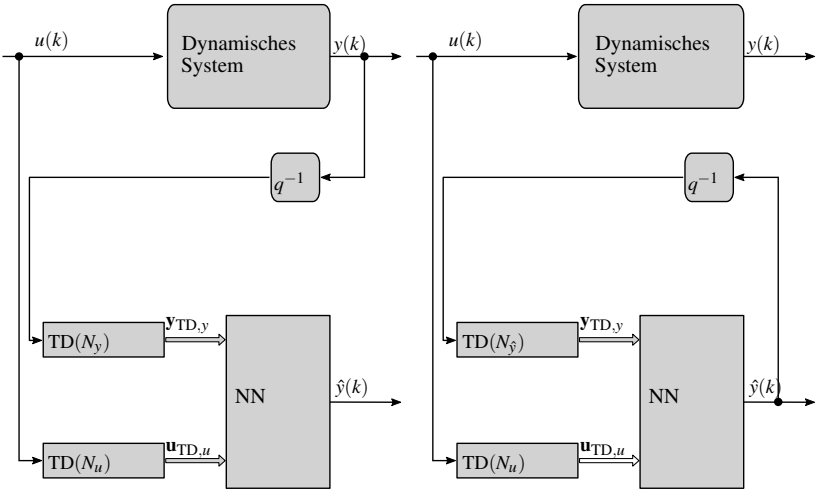


Abbildung 4.11: Serien-parallele und parallele Modellstruktur

Im Folgenden werden die Unterschiede dieser beiden Modellkonfigurationen beschrieben, die besonders für die Verwendung als Simulationsmodell relevant sind. Die serien-parallele Konfiguration entspricht einem Einschritt-Prädiktor. Es werden bekannte, vorherige Ausgangssignale aus einer Messung und die dazugehörigen Eingangssignale verwendet, um den Ausgangswert für den nächsten Abtastzeitpunkt vorherzusagen. Dabei können die Modellparameter wie im statischen Fall ermittelt werden. Das bedeutet, dass die oben beschriebenen Verfahren zur Parameteroptimierung weiterhin verwendet werden können. Dies ist allerdings in Simulationen nicht möglich, da keine bekannten Ausgangssignale

verwendet werden dürfen. Die Verhältnisse ändern sich jedoch, wenn eine parallele Modellstruktur verwendet wird. Dann werden zuvor berechnete Modellausgänge zeitverzögert verwendet, wodurch ein Modell entsteht, das auch in der Simulation verwendet werden kann. Allerdings steigt der Aufwand bei der Parameteroptimierung an, da nun ein implizites Optimierungsproblem vorliegt. Diese Problematik kann beispielhaft an einem linearen dynamischen System erster Ordnung illustriert werden. Die serien-parallele Modellkonfiguration ist dann gegeben durch

$$\hat{y}(k+1) = (1-a) \cdot u(k) + a \cdot y(k), \tag{4.24}$$

und es liegt ein lineares Optimierungsproblem vor. Die parallele Modellkonfiguration ist im Gegensatz dazu gegeben durch

$$\begin{aligned} \hat{y}(k+1) &= (1-a) \cdot u(k) + a \cdot \hat{y}(k) \\ &= (1-a) \cdot u(k) + a \cdot ((1-a) \cdot u(k-1) + a \cdot \hat{y}(k-1)) \\ &\ldots . \end{aligned} \tag{4.25}$$

Da nun das gesuchte Modell auf beiden Seiten des Gleichungssystems steht, entsteht ein implizites Optimierungsproblem. Sollen die Parameter einer parallelen Modellkonfiguration optimiert werden, sind aufwändigere Verfahren notwendig. Ein Beispiel dafür ist das Backpropagation-Through-Time-Verfahren [RHW86; Nel01; Sch10]. Die Idee ist hierbei, das dynamische neuronale Netz in der Zeit auszufalten. Dabei entsteht ein mehrschichtiges statisches neuronales Netz, dessen Parameter mit Verfahren der nichtlinearen lokalen Optimierung bestimmt werden können. Da durch das Ausfalten des neuronalen Netzes die Anzahl der verdeckten Schichten ansteigt, ist auch der Aufwand zur Parameteroptimierung deutlich höher, da dieser exponentiell mit der Anzahl der verdeckten Schichten steigt. Zudem gibt es keine Garantie, dass die Parameteroptimierung konvergiert [Nel01].

Für die Verwendung in einer Simulation, wie es in dieser Arbeit angedacht ist, muss eine parallele Modellkonfiguration optimiert werden, da die realen, zeitverzögerten Ausgänge nicht vorliegen. Ein häufig in der Praxis anzutreffender Ansatz ist, dass die Parameter einer serien-parallelen Modellkonfiguration optimiert werden und anschießend das erstellte Modell als parallele Modellkonfiguration verwendet wird. Dies führt in den meisten Fällen zu zufriedenstellenden Ergebnissen. Jedoch kann es passieren, dass die Güte des parallelen Modells verglichen mit dem serien-parallelen deutlich schlechter ist [Nel01]. In solchen Fällen ist die Optimierung des Ein-Schritt-Prädiktors unzureichend, und die Fehler summieren sich rasch auf. In manchen Fällen wird das erstellte Modell auch instabil, wie unter anderem in folgenden Arbeiten gezeigt wird: [JSM00; Nel01; Hof03; ZI08; Ren12].

Modellierung ohne zeitverzögerte Modellausgänge

In diesem Abschnitt wird auf die Vor- und Nachteile eingegangen, wenn keine zeitverzögerten Ausgangssignale verwendet werden. Der Vorteil ist hier, dass es eine garantierte Stabilität des erstellten Modells in einer numerischen Simulation gibt. Außerdem müssen keine dynamischen Gradienten berechnet werden, um die Parameter einer parallelen Modellstruktur zu optimieren.
Nachteilig ist, dass eine Vielzahl von zeitverzögerten Eingangssignalen benötigt wird, um die Dynamik zu erfassen. Dies ist theoretisch dadurch begründet, dass ein nichtlinearer Filter mit endlicher Impulsantwort (FIR-Filter[6]) vorliegt [Nel01]. Deshalb orientiert sich die Anzahl der benötigten, zeitverzögerten Signale an der Zeitkonstanten des Systems und der verwendeten Abtastzeit. Die hohe Anzahl ist deshalb notwendig, weil der gesamte Zeitraum abgedeckt sein muss, den das dynamische System zum Erreichen eines stationären Zustands

[6] FIR-Filter ... Finite Impulse Response Filter

benötigt. Nur unter dieser Voraussetzung kann das gesamte dynamische Übertragungsverhalten modelliert werden.
Beträgt beispielsweise die Zeitkonstante $1\,[\mathrm{s}]$ und es soll eine Abtastzeit von $\frac{1}{15}\,[\mathrm{s}]$ verwendet werden, sind 15 zusätzliche zeitverzögerte Eingangssignale zu berücksichtigen, um die Dynamik erfassen zu können. In den meisten Anwendungen sind es zudem viele Eingangssignale, so auch beim Stufenautomaten. Liegen beispielsweise 10 Eingangssignale vor, müssen beim neuronalen Netz insgesamt 150 zeitverzögerte Eingangssignale verwendet werden. Damit steigen die Anzahl der zu bestimmenden Modellparameter und die Dimension des Eingangsraums stark an (Fluch der Dimensionalität).
Einen Ausweg aus dieser Problematik bietet die Verwendung externer orthonormal-basis-function-Filter[7], die im Folgenden als OBF-Filter bezeichnet werden und beispielsweise in [Nel01; Hof03; Sch10; TR12] beschrieben sind. Diese werden dazu verwendet, um die Übertragungsfunktion eines dynamischen Systems durch eine Reihenentwicklung aus den Impulsantworten der einzelnen OBF-Filter $f_i(q)$ darzustellen[8]. Mathematisch entspricht dies der Entwicklung einer gesuchten und unbekannten Funktion in Basis-Funktionen, die eine orthonormale Basis bilden. Für ein dynamisches SISO-System ist diese nichtlineare Überlagerung gegeben durch

$$\hat{y} = \mathsf{NN}\left(f_1(q)u(k), f_2(q)u(k), \ldots, f_\eta(q)u(k), \mathbf{w}\right). \tag{4.26}$$

Bei der Auswahl der OBF-Filter kann vorhandenes Wissen über das Verhalten des dynamischen Systems in die Modellierung eingebracht werden, da durch die externen Filter ein bestimmtes dynamisches Verhalten beschrieben wird. Durch die Verwendung externer Filter bleiben die oben beschriebenen Vorteile von Modellen ohne Ausgangsrückführung bestehen. Jedoch wird eine Reduk-

[7] OBF-Filter ... orthonormal-basis-function-Filter: Filter, deren Übertragungsfunktionen eine orthonormale Basis bilden.

[8] Die Zeitverzögerungsoperatoren sind ein Spezialfall von OBF-Filtern, die einen Pol bei null besitzen [Nel01].

tion der Anzahl der Parameter des dynamischen neuronalen Netzes erreicht, da nun zur Modellierung wesentlich weniger externe Filter verwendet werden können als externe Zeitverzögerungsoperatoren. In [Hof03] wird gezeigt, dass eine Reduktion der Modellparameter von 3600 auf 400 möglich ist, was für die Modellerstellung ein sehr großer Gewinn ist, da die Validierung deutlich einfacher wird. Zudem ist die Gefahr für eine Überanpassung eingegrenzt.
Ein Beispiel für OBF-Filter sind Laguerre-Filter, die zur Modellierung stark gedämpfter Systeme verwendet werden können. Als stark gedämpftes System wird dabei ein System bezeichnet, das auf sich schnell ändernde Eingangssignale gedämpft reagiert. Mit Hilfe der Laguerre-Filter kann solch ein Verhalten durch Überlagerung der einzelnen Ordnungen modelliert werden. Die einzelnen Ordnungen der Laguerre-Filter sind gegeben durch [Wah91b; Nel01; TR12]

$$L(q,\alpha,i) = \sqrt{1-\alpha^2}\frac{(1-\alpha q)^{i-1}}{(q-\alpha)^i},\ |\alpha| < 1. \tag{4.27}$$

Der Index i variiert zwischen 1 und η, wobei η für die Ordnung des Laguerre-Filters steht. Für die Modellerstellung stellen die Ordnungen η und Pole α von Laguerre-Filtern zusätzliche Hyperparameter dar, die gefunden und bei der Strukturoptimierung berücksichtigt werden müssen.
Ein weiteres Beispiel ist die Verwendung von Kautz-Filtern für schwach gedämpfte Systeme. Bei dieser Art dynamischer Systeme wird ein oszillatorisches Verhalten angeregt, das nur sehr langsam abklingt [Wah91a; Nel01; TR12]. Da in dieser Arbeit Laguerre-Filter zur Modellierung der Schaltdynamik zum Einsatz kommen, wird auf Kautz-Filter hier nicht weiter eingegangen.

4.4.2 Statische neuronale Netze mit interner Dynamik

Zum Abschluss wird auf statische neuronale Netze mit interner Dynamik eingegangen. Bekannte Vertreter neuronaler Netze mit interner Dynamik sind das

Elman-Netz und das Jordan-Netz [Nel01]. Bei diesen bleibt die Anzahl der Eingangssignale unverändert, und die Zeitverzögerungsoperatoren bzw. Übertragungsfunktionen werden nur intern verwendet[9]. Dies stellt einen Vorteil für die Erstellung von Modellen dar, da so nicht die Dimension des Eingangsraums ansteigt. Nachteilig ist jedoch, dass die Parameteroptimierung hier sehr schwierig ist und häufig nicht konvergiert [Nel01]. Somit ist es schwer möglich, eine automatisierte Modellerstellung zu realisieren.

4.4.3 Überblick über dynamische Modellierung

In Tabelle 4.2 sind die Vor- und Nachteile der verschiedenen Ansätze zur dynamischen Modellbildung gegenübergestellt. Für die Verwendung der erstellten neuronalen Netze in einer Simulation ist der Ansatz mit externen Filtern vorteilhaft, da so eine Entkopplung von dynamischer und statischer Modellierung stattfindet. So werden die Filter an die Ordnung der Dynamik des Systems angepasst, und durch das neuronale Netz können anschließend statische nichtlineare Zusammenhänge berücksichtigt werden. Dabei ist auch vorteilhaft, dass keine dynamischen Gradienten berechnet werden müssen, sondern zur Parameteroptimierung die gleichen Verfahren wie zur Erstellung statischer neuronaler Netze verwendet werden können. Bei einem FIR-Ansatz ist dies zwar ähnlich, jedoch wird hier eine deutlich größere Anzahl von Zeitverzögerungsoperatoren benötigt, um das dynamische Verhalten zu beschreiben.
Die Verwendung von neuronalen Netzen mit interner Dynamik bzw. mit Rückführung von Modellausgängen bedeutet einen deutlich größeren Aufwand bei der Parameteroptimierung. Weiter ist auch nicht garantiert, dass die Parameteroptimierung konvergiert. Außerdem besteht die Gefahr, dass die erstellten Modelle in der Simulation instabil werden. Somit ist die Verwendung neuro-

[9] Bei genauer Betrachtung können von einem lokalen Modellnetz die zeitverzögerten Signale im **x**-Regressor als interne Dynamik interpretiert werden [NMG93].

naler Netze mit externen OBF-Filtern ein vielversprechender Ansatz, um eine automatische numerische Vereinfachung durchzuführen und die so erstellten Modelle in der Simulation zu verwenden.

	Vorteile	**Nachteile**
IIR-Struktur	Geringe Anzahl von Zeitverzögerungsoperatoren	Dynamische Gradienten Möglicherweise instabil bei der Verwendung in der Simulation Teilweise starke Abhängigkeit von der gewählten Abtastzeit T_S
FIR-Struktur	Stabil in der Simulation Keine dynamischen Gradienten	Hohe Anzahl von Zeitverzögerungsoperatoren Teilweise starke Abhängigkeit von der gewählten Abtastzeit T_S Starker Anstieg der Modellparameter
OBF-Struktur	Stabil in der Simulation Verwendung geringer Anzahl von externen Filtern Weniger Einschränkungen bei der Wahl der Abtastzeit T_S	Zusätzliche Hyperparameter der externen Filter
Interne Dynamik	Keine Verwendung von externen Filtern bzw. Zeitverzögerungsoperatoren	Dynamische Gradienten Aufwändige Parameteroptimierung

Tabelle 4.2: Vor- und Nachteile der Ansätze zur dynamischen Modellierung mit neuronalen Netzen

4.4.4 Dynamische Modellbildung des Schaltverhaltens

Im Folgenden wird auf verschiedene Ansätze eingegangen, die zur dynamischen Modellierung des Schaltverhaltens verwendet werden können. Dabei werden die Vor- und Nachteile der jeweiligen Ansätze dargestellt. In [CNG13]

ist beschrieben, dass die Verwendung zeitverzögerter Ausgänge zu instabilem Verhalten in der Simulation führt. Zudem ist die Wahl der Abtastzeit sehr stark eingeschränkt, so dass auch hier keine zufriedenstellende Modellierung möglich ist. Zur Minimierung des Fehlers, der durch die Rückführung von Modellausgängen entsteht, muss auf spezielle Verfahren zur Parameteroptimierung zurückgegriffen werden. Diese erhöhen jedoch den Aufwand zur Modellerstellung. Zudem ist nicht garantiert, dass diese Verfahren konvergieren. In [CNG14] wird gezeigt, dass durch die Modellierung mit externen Laguerre-Filtern gute Resultate erzielt werden können. Dies bedeutet, dass auf die Rückführung von Modellausgängen verzichtet werden kann. Da es sich hier um ein System mit vielen Eingangssignalen handelt, ist die Reduktion der Modellparameter durch die Verwendung externer Laguerre-Filter besonders signifikant im Vergleich zur Verwendung eines reinen FIR-Modells. Zur Modellierung der Schaltdynamik können Laguerre-Filter verwendet werden, da es sich um einen stark gedämpften Prozess handelt. Dies zeigt sich durch Vergleich des zeitlichen Verlaufs der Ansteuerströme mit dem Verlauf des Drehmoments bzw. der Winkelgeschwindigkeit, wobei sich die Ansteuerströme deutlich schneller ändern als dies Auswirkungen auf das Drehmoment bzw. die Winkelgeschwindigkeit hat (siehe Abbildung 3.8).

4.5 Fazit

In diesem Kapitel wurde gezeigt, wie neuronale Netze zur Modellierung des dynamischen und des statischen Schaltverhaltens von Stufenautomaten verwendet werden können. Dabei wurde insbesondere diskutiert, welcher Ansatz sich am besten zur Modellierung eignet und eine numerische Vereinfachung sowie eine automatische Modellerstellung ermöglicht. Da eine statische Modellbildung nicht ausreicht, um das Schaltverhalten zu modellieren, wurden verschiedene Ansätze zur dynamischen Modellierung mit neuronalen Netzen vorgestellt und

miteinander verglichen. Wichtige Kriterien zur Auswahl waren dabei die Stabilität in der Simulation, der Aufwand zur Modellerstellung und das Potential zur numerischen Vereinfachung. Dabei hat sich herausgestellt, dass die Verwendung externer Laguerre-Filtern in Kombination mit neuronalen Netzen ein vielversprechender Ansatz zur automatischen numerischen Vereinfachung von Getriebemodellen ist. Für die Simulation entsteht dabei ein garantiert stabiles Modell. Zudem ist eine numerisch effiziente Implementierung neuronaler Netze mit externen Laguerre-Filtern möglich, wodurch eine numerische Vereinfachung erzielt werden kann.

5 Automatische Erstellung neuronaler Netze mit externen Laguerre-Filtern

Ziel dieses Kapitels ist die Erarbeitung eines Verfahrens zur automatischen Erstellung neuronaler Netze mit externen Laguerre-Filtern. Dazu wird zunächst auf die Modellierung linearer dynamischer Systeme mit Laguerre-Filtern eingegangen und erläutert, welche Rolle dabei die Wahl der Ordnungen und der Pole spielen. Die Ordnungen und Pole stellen Hyperparameter dar, die bei einer automatischen Modellerstellung neben den Modellparametern ermittelt werden müssen. Um dies zu erleichtern, wird gezeigt, wie die Zeitkonstante eines dynamischen Systems mit der Wahl des Pols zusammenhängt.

Anschließend wird beschrieben, wie die Kombination von Laguerre-Filtern und neuronalen Netzen zur Modellierung nichtlinearer dynamischer Systeme verwendet werden kann. Dabei werden verschiedene Ansätze diskutiert, um die Hyperparameter, die aus den Polen und Ordnungen der externen Laguerre-Filter und der Netzarchitektur bestehen, zu optimieren. Als Ergebnis dieser Diskussion wird die Verwendung einer Liniensuche vorgestellt, die bei der automatischen Erstellung neuronaler Netze zum Einsatz kommt.

5.1 Modellierung linearer dynamischer Systeme mit Laguerre-Filtern

Wie bereits beschrieben, eignen sich Laguerre-Filter besonders gut zur Modellierung stark gedämpfter dynamischer Systeme. Dabei wird unter einem stark

gedämpften Verhalten ein System verstanden, das auf sich schnell ändernde Eingangssignale träge reagiert. Wird solch ein System mit einem Sprung angeregt, passt sich dieses verzögert an das neue Niveau an. Die Sprungantwort kann nun dazu verwendet werden, die Zeitkonstante T_K des dynamischen Systems zu definieren. Eine mögliche Definition ist dabei, dass die Zeitkonstante T_K der Zeit entspricht, zu der das Ausgangssignal 63 % des Niveaus des Sprungsignals erreicht hat [LW07; RB13].
Eine weitere Eigenschaft eines stark gedämpften Systems ist, dass bei Anregung dieser Systeme keine bzw. nur sehr schwache Oszillationen auftreten. Ein Beispiel für solch ein System ist der aus der Elektrotechnik bekannte Aufladevorgang eines Kondensators [Jan09]. Ein weiteres Beispiel ist ein harmonischer Oszillator mit hohem Dämpfungsfaktor (siehe Kapitel 2).

Laguerre-Filter können nun folgendermaßen verwendet werden, um das Übertragungsverhalten solcher dynamischer Systeme zu modellieren. Dazu wird je Eingangssignal ein Laguerre-Filter $L(q, \alpha_i, \eta_i)$ verwendet, wobei der Pol α_i und die Ordnung η_i verwendet werden. Die gefilterten Signale werden mit den in Gleichung (4.27) beschriebenen Übertragungsfunktionen berechnet. Dabei entstehen pro Eingangssignal η gefilterte Signale, deren Verlauf durch den Pol α bestimmt wird. Der Verlauf der gefilterten Signale ist beispielhaft in Abbildung 5.1 zu sehen, in der ein Sprungsignal durch einen Laguerre-Filter gefiltert wurde. Dabei ist das für stark gedämpfte Systeme typische Dämpfungsverhalten zu sehen. Durch den Pol α kann der Charakter des Dämpfungsverhaltens beeinflusst werden. Zudem wird durch Erhöhung der Ordnungen die dynamische Ordnung des Modells erhöht.

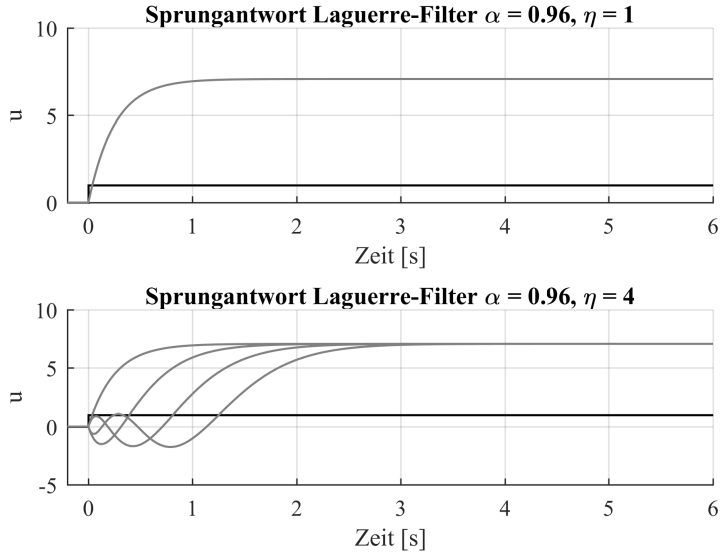


Abbildung 5.1: Sprungantworten der ersten vier Ordnungen von Laguerre-Filtern ($T_S = 0{,}01$ [s])

Das statische und dynamische Verhalten eines linearen dynamischen MISO-Systems kann durch eine Überlagerung der gefilterten Eingangssignale beschrieben werden und ist gegeben durch [Nel01; Sch10]

$$\hat{y}(k) = \sum_{i=1}^{\mathrm{p}} \sum_{j=1}^{\eta_i} w_{i,j} \underbrace{L(q, \alpha_i, j) u_i(k)}_{u_{f,i,j}(k)}, \tag{5.1}$$

wobei die Modellparameter $w_{i,j}$ und $u_{f,i,j}(k)$ die gefilterten Eingangssignale sind. Durch die Modellparameter $w_{i,j}$ findet eine Gewichtung der gefilterten Eingangssignale statt, um so das dynamische Verhalten zu modellieren. Die Pole α_i und die Ordnungen η_i stellen Hyperparameter dar. Sind diese fest

vorgegeben, kann zur Bestimmung der Modellparameter $w_{i,j}$ das Least-Squares-Verfahren verwendet werden (siehe Kapitel 4). Dazu werden die gefilterten Signale $u_{f,i,j}(k)$ in die Messmatrix $\mathbf{X}$ aus Gleichung (4.10) geschrieben, mit der dann die Modellparameter $w_{i,j}$ bestimmt werden können. Die optimalen Werte für die Pole und die Ordnungen müssen durch eine Strukturoptimierung ermittelt werden. Später wird in diesem Kapitel gezeigt, wie dazu eine Liniensuche verwendet werden kann.
Die Strukturoptimierung kann erleichtert werden, da es einen Zusammenhang zwischen dem Pol der Laguerre-Filter und der Zeitkonstante T_K des zu modellierenden dynamischen Systems gibt. Dies kann verwendet werden, um die Größenordnungen des Pols α abzuschätzen. Dabei hängen Pol α, Zeitkonstante T_K und Abtastzeit T_S der zeitdiskreten Laguerre-Filter folgendermaßen zusammen

$$\alpha = \exp\left(\frac{-T_\mathrm{S}}{T_\mathrm{K}}\right). \tag{5.2}$$

Wie oben beschrieben, gibt die Zeitkonstante T_K dabei an, nach welcher Zeit das gefilterte Signal der ersten Ordnung der Laguerre-Filter auf etwa 63 Prozent des Niveaus der Sprungantwort angestiegen ist. Eine graphische Darstellung ist in Abbildung 5.2 zu sehen, in der die ersten vier durch Laguerre-Filter gefilterten Sprungsignale bei verschiedenen Zeitkonstanten dargestellt sind.
Dabei wurden normierte Sprungantworten verwendet, um die Auswirkung der Zeitkonstanten besser diskutieren zu können, da Signale durch die Anwendung von Laguerre-Filtern zusätzlich verstärkt werden (siehe Abbildung 5.1). Die dabei verwendete Normierung ist weiter unten gegeben (siehe Gleichung 5.6) und bewirkt, dass die gefilterten Signale auf Werte zwischen null und eins normiert werden. Die Vorteile dieser Normierung werden weiter unten beschrieben.
Beim Vergleich der verschiedenen Sprungantworten wird deutlich, dass die erste Ordnung der Laguerre-Filter durch die vorgegebene Zeitkonstante T_K beschrieben werden kann. Die höheren Ordnungen erreichen den Sättigungswert des Sprungsignals später und können dazu verwendet werden, das Übertragungsver-

halten des dynamischen Systems besser zu modellieren. Dies zeigt, dass durch Kenntnis der Größenordnung der Zeitkonstante eines dynamischen Systems auch die Größenordnung des Pols der Laguerre-Filter bestimmt werden kann.

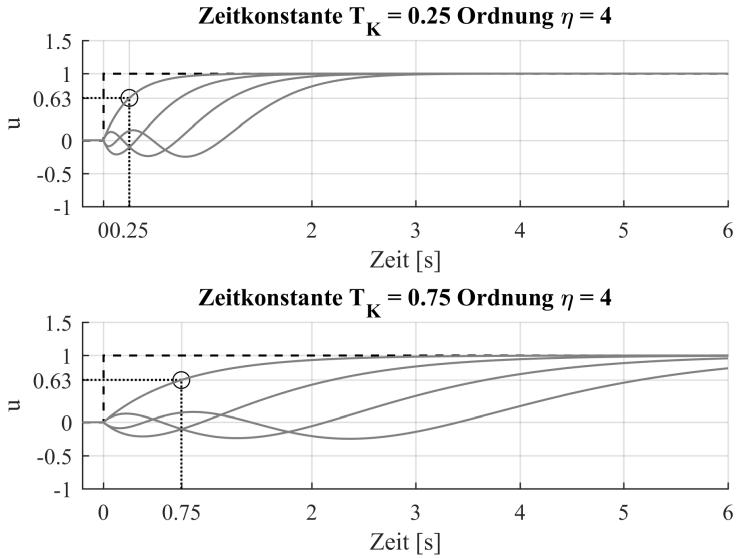


Abbildung 5.2: Normierte Sprungantworten von Laguerre-Filtern der Ordnung $\eta = 4$ für verschiedene Zeitkonstanten ($T_S = 0,01$ [s])

5.1.1 Ausdünnen der Trainingsdaten

Im Folgenden wird gezeigt, wie der Aufwand der Parameteroptimierung durch Ausdünnen der Trainingsdaten gesenkt werden kann. Dabei wird davon ausgegangen, dass die Zielanwendung der erstellten Modelle eine Simulation ist und entsprechend kleine Abtastzeiten T_S in den Laguerre-Filtern verwendet werden

sollen. Dies bedeutet, dass die gefilterten Trainingsdaten mit einer vergleichsweise kurzen Abtastzeit verwendet werden. Die Ausdünnung der gefilterten Simulationsdaten entsteht dadurch, dass statt jeder Zeile der Messmatrix nur jede *n*-te Zeile verwendet wird, wodurch der Aufwand für die Parameteroptimierung durch Bestimmung der Pseudo-Inversen merklich sinkt [1].

Solche eine Ausdünnung kann beispielsweise dadurch gerechtfertigt werden, dass die gefilterten Sprungantworten mit einer Abtastzeit von $T_S = 0,01[\mathrm{s}]$ so fein abgetastet sind, dass durch Ausdünnung kein Informationsverlust zu befürchten ist. Begründet wird dies mit dem Nyquist-Theorem, das besagt, dass die Abtastfrequenz eines Signals mindestens doppelt so groß sein muss, wie die höchste enthaltene Frequenz im Signal [LW07; RB13]. Umgekehrt bedeutet dies, die Abtastrate darf höchstens halb so groß sein wie die Periodendauer der höchsten Frequenz im Signal. Da die Laguerre-Filter jedoch als Tiefpass-Filter wirken, sind oberhalb einer bestimmten Grenzfrequenz keine Frequenzen mehr in den gefilterten Signalen enthalten. Die Ausdünnung bedeutet für die Simulation keine Einschränkung. Dort werden die gefilterten Signale mit der ursprünglichen Abtastzeit erstellt.
Für die Modellierung eines linearen dynamischen Systems kann die Ausdünnung folgendermaßen verwendet werden. Die Trainingsdaten, die zur Parameteroptimierung verwendet werden, können auf eine neue, größere Abtastzeit umgerechnet werden. Als obere Grenze wird dabei die Zeitkonstante T_K der Laguerre-Filter verwendet. Um eine ausreichende Abtastung der gefilterten Signale zu gewährleisten, wird als Richtwert etwa der halbe Wert der Zeitkonstanten T_K verwendet.

[1] In den Zeilen der Messmatrix stehen die Eingangssignale der betrachteten N Abtastpunkte (vgl. Gleichung (4.9)).

5.1.2 Modellierung eines linearen dynamischen Systems

Nun wird gezeigt, wie durch Verwendung der Laguerre-Filter ein lineares dynamisches SISO-System modelliert werden kann. Dazu wird als Beispiel die Sprungantwort y eines stark gedämpften linearen dynamischen Systems zweiter Ordnung (PT2-System [LW07]) in Abhängigkeit von einer externen Anregung $u(t)$ verwendet. Das dynamische Verhalten eines PT2-Systems ist gegeben durch folgende umgeformte zeitkontinuierliche Modellgleichung (siehe Anhang A.1):

$$\begin{aligned} \ddot{y}(t) + \frac{1}{\tau} \cdot \dot{y}(t) + \tilde{\omega}^2 \cdot y(t) &= \tilde{\omega}^2 \cdot u(t) \\ \tilde{\omega}^2 &= (2\pi f_{\mathsf{D}})^2 + \frac{1}{\tau^2} \end{aligned} \tag{5.3}$$

Dabei beschreibt f_{D} die Frequenz der erzeugten Schwingung, die bei Anregung durch ein Sprungsignal entsteht, und τ die Abklingzeit der Schwingung.
Das dynamische Verhalten solch eines gedämpften PT2-Systems soll nun durch eine lineare Überlagerung von durch Laguerre-Filter gefilterten Signalen beschrieben werden. Das zugehörige zeitdiskrete Modell ergibt sich aus Gleichung (5.1) mit $\mathsf{p} = 1$. Zur Bestimmung der optimalen Ordnung und des optimalen Pols wurde die weiter unten vorgestellte Liniensuche mit stochastischen Elementen verwendet. Für die Simulation wurden dabei folgende Parameter des PT2-Systems verwendet:

- $f_{\mathsf{D}} = 0,25 \left[\frac{1}{\mathrm{s}}\right]$,
- $\tau = 0,1 \left[\frac{1}{\mathrm{s}}\right]$.

Die für die Struktur- und Parameteroptimierung notwendigen Simulationsdaten wurden durch 140 verschiedene Sprungsequenzen erzeugt, deren Amplituden zufällige Werte zwischen null und eins annahmen und für $10[\mathrm{s}]$ gehalten wurden, so dass die angeregte Schwingung abklingen konnte. Zudem wurden sie

mit einer Abtastzeit von $T_S = 0,01$ [s] aufgezeichnet. Somit sind in den Simulationsdaten genügend Informationen über das dynamische und das statische Verhalten des Sprungverhaltens enthalten. Zur Modellerstellung wurde die weiter unten beschriebene Liniensuche verwendet, mit folgenden Einstellungen und Suchräumen:

- Anzahl der Durchläufe: 4
- Suchbereich Ordnung $\eta = \begin{pmatrix} 3 & 5 & \dots & 39 \end{pmatrix}$.
- Suchbereich Zeitkonstante T_K: Aufteilung des Bereichs $0,1$ bis 1 in elf gleichgroße Intervalle.

Damit ergaben sich insgesamt etwa 130 Iterationen, die zur Optimierung des Pols und der Ordnung durchgeführt wurden. Der Verlauf der Strukturoptimierung ist in Abbildung 5.3 zu sehen.
Dabei wird deutlich, dass das Minimum nach bereits 20 Iterationen gefunden wird. Danach wird kaum noch eine Verbesserung der Modellgüte erzielt.
In Abbildung 5.4 ist zu sehen, wie mit steigender Ordnung bei optimaler Zeitkonstanten die Approximationsgüte der Sprungantwort weiter zunimmt. Dabei ist zu erkennen, dass bei Ordnung $\eta = 9$ bereits eine gute Überstimmung erreicht ist. Bei Ordnung $\eta = 15$ ist schließlich eine sehr gute Übereinstimmung erreicht. Eine höhere Ordnung bewirkt kaum eine Verbesserung der Modellgüte.
In Anhang A.1 wird auf dieses Beispiel näher eingegangen und gezeigt, wie die Liniensuche das Minimum der Kostenfunktion findet.
Zum Abschluss dieses Abschnitts wird darauf eingegangen, wie die Dauer der Liniensuche in diesem Beispiel durch Ausdünnung der aufgezeichneten Simulationsdaten verkürzt werden konnte. Die Zeitdauer der Simulationszeit war 2100[s], wodurch eine sehr große Menge an Daten für die Parameteroptimierung entsteht ($\mathsf{N} = 210001$). Deshalb wurden ausgedünnte Simulationsdaten für die Parameteroptimierung verwendet, um zu überprüfen, ob das Auswirkungen auf die Modellgüte hat. Dabei war bei Ausdünnungen bis zu einem Wert von $n = 10$

keine Verschlechterung der Modellgüte festzustellen, da hier eine immer noch genügend hohe Abtastung der Dynamik gegeben war. Dies deckt sich gut mit der oben genannten Abschätzung, wie groß die Abtastzeit der ausgedünnten Daten maximal sein darf.

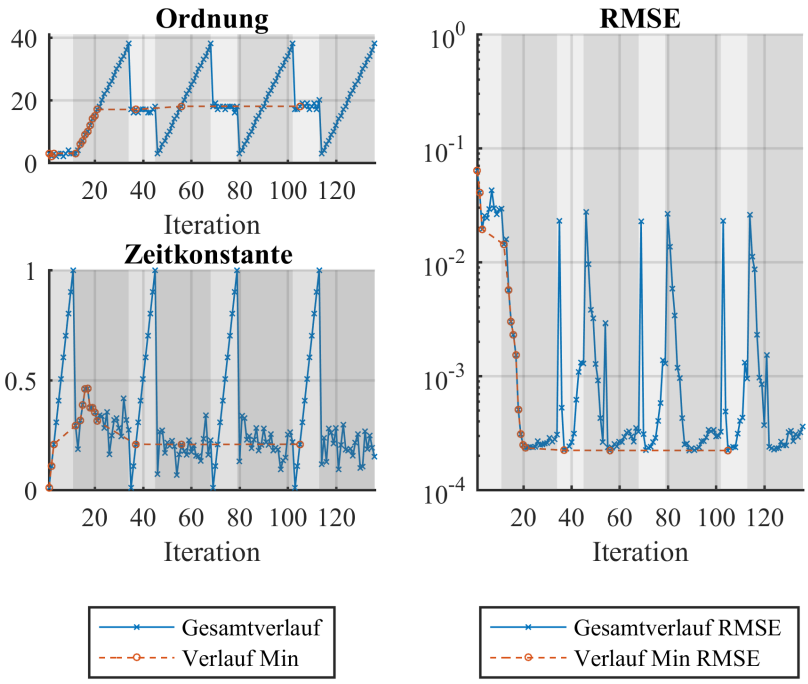


Abbildung 5.3: Verlauf der Suchparameter und des Minimums während der Liniensuche

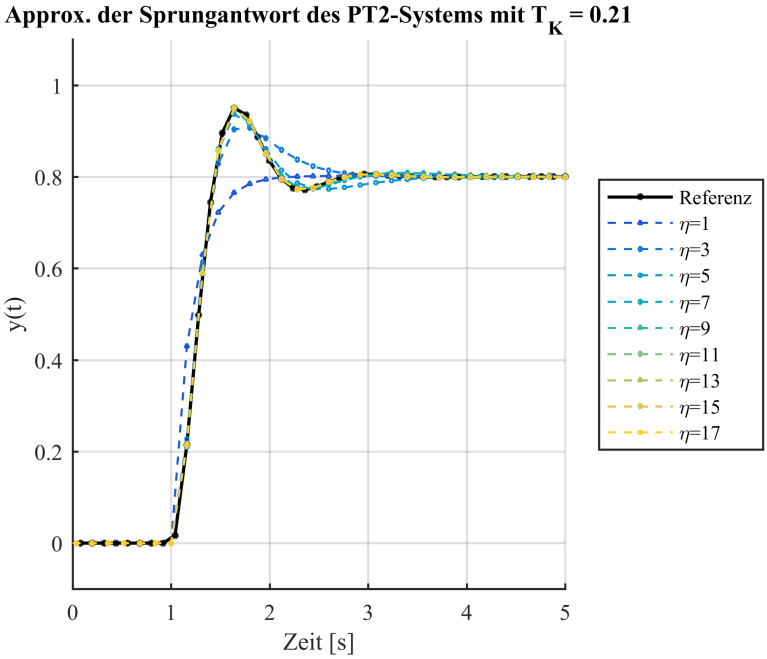


Abbildung 5.4: Modellierung eines stark gedämpften PT2-Systems mit Laguerre-Filtern

5.2 Neuronale Netze mit externen Laguerre-Filtern

Durch Kombination eines statischen neuronalen Netzes mit externen Laguerre-Filtern können nichtlineare dynamische Systeme modelliert werden. Dabei werden die gefilterten Eingangssignale in einem statischen neuronalen Netz in der Eingangsschicht verwendet. Dies führt zur einer nichtlinearen Überlagerung der gefilterten Eingangssignale. Um die Kombination statischer neuronaler Netze mit externen Laguerre-Filtern kompakter darstellen zu können, wird eine Kurzschreibweise eingeführt, ähnlich wie die zuvor eingeführte Schreibweise

für Zeitverzögerungsoperatoren. Dabei werden die gefilterten Signale eines Laguerre-Filters zu einem Vektor $\mathbf{u}_f$ zusammenfasst, der gegeben ist durch

$$\begin{aligned}\mathbf{u}_f(k) &= \Big(L(q,\alpha,1)u(k), \quad L(q,\alpha,2)u(k), \quad \dots \quad L(q,\alpha,\eta)u(k)\Big) \\ &= \Big(u_{f,1}(k), \quad u_{f,2}(k), \quad \dots \quad u_{f,\eta}(k)\Big)\end{aligned} \tag{5.4}$$

und in Abbildung 5.5 zu sehen ist. Zudem ist dort gezeigt, wie die einzelnen gefilterten Signale rekursiv berechnet werden können. Diese rekursive Berechnung eignet sich für eine Implementierung der erstellten neuronalen Netze als Simulationsmodell. Jedes Eingangssignal kann nun durch einen Laguerre-Filter gefiltert werden. Dazu kann für jedes Eingangssignal u_i ein eigener Pol α_i und eine eigene Ordnung η_i verwendet werden. Dabei entstehen die gefilterten Signale $\mathbf{u}_{f,1}$, $\mathbf{u}_{f,2}$, …, $\mathbf{u}_{f,\mathsf{p}}$.
Diese werden dann als Eingangssignale in einem neuronalen Netz verwendet, wodurch die nichtlineare Überlagerung entsteht:

$$\hat{y} = \mathsf{NN}(\mathbf{u}_{f,1}, \mathbf{u}_{f,2}, \dots, \mathbf{u}_{f,\mathsf{p}}, \mathbf{w}, \mathsf{P}_1, \mathsf{P}_2, \dots, \mathsf{P}_\mathsf{K}). \tag{5.5}$$

Die Hyperparameter sind in diesem Fall die Ordnungen und Pole der externen Laguerre-Filter und Parameter, die die Architektur des neuronalen Netzes beeinflussen. Durch die individuellen Pole und Ordnungen kann der unterschiedliche Einfluss der Eingangssignale auf das statische und das dynamische Verhalten berücksichtigt werden. Dadurch ist es möglich, eine hohe Modellgüte zu erhalten. Besonders durch individuelle Pole können unterschiedliche dynamische Einflüsse von Eingangssignalen auf das Systemverhalten berücksichtigt werden.

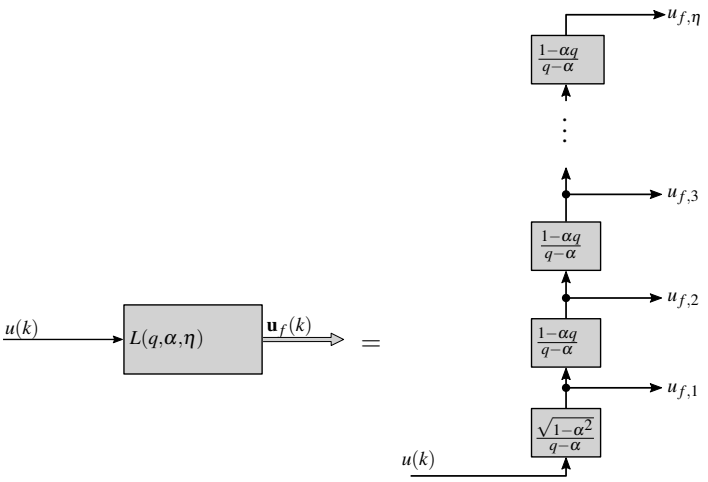


Abbildung 5.5: Kurzschreibweise für gefilterte Signale von Laguerre-Filtern

Zur Bestimmung der Hyperparameter gibt es folgende mögliche Ansätze:
Eine Möglichkeit zur Optimierung der Pole und Ordnungen der externen Laguerre-Filter ist die Verwendung globaler Optimierungsverfahren, wie zum Beispiel genetische Algorithmen. Diese Verfahren wurden in Kapitel 4 angerissen und sind näher in [Nel01; Sch10] beschrieben. Wie bereits in Kapitel 4 erwähnt, konvergieren diese sehr langsam, da die Pole und die Ordnungen alle gleichzeitig optimiert werden.
Eine weitere Möglichkeit ist es, die Pole und die Ordnungen als nichtlineare Modellparameter zu betrachten und diese durch ein lokales, nichtlineares Optimierungsverfahren zu verwenden [DW10]. Dies bedeutet jedoch Implementierungsaufwand, da nicht mehr auf Standard-Implementierungen zurückgegriffen werden kann. Da für eine automatische Modellerstellung in der Praxis ein weitestmöglicher Zugriff auf Standard-Implementierungen erwünscht ist, wird dieser Ansatz in dieser Arbeit nicht weiter verfolgt.

Im Folgenden wird näher beschrieben, wie zur Strukturoptimierung eine Liniensuche verwendet werden kann. Diese erlaubt zudem eine Implementierung, die mit wenig Aufwand verbunden ist. Bei dieser werden die Hyperparameter entlang definierter Linien im Suchraum optimiert. Der Ablauf wird weiter unten genauer beschrieben. Die Liniensuche ist dabei ein Kompromiss aus einfacher Implementierung und der Qualität des gefundenen Optimums, da es passieren kann, dass bei der Liniensuche ein lokales Minimum gefunden wird. In den meisten Fällen ergibt sich jedoch auf diese Weise ein zufriedenstellendes Ergebnis, das nicht wesentlich schlechter ist als im Fall einer globalen Optimierung. Für die Liniensuche, die zur Struktur- und Parameteroptimierung verwendet wird, werden Trainings-, Validierungs- und Trainingsdaten benötigt (siehe Kapitel 4). Im Folgenden wird zunächst auf Vorteile eingegangen, wenn dazu normierte Daten verwendet werden.

5.2.1 Normierung von Trainings-, Validierungs- und Testdaten

Für die Struktur- und Parameteroptimierung neuronaler Netze ist es vorteilhaft, die Eingangssignale auf einen einheitlichen Wertebereich zu normieren. Besonders bei der Verwendung eines gradientenbasierten Optimierungsverfahrens kann es passieren, dass bei zu starken Unterschieden bei den Größenordnungen der Eingangssignale manche Raumrichtungen kaum bis gar nicht ins Gewicht fallen. Dies führt dazu, dass die Bestimmung der Modellparameter erschwert oder sogar verfälscht wird [Nel01]. Aber auch für die Verwendung eines Clustering-Verfahrens zur Partitionierung des Eingangsraums ist es vorteilhaft, wenn dieser normiert ist. So können allgemeinere Aussagen über Einstellparameter getroffen werden. Da meist bei den Clustering-Verfahren eine euklidische Norm verwendet wird, wirkt sich auch hier eine Normierung vorteilhaft aus. Die Normierung der Signale wird dabei so durchgeführt, dass sie Werte zwischen null und eins liefert. Dies wird zum Beispiel erreicht durch folgende Transformation

$$u_n(k) = \frac{u(k) - \min(\mathbf{u})}{\max(\mathbf{u}) - \min(\mathbf{u})}, \tag{5.6}$$

wobei u_n das normierte Signal ist. Durch $\min(\mathbf{u})$ und $\max(\mathbf{u})$ wird der jeweils kleinste bzw. größte Wert des Signals aller Abtastpunkte gesucht und zur Normierung verwendet. Bei der Normierung gefilterter Signale wird der kleinste bzw. der größte Wert aller Ordnungen verwendet, so dass eine einheitliche Normierung der gefilterten Signale eines Eingangssignals durchgeführt wird. Bei dieser Vorschrift zur Normierung handelt es sich zudem um eine invertierbare Abbildung, so dass jederzeit die Signale aus dem normierten Raum in den originalen Raum zurücktransformiert werden können. Somit können beim Simulationsmodell die Ausgangssignale wieder ihre ursprünglichen Werte annehmen.

5.2.2 Liniensuche zur automatischen Struktur- und Parameteroptimierung

In diesem Abschnitt wird die Liniensuche mit stochastischen Elementen vorgestellt, die zur Struktur- und Parameteroptimierung neuronaler Netze mit externen Laguerre-Filtern verwendet wird. Ziel der Liniensuche ist es, für jeden Hyperparameter P_i des neuronalen Netzes mit externen Laguerre-Filtern den optimalen Wert BP_i zu finden. Als Kriterium, das hier minimiert werden soll, wird der RMSE_V der Validierungsdaten verwendet (siehe Kapitel 4). Am Ende werden diejenigen Werte BP_i der Hyperparameter herausgegeben, die zum niedrigsten Validierungsfehler geführt haben.
Für die Liniensuche müssen die Suchräume SP_1, SP_1, …, SP_m vorgegeben werden, in denen jeweils NP Werte enthalten sind. Dabei wird unterschieden, ob es sich um einen Suchraum mit ganzzahligen oder mit kontinuierlichen Werten handelt. Im Fall von kontinuierlichen Werten, wie zum Beispiel den Polen der Laguerre-Filter, werden Intervallgrenzen definiert. Diese sind durch eine untere

Grenze lb und eine obere Grenze ub gegeben. Anschließend wird vorgegeben, in wieviele gleichgroße Intervalle NP das gegebene Intervall eingeteilt werden soll. Somit ist ein Suchraum mit kontinuierlichen Werten gegeben durch

$$\mathsf{SP}_j = \text{linspace}\left(\text{lb}_j, \text{ub}_j, \mathsf{NP}_j\right). \tag{5.7}$$

Für ganzzahlige Hyperparameter, wie zum Beispiel die Ordnungen der Laguerre-Filter oder die Anzahl von Neuronen, wird der Suchraum entsprechend durch ganzzahlige Werte vorgegeben, die bei der Liniensuche berücksichtigt werden sollen. Dabei ist zu berücksichtigen, dass die Werte im Suchraum ihrer Größe nach sortiert werden. Ein Suchraum mit ganzzahligen Werten ist gegeben durch

$$\mathsf{SP}_k = \Big(\mathsf{SP}_k(1) \quad \mathsf{SP}_k(2) \quad \ldots \quad \mathsf{SP}_k(\mathsf{NP}_k)\Big). \tag{5.8}$$

Im Folgenden wird beschrieben, wie die so definierten Suchräume bei der Liniensuche verwendet werden. Der Ablauf der Liniensuche orientiert sich dabei an dem in Abbildung 5.6 dargestellten Ablaufdiagramm. Die Liniensuche startet mit der Optimierung des Hyperparameters P_1. Dazu wird der erste Wert aus dem Suchraum $\mathsf{SP}_1(1)$ verwendet. Anschließend werden für die restlichen Hyperparameter die bisher besten Werte angenommen und zusätzlich verrauscht. Die gewählten Hyperparameter werden dann im neuronalen Netz NN verwendet, und es werden die Modellparameter $\mathbf{w}$ durch ein Parameteroptimierungsverfahren unter Verwendung der Trainingsdaten ermittelt. Dies entspricht einem Iterationsschritt und ist im Ablaufdiagramm durch $\mathsf{Train}(\mathsf{NN}(\mathbf{w}, \mathsf{P}_1, \ldots, \mathsf{P}_\mathsf{m}))$ dargestellt.
Anschließend wird der Validierungsfehler RMSE_V bestimmt und mit dem bisher kleinsten Validierungsfehler ξ verglichen, und es wird folgende Fallunterscheidung durchgeführt:

- Gilt $\text{RMSE}_\mathsf{V} < \xi$, werden die verwendeten Werte für die Hyperparameter als neue beste Werte der Hyperparameter und der neu ermittelte kleinste

Validierungsfehler gespeichert. Anschließend wird mit dem nächsten Wert des Suchraums SP_1 fortgefahren.

- Gilt $\mathsf{RMSE}_\mathsf{V} \geq \xi$, wird direkt mit dem nächsten Wert des Suchraums fortgefahren.

Sind alle NP_1 Werte des Suchraums SP_1 verwendet worden, wird mit der Optimierung des zweiten Hyperparameters P_2 fortgefahren. Die Vorgehensweise ist dabei analog wie zuvor, wobei die Werte des Suchraums SP_2 verwendet werden. Ein Suchdurchlauf ist beendet, wenn die Optimierung des Hyperparameters P_m beendet ist.
Ist die maximale Anzahl von Suchdurchläufen erreicht, ist die gesamte Liniensuche beendet. Die Anzahl der durchzuführenden Iterationen $\mathsf{itr}_{\max}$ ist dabei gegeben durch

$$\mathsf{itr}_{\max} = \left(\sum_{k=1}^{\mathsf{m}} \mathsf{NP}_k \right) \cdot \mathsf{N}_\mathsf{O}. \tag{5.9}$$

Im Anhang A.1 wird gezeigt, dass die Verwendung stochastischer Elemente bei der Liniensuche Vorteile bringt, wenn die Kostenfunktion schräg im Raum liegt. Dies ist der Fall, wenn die Hyperparameter stark voneinander abhängen.
Zusammenfassend ergeben sich folgende Ergebnisse:

- Bei der Liniensuche ohne Verrauschen hat die Wahl der Werte der Suchräume einen starken Einfluss darauf, wie gute das reale Minimum gefunden wird. Dabei kann es passieren, dass nicht einmal die Nähe des Minimums erreicht wird. Bei Verwendung eines Rauschens ist die Liniensuche robuster bezüglich der Wahl der Suchräume.
- Mit Verrauschen kann zudem in manchen Fällen das Minimum nach weniger Iterationen gefunden werden als im Vergleich zum nicht verrauschten Fall. Dies ist davon abhängig, wie stark die Hyperparameter voneinander abhängen, und wie der Verlauf der Kostenfunktion ist.

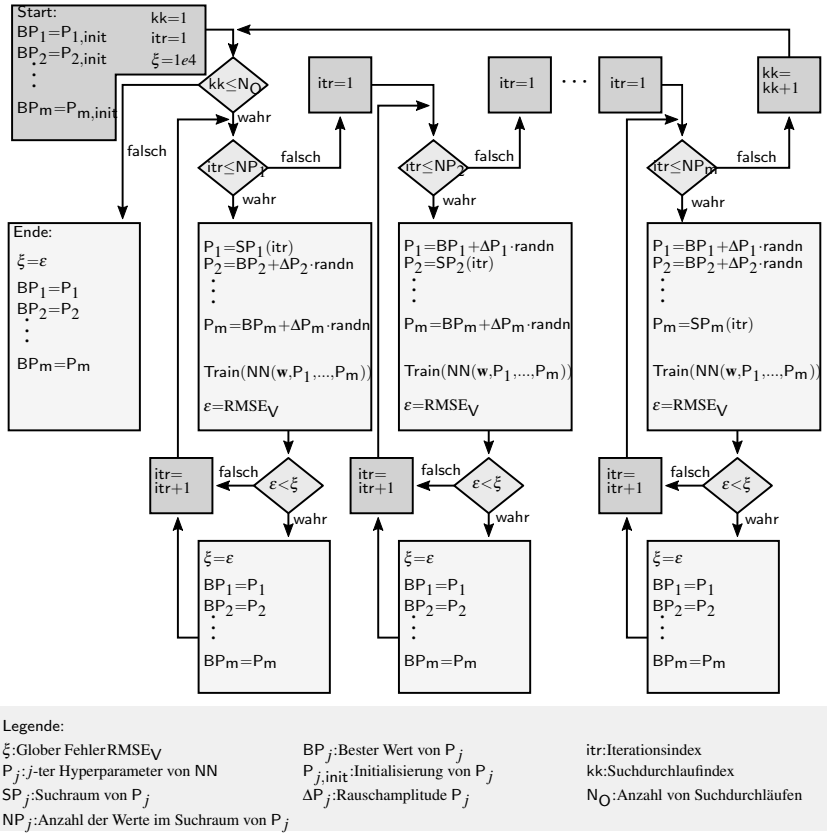


Abbildung 5.6: Ablaufdiagramm der Liniensuche

Weiter wird im Anhang diskutiert, wie stark das Rauschen bzw. wie groß die Amplitude gewählt werden sollte. Dabei ist das Ergebnis, dass sich ein zu starkes Rauschen wiederum negativ auf die Liniensuche auswirkt, vor allem wenn die Kostenfunktion nicht schräg im Raum liegt, also es kaum Abhängigkeiten zwischen den Hyperparametern gibt. Daher sollte die Rauschamplitude nicht zu stark gewählt werden. Das Ergebnis dieser Diskussion ist ein optimaler Wert

der Rauschamplitude von $\Delta P_2 = 0,075$ für das gaußsche Verrauschen kontinuierlicher Werte. Dabei bezieht sich dieser Wert darauf, dass die Grenzen des Suchraums bei null und eins sind. Bei anderen Grenzen wird die Rauschamplitude linear mit den Grenzen skaliert. Für ganzzahlige Werte in den Suchräumen beträgt die Rauschamplitude $\Delta P_1 = 0,75$ unabhängig von den Grenzen.

Die Liniensuche kann nun folgendermaßen zur Struktur- und Parameteroptimierung neuronaler Netze mit externen Laguerre-Filtern verwendet werden. Dazu wird die Optimierung der Hyperparameter folgendermaßen aufgeteilt, was gleichzeitig einen Suchdurchlauf der Liniensuche darstellt:

1. Optimierung der Hyperparameter der Modellarchitektur:
   Im ersten Schritt wird die Architektur des statischen neuronalen Netzes optimiert. Diese wird dabei durch verschiedene Hyperparameter beeinflusst. Die Frage, ob es sich um ganzzahlige oder kontinuierliche Hyperparameter handelt, hängt dabei ebenso vom gewählten Typ des neuronalen Netzes ab wie von deren Anzahl.
2. Optimierung der Ordnungen η_i:
   Im Allgemeinen kann für jedes Eingangssignal u_i eine eigene Ordnung η_i verwendet werden, so dass hier maximal p Hyperparameter vorgegeben werden können.
3. Optimierung der Zeitkonstanten $T_{K,i}$ $T_{K,i}$ bzw. der Pole α_i:
   Wie bei den Ordnungen kann auch hier für jedes Eingangssignal u_i eine eigene Zeitkonstante $T_{K,i}$ zur Bestimmung des Pols α_i verwendet werden.
4. Die Schritte zwei und drei werden für alle vorkommenden Ordnungen und Zeitkonstanten durchgeführt.

Für eine hohe Anzahl von Eingangssignalen steigt jedoch entsprechend die Anzahl zu optimierender Ordnungen und Polen und damit auch die Anzahl der Suchräume. Damit steigt auch der Aufwand der Liniensuche.
Bei den Untersuchungen der Liniensuche kam als Erfahrungswert heraus, dass

die Anzahl der Hyperparameter nicht größer als acht sein sollte, da sonst kein vernünftiges Minimum der Kostenfunktion gefunden wird. Daher werden im nächsten Abschnitt Möglichkeiten aufgezeigt, wie durch Untersuchung der Systemdynamik eine Reduktion der Anzahl von Polen bzw. Zeitkonstanten und Ordnungen erzielt werden kann.

5.2.3 Möglichkeiten zur Reduktion der Anzahl von Hyperparametern

Um ein beliebiges dynamisches MISO-System exakt durch ein neuronales Netz mit externen Laguerre-Filtern zu beschreiben, müssen theoretisch für jedes Eingangssignal die Ordnung und der Pol separat optimiert werden [BC85; Nel01]. In der Praxis können jedoch Eingangssignale zusammengefasst werden, für die dann ein gemeinsamer Pol und eine gemeinsame Ordnung verwendet werden, ohne dass eine zu starke Abnahme der Modellgüte zu beobachten ist [Sba97]. Dies kann dazu verwendet werden, die Anzahl der Hyperparameter zu reduzieren, um so die Dimension des Suchraums für die Liniensuche zur Strukturoptimierung zu verkleinern. Dazu werden folgende beide Möglichkeiten näher beschrieben:

- Aufteilung der Eingangssignale, je nach statischem oder dynamischem Einfluss auf das Systemverhalten.
- Verwendung eines gemeinsamen Pols und einer gemeinsamen Ordnung bei Eingangssignalen mit gleichem dynamischen Einfluss auf das Systemverhalten.

Die erste Möglichkeit, die Anzahl von Hyperparametern zu reduzieren, besteht darin, dass untersucht wird, ob die Eingangssignale überhaupt einen dynamischen Einfluss auf das Systemverhalten haben. Dazu wird betrachtet, ob eine Filterung der Eingangssignale sinnvoll erscheint, wobei der Verlauf der Eingangssignale mit dem Verlauf des Ausgangssignals verglichen wird. Wenn

bei diesem Vergleich auffällt, dass der Verlauf eines bestimmten Eingangssignals proportional zum Verlauf des Ausgangssignals ist, hat dieses Eingangssignal eine statische Wirkung. Die Wirkung der Filterung entspricht hier näherungsweise der eines Zeitverzögerungsoperators, und es werden zusätzliche, zeitversetzte Kopien dieses Eingangssignals im neuronalen Netz verwendet. Da die Laguerre-Filter dazu verwendet werden, um ein gedämpftes Verhalten zu modellieren, kann in diesem Fall auf eine Filterung verzichtet und dieses Eingangssignal als Signal mit statischem Einfluss verwendet werden.

Die zweite Möglichkeit zielt darauf ab, die Signale zu bestimmen, die durch einen gemeinsamen Pol und eine gemeinsame Ordnung beschrieben werden können. Dazu kann betrachtet werden, ob das Sprungverhalten bestimmter Signale durch eine gemeinsame Zeitkonstante beschrieben werden kann. Ist dabei die beobachtete Ordnung des Einschwingverhaltens auch gleich, kann für diese Signale eine gemeinsame Ordnung verwendet werden.
Außerdem kann es helfen, Signale mit gleichem dynamischen Einfluss zu finden, indem Wissen über die Wirkprinzipien des dynamischen Systems verwendet wird. Ein Beispiel dafür ist, dass die Größenordnung der Zeitkonstante des dynamischen Verhaltens bekannt ist. Ein weiteres Beispiel ist, dass bekannt ist, welche Subsysteme zusammenwirken, und welche Zeitkonstanten diese besitzen. Um die Liniensuche zu erleichtern, sollte so viel wie möglich vorliegendes Wissen genutzt werden. So können zudem die Intervalle der Suchräume eingeschränkt werden, die bei der Liniensuche verwendet werden.

Für die Modellerstellung kann diese Unterteilung folgendermaßen verwendet werden:
Eingangssignale mit statischem Einfluss werden mit $\mathbf{u}_s$ und Eingangssignale mit dynamischem Einfluss werden mit $\mathbf{u}_d$ gekennzeichnet. In Abbildung 5.7 ist zu sehen, wie die Einteilung in Signale mit statischem und dynamischem

Einfluss bei einem statischen neuronalen Netz mit externen Laguerre-Filtern verwendet werden kann.

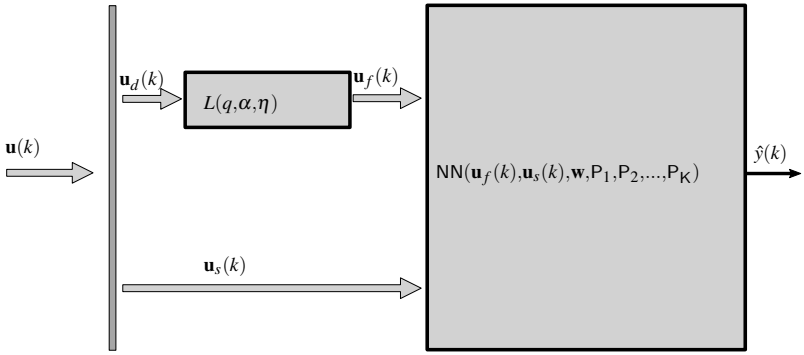


Abbildung 5.7: Neuronales Netz mit externen Laguerre-Filtern

Dabei wurden die gefilterten Signale zu einem Block zusammengefasst, wie es in Abbildung 5.8 zu sehen ist. Eingangssignale mit dynamischem Einfluss können dabei weiter zu m Gruppen zusammengefasst werden, bei denen zur Filterung ein Laguerre-Filter mit einheitlichem Pol und einheitlicher Ordnung verwendet wird. In manchen Fällen können auch alle Signale mit dynamischem Einfluss zu einer Gruppe zusammengefasst werden, so dass für alle Eingangssignale ein einheitlicher Pol und eine einheitliche Ordnung verwendet werden.

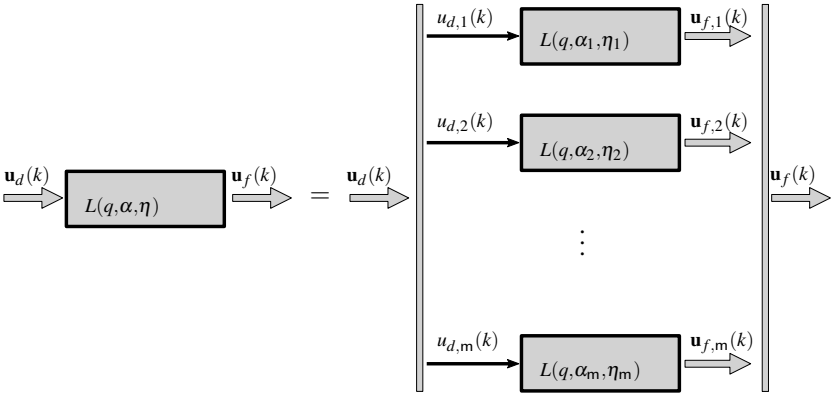


Abbildung 5.8: Kurzschreibweise für Filterung von Eingangssignalen mit dynamischem Einfluss durch Laguerre-Filter

5.2.4 Ausdünnung von Trainingsdaten

Die oben beschriebene Möglichkeit zur Ausdünnung der gefilterten Signale kann auch bei der Erstellung neuronaler Netze verwendet werden, um eine Reduktion des Rechenaufwands bei der Parameteroptimierung zu erreichen. Dabei muss auch hier wieder überprüft werden, ob durch die Ausdünnung kein Informationsverlust stattfindet. Dazu ist die Frage zu klären, ob die oben genannte Voraussetzung einer sehr feinen Abtastung der gefilterten Signale erfüllt ist. Da die erstellten Modelle in einer Simulation verwendet werden, ist meist die Voraussetzung erfüllt, dass die gefilterten Signale deutlich höher abgetastet sind, als es benötigt wird.
Vor allem bei Optimierungsverfahren, bei denen die Jacobi-Matrix berechnet wird (siehe Kapitel 4), bringt die Ausdünnung einen deutlichen Zeitgewinn bei der Parameteroptimierung. Da die Parameteroptimierung bei der Liniensuche in jedem Iterationsschritt durchgeführt wird, wird die Dauer der Liniensuche stark verkürzt.

Weiter wirken sich die glättenden und generalisierenden Eigenschaften neuronaler Netze [Nel01] vorteilhaft bei der Modellierung des statischen nichtlinearen Teils des Modells aus, so dass in einer Simulation die ursprüngliche, feinere Abtastzeit T_S verwendet werden kann.

5.3 Schritte der automatischen Erstellung neuronaler Netze

Im Folgenden wird gezeigt, wie die Liniensuche verwendet werden kann, um neuronale Netze mit externen Laguerre-Filtern automatisch zu erstellen. Die Erstellung neuronaler Netze läuft nach folgenden drei Schritten ab:

1. Vorgabe und Aufteilung von Daten:
   Vorgabe von Daten, die mit fester Abtastzeit T_S in einer Simulation bzw. in einem Experiment aufgezeichnet wurden. Weiter findet eine Aufteilung der Daten in einen Teil zur Parameteroptimierung (die Trainingsdaten) und in einen weiteren Teil zur Validierung (die Validierungsdaten) statt. Zusätzlich werden unabhängige Testdaten zur abschließenden Bewertung der Modellgüte verwendet. Die Daten werden wie oben beschrieben normiert. Für gefilterte Signale werden dabei das Maximum und das Minimum der gefilterten Signale eines Eingangssignals verwendet.
2. Reduktion der Anzahl von Hyperparametern:
   Um den Aufwand der Liniensuche zu begrenzen, wird analysiert, welche der Eingangssignale einen statischen bzw. dynamischen Einfluss haben. Weiter wird untersucht, ob für bestimmte Eingangssignale eine einheitliche Ordnung und Zeitkonstante in den Laguerre-Filtern verwendet werden kann. Dabei entstehen m Zeitkonstanten und m Ordnungen, die bei der Liniensuche ermittelt werden müssen.

3. Liniensuche zur Struktur- und Parameteroptimierung:
   Dazu werden vorgegebene Suchräume für die entsprechenden Hyperparameter verwendet. Zusätzlich wird die Anzahl N_O der Suchdurchläufe vorgegeben. Am Ende wird das Modell mit dem kleinsten Validierungsfehler ausgegeben. Zusätzlich wird der Testfehler aufgezeichnet, allerdings hat dieser keinen Einfluss auf die Wahl der Hyperparameter.
   Bei der Liniensuche werden zunächst die Hyperparameter der Architektur, dann die der Ordnungen und zuletzt die der Zeitkonstanten optimiert, wie es oben beschrieben wurde.
   Ist die Abtastzeit T_S höher, als nach Anwendung der Laguerre-Filter benötigt wird, kann die Parameteroptimierung durch Ausdünnung der Trainingsdaten beschleunigt werden. Dazu werden die Trainingsdaten auf eine gröbere Abtastzeit umgerechnet, und anschließend wird das Verfahren zur Parameteroptimierung gestartet.

Für die Liniensuche werden Suchräume benötigt, um die Ordnungen und die Zeitkonstanten der Laguerre-Filter zu optimieren. Außerdem werden r Suchräume zur Optimierung der Architektur des neuronalen Netzes benötigt. Dabei wird davon ausgegangen, dass es insgesamt r Hyperparameter der Modellarchitektur und m jeweils für Ordnungen bzw. Zeitkonstanten der Laguerre-Filter. Somit ergeben sich insgesamt $k = r + m$ Suchräume. In dieser Arbeit werden LM-Netze und MLP-Netze verwendet, wobei sich unterschiedliche Suchräume ergeben. Unabhängig vom neuronalen Netz können jedoch die Suchräume zur Optimierung der Parameter der Laguerre-Filter vorgegeben werden. Daher wird zunächst auf diese Suchräume näher eingegangen.
Für die Ordnungen η_i werden m Suchbereiche für die Signale mit dynamischem Einfluss vorgegeben, die aus der Analyse und der beobachteten Ordnung des Einschwingverhaltens der Sprungantwort erfolgt sind.

Dabei ergeben sich folgende Suchbereiche mit ganzzahligen Werten

$$\begin{aligned}
\mathsf{SP}_{\mathsf{r}+1} &= \begin{pmatrix} \eta_{1,1} & \eta_{1,2} & \dots & \eta_{1,\mathsf{NP}_{\mathsf{r}+1}} \end{pmatrix} \\
\mathsf{SP}_{\mathsf{r}+2} &= \begin{pmatrix} \eta_{2,1} & \eta_{2,2} & \dots & \eta_{2,\mathsf{NP}_{\mathsf{r}+2}} \end{pmatrix} \\
&\vdots \\
\mathsf{SP}_{\mathsf{r}+\mathsf{m}} &= \begin{pmatrix} \eta_{\mathsf{m},1} & \eta_{\mathsf{m},2} & \dots & \eta_{\mathsf{m},\mathsf{NP}_{\mathsf{r}+\mathsf{m}}} \end{pmatrix}.
\end{aligned} \tag{5.10}$$

Dabei kann bei jedem Suchraum eine unterschiedliche Anzahl NP_j von Ordnungen verwendet werden. Die Werte sind dabei ihrer Größe nach sortiert. Üblicherweise wird mit der Ordnung 0 gestartet, und dann wird in ganzzahligen Schritten jede Ordnung bis zur maximalen Ordnung im Suchraum verwendet. Ist die maximale Ordnung jedoch sehr hoch, können auch Zwischenwerte ausgelassen werden. Da es sich bei den Ordnungen um ganzzahlige Werte handelt, wird hier als Rauschamplitude $\Delta\mathsf{P} = 0,75$ verwendet.

Zur Optimierung der Pole α_j bzw. Zeitkonstanten $T_{\mathrm{K,j}}$ werden m Suchbereiche vorgegeben. Dabei werden die Größenordnungen für die Zeitkonstanten verwendet, die aus der Analyse des dynamischen Einflusses der Signale gewonnen werden. Diese werden dann in die entsprechenden Pole umgerechnet, wie oben beschrieben wurde. Für die Zeitkonstanten ergeben sich folgende Suchbereiche:

$$\begin{aligned}
\mathsf{SP}_{\mathsf{r}+\mathsf{m}+1} &= \mathrm{linspace}\left(\min(T_{\mathrm{K},1}), \max(T_{\mathrm{K},1}), \mathsf{NP}_{\mathsf{m}+1+\mathsf{r}}\right) \\
\mathsf{SP}_{\mathsf{r}+\mathsf{m}+2} &= \mathrm{linspace}\left(\min(T_{\mathrm{K},2}), \max(T_{\mathrm{K},2}), \mathsf{NP}_{\mathsf{m}+2+\mathsf{r}}\right) \\
&\vdots \\
\mathsf{SP}_{\mathsf{r}+2\cdot\mathsf{m}} &= \mathrm{linspace}\left(\min(T_{\mathrm{K},\mathsf{m}}), \max(T_{\mathrm{K},\mathsf{m}}), \mathsf{NP}_{2\cdot\mathsf{m}+\mathsf{r}}\right)
\end{aligned} \tag{5.11}$$

Pro Suchraum kann wieder eine unterschiedliche Anzahl von zu untersuchenden Zeitkonstanten gewählt werden. Da es sich um kontinuierliche Werte handelt, wird als Rauschamplitude $\Delta P = 0,075$, jeweils mit der kleinsten und größten Zeitkonstanten des Suchbereichs skaliert, verwendet.
Im Folgenden wird näher auf die Unterschiede bei der Liniensuche eingegangen, die sich durch Verwendung von LM-Netzen bzw. MLP-Netzen ergeben.

5.3.1 Ablauf der Liniensuche bei LM-Netzen

In diesem Abschnitt wird auf die Hyperparameter der Netzarchitektur eingegangen, die bei der Liniensuche optimiert werden. Eine Besonderheit der LM-Netze stellt dabei die Unterteilung in einen **x**-Regressor und einen **z**-Regressor dar, die folgendermaßen verwendet werden kann:

- Es kann unterschieden werden, ob ein Signal nichtlinear oder linear das Systemverhalten beeinflusst. Im letzteren Fall kann ein entsprechendes Signal im **z**-Regressor weggelassen werden. So kann entschieden werden, ob ein bestimmtes Signal im jeweiligen Regressor verwendet werden soll oder nicht.
- Prinzipiell können für die Filterung der Signale, die in den beiden Regressoren verwendet werden, unterschiedliche Ordnungen und Zeitkonstanten vorgegeben werden. Allerdings wird davon ausgegangen, dass der dynamische Einfluss auf das Systemverhalten nicht davon abhängt, ob dieser im **z**-Regressor oder im **x**-Regressor verwendet wird. Deshalb wird für die Signale die gleiche Zeitkonstante bzw. der gleiche Pol verwendet, egal in welchem Regressor die gefilterten Signale verwendet werden.
- Um die Dimension für die Partitionierung zu verringern, werden zur Filterung der Signale im **z**-Regressor geringere Ordnungen η_i verwendet. Besonders bei vielen Eingangssignalen ist dies hilfreich. Um nicht zusätzliche Hyperparameter für die Liniensuche zu erzeugen, wird für den

x-Regressor und den **z**-Regressor die jeweils gleiche Ordnung verwendet, jedoch wird für die Verwendung eine Obergrenze für die Ordnung im **z**-Regressor vorgegeben.

In Abbildung 5.9 ist der Aufbau der vorgestellten Struktur eines statischen lokalen Modellnetzes mit externen Laguerre-Filtern zu sehen.

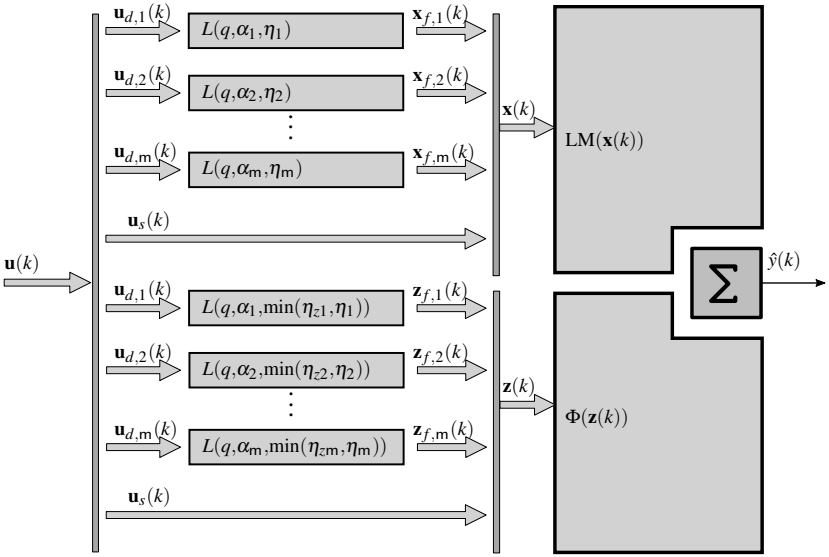


Abbildung 5.9: Lokales Modellnetz mit externen Laguerre-Filtern mit Gruppierung der Signale nach dynamischem und statischem Einfluss

Dabei ist gezeigt, dass für die Eingangssignale $\mathbf{u}$ m Gruppen gebildet werden können, die einen dynamischen Einfluss auf das Systemverhalten haben. Für diese werden dann jeweils der gleiche Pol und die gleiche Ordnung verwendet. Hinzu kommen die Eingangssignale $\mathbf{u}_s$ mit statischem Einfluss. Um den Raum für die Partitionierung nicht zu stark anwachsen zu lassen, können die maximalen Ordnungen η_{zj} für die jeweiligen gefilterten Signale im

z-Regressor vorgegeben werden. Das bedeutet, dass in den lokalen Modellen eine höhere Ordnung verwendet werden kann. Zur Partitionierung und Bestimmung der Parameter der lokalen Modelle wird ein angepasstes inkrementelles Clustering-Verfahren verwendet. Wesentliches Ziel der Anpassung ist dabei, dass pro Iterationsschritt mehrere lokale Modelle hinzugefügt werden können. Besonders bei hochdimensionalen Eingangsräumen ist das von Vorteil, da beim Hinzufügen eines einzelnen lokalen Modells selten eine signifikante Verbesserung erzielt werden kann.
Der genaue Ablauf ist in Anhang A.2.2 beschrieben, und es werden folgende Vorgaben benötigt:

- Vorgaben eines kleinsten Abstands r_{min} und eines Skalierungsfaktors σ:
  Durch den kleinsten Abstand wird vorgegeben, wie nah sich zwei Zentren von Gaußfunktionen maximal kommen dürfen, und durch die Skalierung kann die Schärfe der Übergänge zwischen den lokalen Modellen bestimmt werden. Diese beiden Parameter gehen jedoch nichtlinear in die Optimierung ein, was bei der Modellerstellung berücksichtigt werden muss.
- Vorgabe der Anzahl von Iterationen:
  Durch die Anzahl der Iterationen kann der Aufwand bei der Modellerstellung skaliert werden. Im extremsten Fall würde pro Iteration ein lokales Modell hinzugefügt. Dieser Fall bedeutet auch den maximalen Rechenaufwand. Es hat sich gezeigt, dass mit einer wesentlich geringeren Anzahl von Iterationen bereits gute Ergebnisse erzielt werden. Dies bedeutet, dass pro Iteration mehrere lokale Modelle hinzugefügt werden. Es wird empfohlen, 12 Iterationen zu verwenden. Diese Anzahl von Iterationen stellt einen Kompromiss zwischen der Optimierung der notwendigen Anzahl lokaler Modellen, der Güte der Partitionierung des Eingangsraums und dem zeitlichen Aufwand zur Erstellung dar.

- Vorgabe der Anzahl maximaler Modellparameter:
  Durch Vorgabe der maximalen Modellparameter wird ermittelt, wie viele Zentren pro Iterationsschritt hinzugefügt werden dürfen. Die Anzahl der Modellparameter eines lokalen Modellnetzes wird dabei durch die Anzahl M der lokalen Modelle und die Anzahl der Signale im $\mathbf{x}$-Regressor bestimmt und ist gegeben durch

$$n_{\mathsf{w}} = (1 + \mathsf{px}) \cdot \mathsf{M}. \tag{5.12}$$

  Die Entscheidung über die Anzahl der Modellparameter orientiert sich an der Komplexität der gestellten Modellierungsaufgabe. Dabei ist zu beachten: je höher die Anzahl der vorgegebenen Modellparameter ist, desto mehr Daten werden für die Modellerstellung und Modellvalidierung benötigt.

Das beschriebene inkrementelle Clustering-Verfahren wird im Schritt $\mathsf{Train}(\cdot)$ der automatisierten Struktur- und Parameteroptimierung lokaler Modellnetze verwendet, wie es in Abbildung 5.6 dargestellt ist. Im Folgenden wird auf die Suchräume der einzelnen Phasen der Liniensuche eingegangen.
Die Optimierung der Modellarchitektur erfolgt durch das oben beschriebene inkrementelle Clustering-Verfahren. Da dieses beim Hinzufügen lokaler Modelle den Validierungsfehler mit berücksichtigt, findet dabei eine Optimierung der Anzahl von Modellparametern statt, wodurch die Gefahr einer Überanpassung verringert wird. Das Ergebnis wird dabei von folgenden zwei Hyperparametern ($\mathsf{r} = 2$) beeinflusst:

- r_{min}: Kleinster, zulässiger Abstand zwischen zwei Zentren in den Aktivierungsfunktionen
- σ: Einheitlich verwendete Skalierung in den Aktivierungsfunktionen

Für diese beiden Parameter werden daher Suchräume für die Liniensuche definiert:

$$\begin{aligned} \mathsf{SP}_1 &= \text{linspace}\,(\min(r_{\mathsf{min}}), \max(r_{\mathsf{min}}), \mathsf{NP}_1) \\ \mathsf{SP}_2 &= \text{linspace}\,(\min(\sigma), \max(\sigma), \mathsf{NP}_2) \end{aligned} \tag{5.13}$$

Dabei liegen die Werte für eine optimale Wahl von r_{min} zwischen $0,05$ und $0,7$, wenn ein normierter Eingangsraum verwendet wird. Die Werte für eine optimale Wahl von σ liegen zwischen $0,1$ und $0,5$. Untersuchungen des Clustering-Verfahrens haben ergeben, dass jeweils eine Teilung des Suchraums in 7 bis 8 Werte ausreicht, um optimale Werte zu finden. Da es sich um kontinuierliche Werte handelt, wird als Rauschamplitude $\Delta\mathsf{P} = 0,075$ verwendet, linear skaliert mit den jeweiligen Grenzen der Suchräume, wobei als Referenz Grenzen zwischen null und eins verwendet werden (siehe Anhang A.1).

5.3.2 Ablauf der Liniensuche bei MLP-Netzen

In diesem Abschnitt wird auf die Struktur- und Parameteroptimierung statischer MLP-Netze mit externen Laguerre-Filtern eingegangen. Hier werden in der Eingangsschicht des MLP-Netzes die durch Laguerre-Filter gefilterten Signale verwendet (siehe Abbildung 5.10).
Dabei wurden die Eingangssignale wieder unterteilt, je nach dynamischem oder statischem Einfluss. Weiter können m Gruppen von Signalen mit dynamischem Einfluss gebildet werden, für die jeweils der gleiche Pol und die gleiche Ordnung verwendet werden. Wie bei den LM-Netzen werden diese Hyperparameter durch die Liniensuche bestimmt.
Die Eigenschaften der Architektur des MLP-Netzes sind dabei durch die Anzahl n von verdeckten Schichten und die Anzahl M von Basisfunktionen bzw. Neuronen pro Schicht bestimmt.

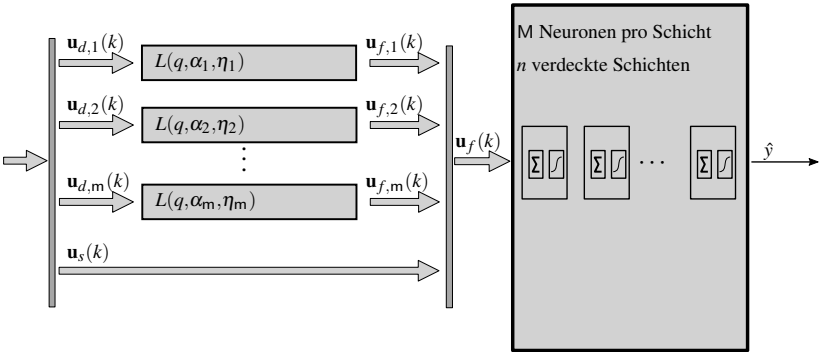


Abbildung 5.10: Statisches MLP-Netz mit mehreren verdeckten Schichten und externen Laguerre-Filtern und Gruppierung der Signale nach dynamischem und statischem Einfluss

Durch eine zweite verdeckte Schicht kann zwar das MLP-Netz besser nichtlineare Zusammenhänge erfassen, allerdings steigt so auch die Gefahr der Überanpassung. Zudem ist die Verwendung mehrerer verdeckter Schichten eher sinnvoll, wenn das MLP-Netz für Klassifikationsaufgaben verwendet werden soll, bei denen die Klassen nicht linear separierbar sind [Hay09]. Für die Modellierung dynamischer Systeme reicht in den meisten Fällen die Verwendung einer verdeckten Schicht, wobei die Optimierung der Werte für die Pole und Ordnungen häufig eine wichtigere Rolle spielt. Ein weiteres Argument gegen die Verwendung einer zweiten verdeckten Schicht (oder mehr) ist eine weitere Zunahme des Aufwands der Parameteroptimierung.

Wie bei der Verwendung lokaler Modellnetze orientiert sich die Vorgabe der Anzahl der Neuronen an der Komplexität der gestellten Modellierungsaufgabe und der Menge an zu Grunde liegenden Daten. Letztere Einschränkung kann durch die gezielte Erzeugung von Simulationsdaten entschärft werden. Bei einer verdeckten Schicht ist die Anzahl der Modellparameter gegeben durch $n_\mathsf{w} = \mathsf{p} \cdot (\mathsf{M}+1) + \mathsf{M} + 1$.

Zur Parameteroptimierung des MLP-Netzes, das bei fest vorgegebenen Hyperparametern bei der Liniensuche zum Einsatz kommt, wird das Levenberg-Marquardt-Verfahren verwendet, da dieses sehr gute Konvergenzeigenschaften besitzt [Nel01]. Da dieses Verfahren relativ aufwändig zu implementieren ist, empfiehlt es sich, auf eine Implementierung zurückzugreifen, wie beispielsweise die Neural Network Toolbox von Mathworks [Mat].
Im Folgenden wird auf die Phasen und die Suchräume zur Optimierung der Modellarchitektur eingegangen, die bei der Liniensuche verwendet werden. Die Suchräume der Ordnungen und Pole sind identisch zu denen bei den lokalen Modellnetzen.
Die Optimierung der Modellarchitektur ist identisch mit der Optimierung der Anzahl M der Neuronen in der verdeckten Schicht. Somit ist der Suchraum ($r = 1$) gegeben durch:

$$SP_1 = \begin{pmatrix} M_{1,1} & M_{2,2} & \dots & M_{1,NP_1} \end{pmatrix} \tag{5.14}$$

In der Praxis zeigt sich, dass typischerweise zwischen zehn und 30 Neuronen ausreichen, um eine gute Modellgüte zu erhalten. Das bedeutet, dass bei der Erstellung von MLP-Netzen ein Hyperparameter weniger optimiert werden muss als bei den LM-Netzen. Dafür ist jedoch der Rechenaufwand des Levenberg-Marquardt-Verfahrens größer als der des oben verwendeten inkrementellen Clustering-Verfahrens zur Erstellung von LM-Netzen.

5.4 Anwendungsbeispiel zur numerischen Vereinfachung

Zum Ende dieses Kapitel wird die Liniensuche zur automatischen Erstellung neuronaler Netze mit externen Laguerre-Filtern verwendet, um ein stark gedämpftes, nichtlineares dynamisches MISO-System zu modellieren. Dabei wird auch gezeigt, wie durch die Verwendung der erstellten neuronalen Netze eine

numerische Vereinfachung erzielt wird. Das in diesem Beispiel verwendete dynamische System besitzt zwei Eingangssignale und ein Ausgangssignal und ist in Abbildung 5.11 zu sehen.

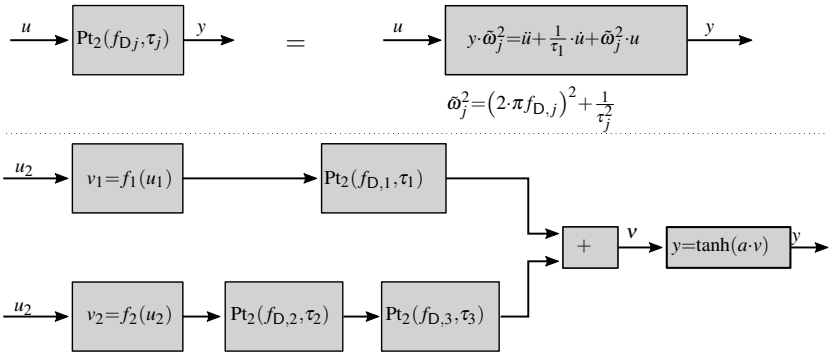


Abbildung 5.11: Nichtlineares dynamisches System mit zwei Eingangssignalen und einem Ausgangssignal

Das gezeigte System besitzt statische Nichtlinearitäten am Eingang und ein statisches Sättigungsverhalten am Modellausgang. Die Dynamik wird durch lineare dynamische Systeme zweiter Ordnung bestimmt, welche in Abbildung mit PT_2 gekennzeichnet gekennzeichnet sind. Für jedes dieser dynamischen Systeme können eine eigene Schwingungsfrequenz $f_{D,j}$ und eine Dämpfungszeit τ_j vorgegeben werden. Die Parameter $f_{D,2}$ und τ_2 sollen dabei so gewählt werden, dass ein steifes Differentialgleichungssystem entsteht. Dazu wird für diese eine deutlich höhere Schwingungsfrequenz und Dämpfungszeit gewählt als in den beiden anderen. Die Auswirkungen auf das dynamische Verhalten am Modellausgang sind dabei gering. Auf diese Weise kann demonstriert werden, wie eine numerische Vereinfachung erzielt werden kann.

Im Folgenden wird mit beispielhaften Modellparameter gezeigt, wie die neuronalen Netze zur numerischen Vereinfachung erstellt wurden, und welche

numerische Vereinfachung sich dadurch erzielen ließ. Im ersten Schritt wurden Simulationsdaten erzeugt. Dazu wurden folgende Modellparameter für die statischen nichtlinearen Anteile des Modells in Abbildung 5.11 vorgegeben:

$$\begin{aligned} f_1(u_1) &= 0,1 \cdot u_1 + 0,9 \cdot u_1^2 \\ f_2(u_2) &= -0,1 \cdot u_2 - 0,9 \cdot u_2^2 \\ a &= 3 \end{aligned} \qquad . \tag{5.15}$$

Für die dynamischen Anteile wurden folgende Modellparameter gewählt:

$$\begin{array}{llllll} \tau_1 = & 0,1[\mathrm{s}] & \tau_2 = & 0,003[\mathrm{s}] & \tau_3 = & 0,1[\mathrm{s}] \\ f_{\mathrm{D},1} = & 0,4[\mathrm{Hz}] & f_{\mathrm{D},2} = & 50[\mathrm{Hz}] & f_{\mathrm{D},3} = & 0,2[\mathrm{Hz}] \end{array} \tag{5.16}$$

Zur Erzeugung von Simulationsdaten, die für die Modellerstellung verwendet wurden, wurden folgende Vorgaben gemacht:
Die simulierte Zeitdauer betrug 2500[s]. Dafür wurden 3[min] Rechenzeit benötigt. Dabei wurden während der ersten 1750[s] als Eingangssignale Sprungsignale mit zufälligen Amplituden zwischen 0 und 1 pro Sprungsequenz gewählt. Die Haltedauer der Sprungsignale wird dabei je Signal zufällig zwischen $0,5$[s] und $1,5$[s] pro Sprungsequenz vorgegeben.
Nach 1750[s] wurde auf synchrone Sprungsequenzen umschaltet. Diese hatten dabei eine feste Haltedauer von 1[s]. Jedoch wurden pro Eingangssignal weiterhin die Amplituden zwischen 0 und 1 zufällig vorgegeben. Der resultierende Verlauf der Eingangssignale und des Ausgangssignals ist ausschnittsweise in Abbildung 5.12 zu sehen.
Die Abtastzeit der erstellten Daten wurde mit $T_S = 0,01$[s] vorgegeben, so dass das Ausgangssignal zu fein abgetastet war. Somit konnte die Parameteroptimierung mit einem ausgedünnten Satz der Trainingsdaten durchgeführt werden, wie es oben beschrieben ist. Für die Parameteroptimierung wurde dabei eine Abtast-

zeit von $T_S = 0,04[s]$ verwendet. Daraus folgte, dass die kleinste Zeitkonstante für die Laguerre-Filter $T_K = 0,08[s]$ betragen darf.

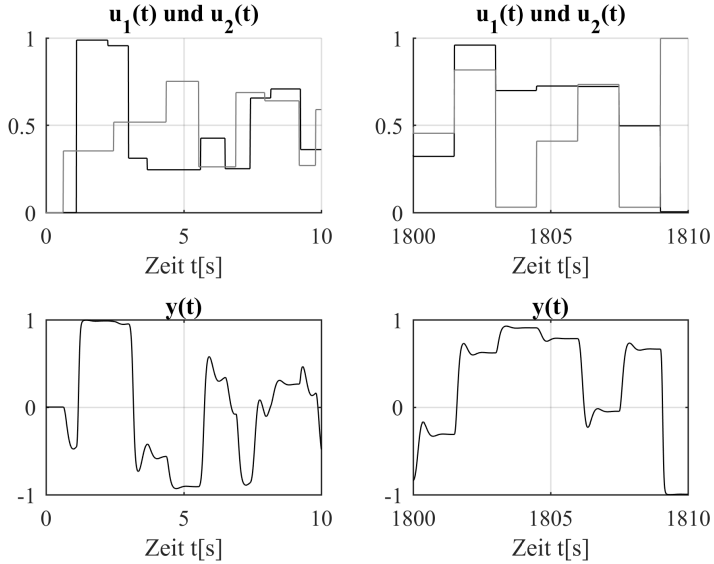


Abbildung 5.12: Verlauf der Eingangssignale und des Ausgangssignals des nichtlinearen dynamischen Systems

Für die automatische Modellerstellung wurde die oben beschriebene Aufteilung der Daten durchgeführt. Die ersten 85% wurden für die Struktur- und Parameteroptimierung verwendet. Diese wurden weiter aufgeteilt in 90% für die Parameteroptimierung und in 10% für die Validierung während der Strukturoptimierung. Die restlichen 15% des gesamten Datensatzes wurden zum Testen des erstellen neuronalen Netzes verwendet. Für die Liniensuche wurden für die beiden Eingangssignale jeweils eigene Ordnungen und Zeitkonstanten gewählt und daher folgende Suchräume definiert:

$$
\begin{aligned}
\mathsf{SP}_{\eta_1} &= \begin{pmatrix} 0 & 2 & 4 & 6 & 8 & 10 \end{pmatrix} \\
\mathsf{SP}_{\eta_2} &= \begin{pmatrix} 0 & 2 & 4 & 6 & 8 & 10 \end{pmatrix} \\
\mathsf{SP}_{T_{\mathrm{K},1}} &= \mathrm{linspace}\left(\frac{1}{20} , \frac{1}{2} , 10 \right) \\
\mathsf{SP}_{T_{\mathrm{K},2}} &= \mathrm{linspace}\left(\frac{1}{20} , \frac{1}{2} , 10 \right) .
\end{aligned}
\tag{5.17}
$$

Für die Liniensuche wurden fünf Suchdurchläufe $\mathsf{N_O} = 5$ vorgegeben. Die Suchräume zur Optimierung der Architekturparameter im Falle eines LM-Netzes bzw. eines MLP-Netzes werden im Folgenden jeweils angegeben. Als Anfangswerte für die Liniensuche werden dabei die Mittelwerte der jeweiligen Suchräume der Ordnungen und Zeitkonstanten verwendet.

5.4.1 Modellierung mit einem LM-Netz

Für die Modellierung des nichtlinearen System aus Abbildung 5.11 durch ein LM-Netz wurden neben den oben genannten Suchräumen noch zusätzlich folgende Suchräume verwendet, um die optimalen Parameter des inkrementelle Clustering-Verfahrens zu bestimmen. Die Suchräume waren dabei gegeben durch

$$
\begin{aligned}
\mathsf{SP}_{r_{\mathrm{min}}} &= \mathrm{linspace}\left(\frac{3}{40} , \frac{3}{10} , 8 \right) \\
\mathsf{SP}_{\sigma} &= \mathrm{linspace}\left(\frac{1}{40} , \frac{3}{10} , 8 \right) .
\end{aligned}
\tag{5.18}
$$

Somit mussten in diesem Fall unter Berücksichtigung der Hyperparameter der Laguerre-Filter durch die Liniensuche 6 Hyperparameter optimiert werden. Als Startwerte der Liniensuche wurden hier jeweils die kleinsten Werte der Suchräume verwendet, wie es oben bei der Erstellung von LM-Netzen beschrieben

ist. Damit die Dimension zur Partitionierung nicht zu groß wurde, wurde im **x**-Regressor nur eine maximale Ordnung von $\eta_{z1} = 2$ und von $\eta_{z2} = 2$ zugelassen. Dabei wurden als maximal zulässige Anzahl von Modellparametern $n_w = 1000$ vorgegeben [2]. In Abbildung 5.13 ist der Verlauf der RMSE-Fehler, der mit den Trainings-, den Validierungs- und den Testdaten während der Liniensuche in den jeweiligen Schritten erhalten wurde, zu sehen.

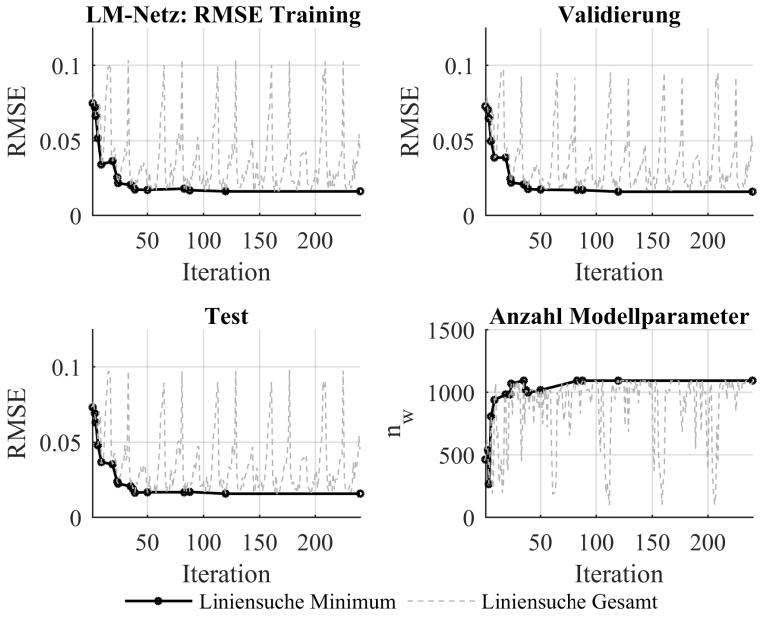


Abbildung 5.13: Verlauf der Modellgüte während der Liniensuche bei Verwendung eines LM-Netzes

[2] Bei der Erstellung der LM-Netze wurde die maximale Anzahl von Modellparametern zwischen 600 und 3600 variiert. Dabei wurde festgestellt, dass die Modellgüte nicht wesentlich besser wurde bei einer höheren Anzahl von Modellparametern. Bei 3600 schließlich wurde wieder eine Verschlechterung beobachtet, die auf eine Überanpassung hindeutete.

Zudem ist der Verlauf der Anzahl der Modellparameter eingezeichnet. Bei allen Verläufen ist der Gesamtverlauf der Liniensuche und der Verlauf des neuen ermittelten Minimums das RMSE-Fehlers dargestellt, der mit den Validierungsdaten erhalten wurde. Dabei ist zu erkennen, dass nach etwa 50 Iterationen die Größenordnung des kleinsten Validierungsfehler gefunden wurde und in den folgenden Iterationen der Liniensuche keine wesentlichen Verbesserungen erzielt wurden. Zudem wird klar, dass anfangs das ermittelte LM-Netz eine kleinere Anzahl von Modellparametern besitzt als gegen Ende. Um die starken Verschlechterungen der Modellgüte im Verlauf zu erklären, ist in Abbildung 5.14 der Verlauf der Hyperparameter zu sehen, die in den jeweiligen Schritten der Liniensuche verwendet wurden. Dabei ist der Verlauf der beiden Hyperparameter des inkrementellen Clustering-Verfahrens und der vier Hyperparameter der Laguerre-Filter zu sehen. Der Verlauf zeigt, dass dabei die richtige Wahl der Zeitkonstanten und die richtige Wahl der beiden Hyperparameter des inkrementellen Clustering-Verfahrens einen großen Einfluss auf die Modellgüte hatte. Die beobachteten starken Verschlechterungen der Modellgüte entstanden, wenn eine der beiden Ordnungen der Laguerre-Filter null war, also wenn ein Eingangssignal als statisches Signal verwendet wird. Die Ergebnisse der Liniensuche zur Erstellung des LM-Netzes, das das oben beschriebene nichtlineare dynamische System approximiert, sind in Tabelle 5.1 zusammengefasst.
Zur Bewertung der numerischen Vereinfachung durch das LM-Netz wurde betrachtet, welche Rechenzeit sich ergab. Dazu wurde betrachtet welche Zeit zum Einen in der Simulation zur Erstellung der gefilterten Daten benötigt wurde. Zum Anderen wurde betrachtet welche Zeit das erstellte LM-Netz benötigte, um mit den gleichen Eingangssignalen die entsprechenden Ausgangssignale zu allen Zeitpunkten zu berechnen. Dabei wurde für die Berechnung die ursprüngliche Abtastzeit $T_S = 0,01[\mathrm{s}]$ verwendet. Das LM-Netz benötigte für die gleichen Eingangssignale nur $20[\mathrm{s}]$. Dies entsprach einer Verkürzung der Rechenzeit um den Faktor 9 und bedeutete somit eine deutliche numerische Vereinfachung.

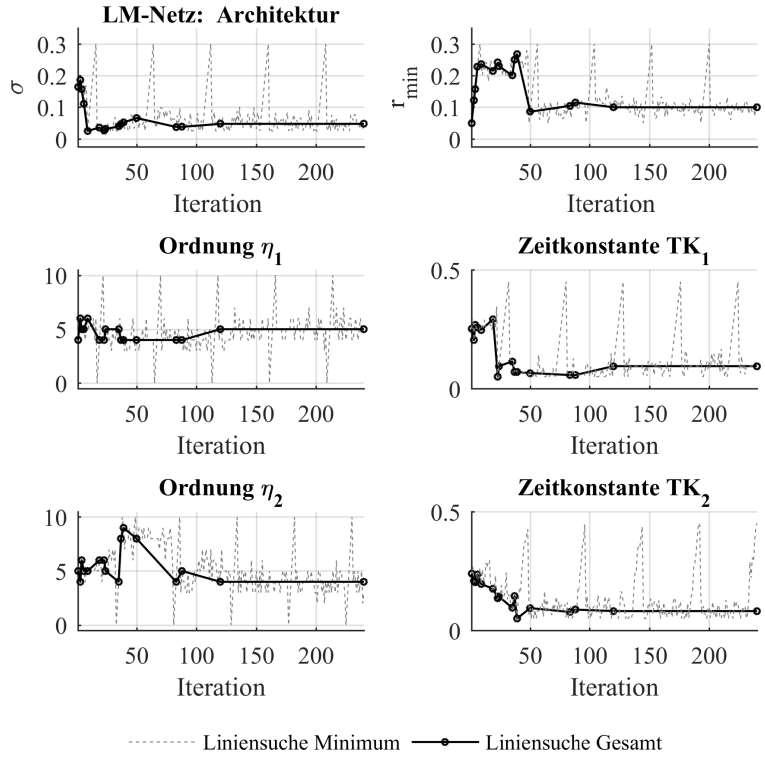


Abbildung 5.14: Verlauf der Hyperparameter während der Liniensuche bei Verwendung eines LM-Netzes

5.4.2 Modellierung mit einem MLP-Netz

Zur Erstellung des MLP-Netzes zur Modellierung des nichtlinearen dynamischen Systems wurde zusätzlich zu oben beschriebenen Suchräumen der Suchraum für die Anzahl M von Neuronen benötigt.

Iterationen Liniensuche	240	Ermittelte Hyperparameter	
Dauer Liniensuche	$t = 1,5[\mathrm{h}]$	Ordnungen	$\mathsf{BP}_{\eta} = \begin{pmatrix} 5 & 4 \end{pmatrix}$
Trainingsfehler	$\mathrm{RMSE}_{\mathsf{T}} = 0,016$	Zeitkonstanten	$\mathsf{BP}_{T_{\mathsf{K}}} = \begin{pmatrix} 0,094 & 0,081 \end{pmatrix}$ [s]
Validierungsfehler	$\mathrm{RMSE}_{\mathsf{V}} = 0,016$	Anzahl lokale Modele	$\mathsf{BP}_{\mathsf{M}} = 109$
Testfehler	$\mathrm{RMSE}_{\mathsf{TS}} = 0,016$	Anzahl Modellparameter	$n_{\mathsf{w}} = 1090$
		Clustering	$\mathsf{BP}_{r_{\min}} = 0,10$ $\mathsf{BP}_{\sigma} = 0,047$

Tabelle 5.1: Ergebnisse der Liniensuche mit einem LM-Netz

Dieser war gegeben durch

$$\mathsf{SP}_{\mathsf{M}} = \begin{pmatrix} 9 & 12 & 15 & 18 & 21 & 24 \end{pmatrix}. \tag{5.19}$$

Als Startwert für die Liniensuche wurde mit der kleinsten Anzahl von Neuronen der verdeckten Schicht begonnen, wie es oben beschrieben ist. Insgesamt mussten hier 5 Hyperparameter durch die Liniensuche bestimmt werden. Die Anzahl der Modellparameter des MLP-Netzes wurde hier durch die Ordnung der Laguerre-Filter und die Anzahl der Neuronen bestimmt. In Abbildung 5.15 ist wie zuvor der Verlauf der RMSE-Fehler mit den Trainings-, den Validierungs- und den Testdaten während der Liniensuche zu sehen. Dabei ist zu beachten, dass diese in diesem Fall auf einer logarithmischen Skala eingezeichnet sind, da sich hier sehr kleine Fehler ergaben. Zusätzlich ist der Verlauf der Anzahl der Modellparameter dargestellt.

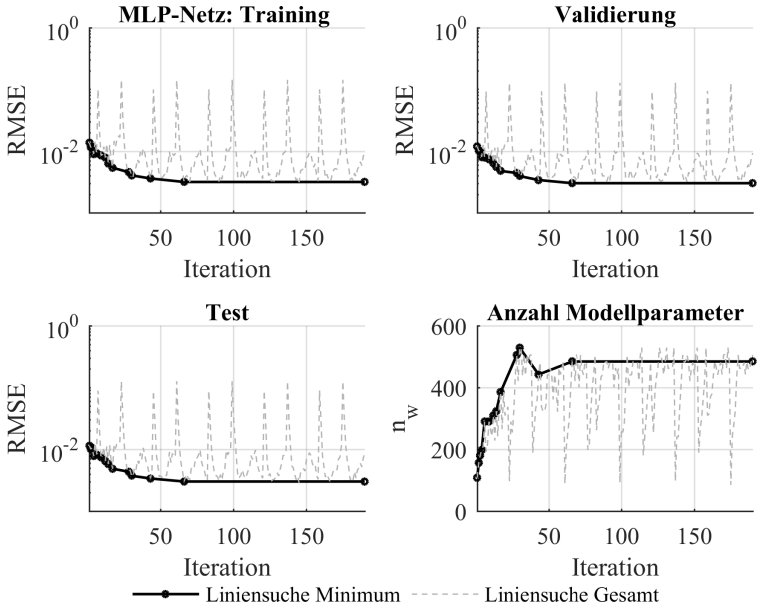


Abbildung 5.15: Verlauf der Modellgüte während der Liniensuche bei Verwendung eines MLP-Netzes

Hier zeigt sich, dass die Modellgüte schon zu Beginn sehr gut war und ebenfalls innerhalb der ersten 50 Iterationen sehr nahe bei der am Ende gefundenen Modellgüte war. Zudem lagen alle drei betrachteten Fehler beim Verlauf des Minimums der Liniensuche immer in der gleichen Größenordnung. Damit konnte eine Überanpassung ausgeschlossen werden. Außerdem fällt hier auf, dass es immer wieder starke Verschlechterungen der Modellgüte gab. Um dies weiter zu untersuchen, ist in Abbildung 5.16 der Verlauf der Hyperparameter während der Liniensuche zu sehen. Dabei sind die Anzahl der Neuronen und die vier Hyperparameter der Laguerre-Filter dargestellt. Hierbei wird deutlich, dass anfangs im Wesentlichen durch eine Erhöhung der Anzahl der Neuronen eine deutliche Verbesserung der Modellgüte erzielt wurde.

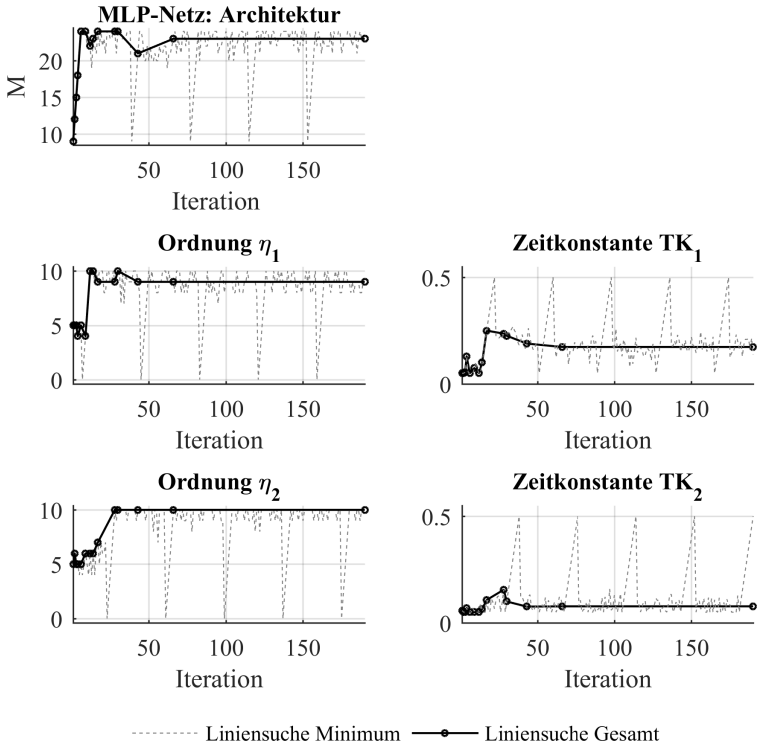


Abbildung 5.16: Verlauf der Hyperparameter während der Liniensuche bei Verwendung eines MLP-Netzes

Im Verlauf der Liniensuche konnte jedoch durch weitere Optimierung der Hyperparameter der Laguerre-Filter die Anzahl der Neuronen etwas reduziert werden, wobei die Modellgüte weiter verbessert wurde. Die Ergebnisse der Liniensuche zur Erstellung eines MLP-Netzes, das das oben beschriebene nichtlineare dynamische System approximiert, sind in Tabelle 5.2 zusammengefasst.
Zur Bewertung der numerischen Vereinfachung durch das MLP-Netz wurde wie beim LM-Netz vorgegangen. Das MLP-Netz benötigte für die gleichen

Eingangssignale nur 15[s], was einer Verkürzung der Rechenzeit um den Faktor 12 entspricht und somit auch hier eine deutliche numerische Vereinfachung bedeutet. Die kürzere Zeit zur Berechnung der Modellausgänge lässt sich durch die geringere Anzahl von Modellparametern des MLP-Netzes erklären.

Iterationen Liniensuche	190	Ermittelte Hyperparameter	
Dauer Liniensuche	$t = 13,5$[h]	Ordnungen	$\mathsf{BP}_{\eta} = \begin{pmatrix} 9 & 10 \end{pmatrix}$
Trainingsfehler	$\mathrm{RMSE}_{\mathsf{T}} = 0,003$	Zeitkonstanten	$\mathsf{BP}_{T_{\mathsf{K}}} = \begin{pmatrix} 0,173 & 0,077 \end{pmatrix}$ [s]
Validierungsfehler	$\mathrm{RMSE}_{\mathsf{V}} = 0,003$	Anzahl Neuronen	$\mathsf{BP}_{\mathsf{M}} = 23$
Testfehler	$\mathrm{RMSE}_{\mathsf{TS}} = 0,003$	Anzahl Modellparameter	$n_{\mathsf{w}} = 484$

Tabelle 5.2: Ergebnisse der Liniensuche mit einem MLP-Netz

5.4.3 Vergleich der erstellten neuronalen Netze

In diesem Abschnitt werden die Ergebnisse der Modellierung durch die beiden neuronalen Netze mit externen Laguerre-Filtern verglichen, die durch Verwendung der oben vorgestellten Liniensuche erstellt wurden. Dazu werden zunächst die Signalverläufe der Referenz und der beiden neuronalen Netze verglichen, die in Abbildung 5.17 zu sehen sind. Dabei ist auch beim Signalverlauf zu erkennen, dass das MLP-Netz eine deutlich bessere Approximation darstellt. Das MLP-Netz kann das dynamische Verhalten sehr genau modellieren, während es beim LM-Netz zu teilweise größeren Abweichungen beim Einschwingverhalten kommt. Die stationären Werte werden jedoch bei beiden Modellen gut approximiert.

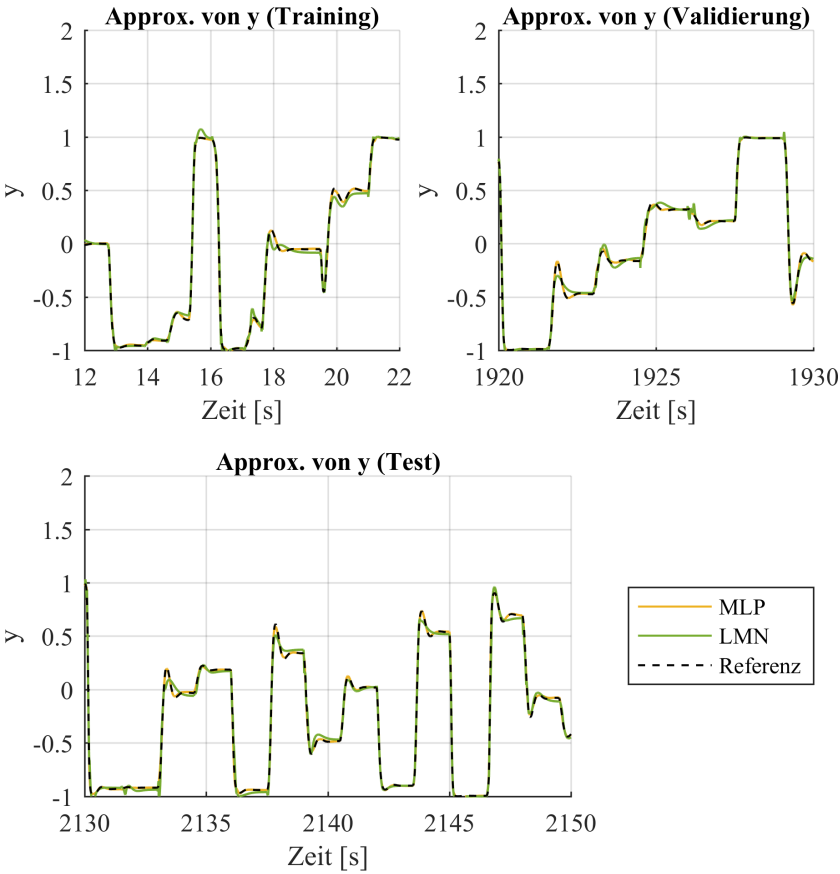


Abbildung 5.17: Approximation des dynamischen nichtlinearen System durch ein MLP-Netz und ein LM-Netz mit externen Laguerre-Filtern

Die Abweichungen beim Einschwingverhalten lassen sich folgendermaßen erklären:

Eine mögliche Erklärung ist, dass beim LM-Netz eine niedrigere Ordnung bei den Laguerre-Filtern verwendet wird. Ein weiterer Grund ist auch der tanh als statische Nichtlinearität am Modellausgang. Da das hier verwendete MLP-Netz diesen auch als Basisfunktion besitzt, ist diese strukturell besser in der Lage das Sättigungsverhalten zu beschreiben. Das erklärt auch, warum dort eine deutlich geringere Anzahl von Modellparametern benötigt wird.
Außerdem ist das verwendete Clustering-Verfahren ein heuristisches Verfahren zur Partitionierung und bedeutet, dass dadurch nicht die optimale Teilung realisiert wird. Zudem werden nicht alle Ordnungen der Laguerre-Filter bei der Erstellung des LM-Netze nichtlinear berücksichtigt. Vorteilhaft ist jedoch, dass der Aufwand zur Erstellung eines LM-Netzes deutlich geringer ist. Allerdings kann hier nicht daraus geschlossen werden, dass sich generell MLP-Netze zur Modellierung besser eignen. In anderen nichtlinearen dynamischen Systemen ist das Sättigungsverhalten weniger stark ausgeprägt oder wird durch eine andere Funktion beschrieben, so dass sich die Unterschiede bei den erhaltenen Modellfehlern wieder relativieren können. Deshalb wird davon ausgegangen, dass sich beide Netze gut zur Modellierung nichtlinearer Systeme verwenden lassen und dass durch Verwendung der Liniensuche eine geeignete Wahl von Hyperparametern gefunden wird. Im nächsten Kapitel wird die Diskussion, welches neuronale Netz sich besser zur Modellierung der Schaltdynamik von Stufenautomaten eignet, erneut aufgegriffen.
Beim Vergleich der optimierten Hyperparameter der Laguerre-Filter fällt weiter auf, dass sich bei den erstellten Netzen andere Werte ergeben. Bei der Erstellung der LM-Netze werden kleinere Ordnungen der Laguerre-Filter bestimmt als bei den MLP-Netzen. Dies kann auch damit begründet werden, dass beim LM-Netz nicht alle Ordnungen nichtlinear berücksichtigt werden. Somit führt auch eine andere Kombination von Ordnungen und Zeitkonstanten zu einem kleinsten Validierungsfehler. Dabei gilt es jedoch zu beachten, dass die Abweichung nur bei der ersten Zeitkonstanten auftritt.

5.5 Fazit

Es wurde gezeigt, wie die Kombination neuronaler Netze mit externen Laguerre-Filtern zur Modellierung nichtlinearer dynamischer Systeme verwendet werden kann. Dazu wurde eine Liniensuche mit stochastischen Elementen zur automatischen Struktur- und Parameteroptimierung beschrieben, wobei sowohl die Modellparameter als auch die Hyperparameter automatisch bestimmt werden. Die Hyperparameter sind aus den Ordnungen und Zeitkonstanten der Laguerre-Filter gegeben. Weitere Hyperparameter beeinflussen die Architektur des verwendeten neuronalen Netzes. Bei vielen Eingangssignalen entsteht durch die Verwendung individueller Pole und individueller Ordnungen eine große Anzahl von Hyperparametern. Deshalb wurde gezeigt, wie durch Unterscheidung zwischen Signalen mit dynamischem und Signalen mit statischem Einfluss die Anzahl der Hyperparameter reduziert werden kann.
Unterschiede beim Ablauf der Liniensuche ergeben sich je nach Auswahl des statischen neuronalen Netzes. Für die Erstellung eines LM-Netzes wird das oben beschriebene inkrementelle Clustering-Verfahren verwendet.
Für MLP-Netze kommt die Anzahl von Neuronen in der verdeckten Schicht als Hyperparameter hinzu. Zur Parameteroptimierung wird das Levenberg-Marquardt-Verfahren verwendet. An einem Beispiel wurde demonstriert, wie sich durch Verwendung der automatisch erstellen Netze eine numerische Vereinfachung erzielen lässt.

6 Methode zur automatischen numerischen Vereinfachung von Getriebemodellen

In diesem Kapitel wird gezeigt, welche Schritte nötig sind, um eine automatische numerische Vereinfachung detaillierter Getriebemodelle zu ermöglichen. Dazu kommt die zuvor vorgestellte Liniensuche zur Struktur- und Parameteroptimierung zum Einsatz. Außerdem wird gezeigt, wie die erstellten neuronalen Netze in einer Simulation der Schaltdynamik verwendet werden können.

6.1 Bearbeitung von Simulationsdaten zur Erstellung der neuronalen Netze

Für die Modellerstellung werden Daten benötigt, die aus Simulationen mit einem energieflussbasierten Modell des Stufenautomaten erzeugt werden. Wie bereits in Kapitel 4 beschrieben, werden diese durch Vorgabe möglichst vieler und unterschiedlicher Betriebsmodi erzeugt, so dass die Schaltdynamik des zu modellierenden Stufenautomaten durch neuronale Netze beschrieben werden kann. Die verschiedenen Betriebsmodi ergeben sich beispielsweise durch unterschiedliche Brems- und Beschleunigungsvorgänge oder Berg- und Talfahrten. Dabei ist zu beachten, dass zur Erstellung der Simulationsdaten möglichst unterschiedliche Szenarien verwendet werden, so dass die Schaltdynamik umfassend enthalten ist, um ein valides neuronales Netz erstellen zu können.
Wie in Kapitel 4 gezeigt wurde, sind die Eingangs- und Ausgangssignale so gewählt, dass die zu erstellenden neuronalen Netze gut in einem signalfluss-

orientierten Fahrzeugmodell, wie es in Abbildung 2.8 zu sehen ist, verwendet werden können. Für die Modellerstellung werden daher aus den Simulationsdaten folgende Signale als Eingangssignale ausgewählt

$$\begin{pmatrix} \mathbf{u}(1) \\ \mathbf{u}(2) \\ \vdots \\ \mathbf{u}(\mathsf{N}) \end{pmatrix} = \begin{pmatrix} i_{\mathsf{G,A,1}}(1) \dots i_{\mathsf{G,A,k}}(1) & T_{\mathsf{m}}(1) & T_{\mathsf{brk}}(1), & \mathsf{m_s}(1) & \omega_{\mathsf{at}}(1) \\ i_{\mathsf{G,A,1}}(2) \dots i_{\mathsf{G,A,k}}(2) & T_{\mathsf{m}}(2) & T_{\mathsf{brk}}(2), & \mathsf{m_s}(2) & \omega_{\mathsf{at}}(2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ i_{\mathsf{G,A,1}}(\mathsf{N}) \dots i_{\mathsf{G,A,k}}(\mathsf{N}) & T_{\mathsf{m}}(\mathsf{N}) & T_{\mathsf{brk}}(\mathsf{N}) & \mathsf{m_s}(\mathsf{N}) & \omega_{\mathsf{at}}(\mathsf{N}) \end{pmatrix}. \tag{6.1}$$

Dabei wird davon ausgegangen, dass zur Ansteuerung des Stufenautomaten k Signale verwendet werden. Hinzu kommen vier Eingangssignale, so dass die gesamte Anzahl von Eingangssignalen $\mathsf{k}+4$ beträgt, die in den neuronalen Netzen verwendet werden. Entsprechend werden die Ausgangssignale aus den Simulationsdaten ausgewählt, die durch die neuronalen Netze modelliert werden

$$\begin{pmatrix} \mathbf{y}(1) \\ \mathbf{y}(2) \\ \vdots \\ \mathbf{y}(\mathsf{N}) \end{pmatrix} = \begin{pmatrix} T_{\mathsf{at}}(1) & \omega_{\mathsf{m}}(1) & i_{\mathsf{G,S,1}}(1) \dots i_{\mathsf{G,S,l}}(1) \\ T_{\mathsf{at}}(2) & \omega_{\mathsf{m}}(2) & i_{\mathsf{G,S,1}}(1) \dots i_{\mathsf{G,S,l}}(1) \\ \vdots & \vdots & \vdots \\ T_{\mathsf{at}}(\mathsf{N}) & \omega_{\mathsf{m}}(\mathsf{N}) & i_{\mathsf{G,S,1}}(1) \dots i_{\mathsf{G,S,l}}(1) \end{pmatrix}. \tag{6.2}$$

Es wird davon ausgegangen, dass es dabei l Sensorsignale gibt. In dieser Arbeit wird angenommen, dass es sich dabei um weitere Drehmomente oder Winkelgeschwindigkeiten handelt, die für die Getriebesteuerung verwendet werden. Dabei bezeichnet N die Anzahl der vorliegenden Abtastpunkte. Eine weitere wichtige Eigenschaft der Simulationsdaten ist die Auswahl der äquidistanten

Abtastzeit T_S. Für jedes Ausgangssignal wird ein separates neuronales Netz mit externen Laguerre-Filtern erstellt, die gegeben sind durch

$$\begin{aligned} T_\mathsf{at}(k) &= \mathsf{NN}_1(\mathbf{u}(k), \mathbf{w}_1, \mathsf{P}_{1,1}, \mathsf{P}_{1,2}, \ldots, \mathsf{P}_{1,\mathsf{m}}) \\ \omega_\mathsf{m}(k) &= \mathsf{NN}_2(\mathbf{u}(k), \mathbf{w}_2, \mathsf{P}_{2,1}, \mathsf{P}_{2,2}, \ldots, \mathsf{P}_{2,\mathsf{m}}) \\ i_{\mathsf{G,S1}}(k) &= \mathsf{NN}_3(\mathbf{u}(k), \mathbf{w}_3, \mathsf{P}_{3,1}, \mathsf{P}_{3,2}, \ldots, \mathsf{P}_{3,\mathsf{m}}) \\ &\vdots \\ i_{\mathsf{G,Sl}}(k) &= \mathsf{NN}_{2+\mathsf{l}}(\mathbf{u}(k), \mathbf{w}_{2+\mathsf{l}}, \mathsf{P}_{2+\mathsf{l},1}, \mathsf{P}_{2+\mathsf{l},2}, \ldots, \mathsf{P}_{2+\mathsf{l},\mathsf{m}}). \end{aligned} \tag{6.3}$$

Um die Modellparameter $\mathbf{w}_i$ und die m Hyperparameter der jeweiligen neuronalen Netze zu bestimmen, kommt die zuvor vorgestellte Liniensuche zum Einsatz. Dabei müssen im Allgemeinen $2 \cdot (\mathsf{k}+4)$ Hyperparameter für die individuellen Ordnungen und Pole der Laguerre-Filter optimiert werden. Hinzu kommen ein oder mehrere Hyperparameter, die die Architektur des neuronalen Netzes beeinflussen. Im Falle eines MLP-Netzes mit externen Laguerre-Filtern ist dies die Anzahl der Neuronen in der verdeckten Schicht. Wird ein LM-Netz verwendet, sind diese Hyperparameter der kleinste Abstand r_{min} und die Standardabweichung σ, die zur Bestimmung der Aktivierungsfunktionen verwendet werden[1].

Für die Modellerstellung werden die Simulationsdaten weiter eingeteilt in Trainings-, Validierungs- und Testdaten. Dabei werden die ersten 85% der erstellten Simulationsdaten für das Training und die Validierung verwendet. Dazu werden diese weiter unterteilt, um die Struktur- und Parameteroptimierung durch die Liniensuche durchführen zu können. Die ersten 85% der Simulationsdaten werden weiter unterteilt in 90%, die zur Parameteroptimierung ,und in 10%, die zur Validierung in jedem Schritt der Liniensuche verwendet werden. Die

[1] Bei Verwendung lokaler Modellnetze sind mit Modellparametern die Parameter der lokalen Modelle gemeint. Durch das Clustering-Verfahren werden dabei die Zentren der jeweiligen Aktivierungsfunktionen bestimmt.

restlichen 15% der gesamten Simulationsdaten werden für die abschließende Bewertung der erstellten neuronalen Netze verwendet.

In den nächsten Abschnitten wird vorgestellt, mit welchen Schritten eine automatische numerischer Vereinfachung ermöglicht wird. Dazu werden folgende Punkte beleuchtet:

- Auswahl des neuronalen Netzes:
  Es wird diskutiert, wann sich ein MLP-Netz und wann sich ein LM-Netz eignet.
- Reduktion der Anzahl von Hyperparametern:
  Es wird beschrieben, wie durch Verwendung von Wissen über die Subsysteme von Stufenautomaten die Anzahl von Hyperparametern reduziert werden kann.
- Verwendung von generischen Modellstrukturen:
  Es wird gezeigt, dass zur Modellierung von Drehmomenten andere Eingangssignale verwendet werden als zur Modellierung von Winkelgeschwindigkeiten. Außerdem wird gezeigt, welche Unterschiede sich bei der Verwendung von MLP-Netzen und LM-Netzen in der Erstellung der Regressoren ergeben.
- Vorgabe der Suchräume:
  Als letzter Punkt wird darauf eingegangen, welche Suchräume sich für die Liniensuche ergeben und welche Werte und Intervalle für diese gewählt werden.

6.2 Auswahl des neuronalen Netzes

Zum Start der Modellerstellung muss ausgewählt werden, ob ein LM-Netz oder ein MLP-Netz in Kombination mit externen Laguerre-Filtern verwendet werden soll. Für die Verwendung eines MLP-Netzes spricht dessen Eignung für

hochdimensionale Eingangsräume, die bei der Modellierung der Schaltdynamik durch die Verwendung externer Laguerre-Filter sehr schnell entstehen können (siehe Kapitel 4). Zudem ist die Berechnung des Modellausgangs bei MLP-Netzen etwas schneller als bei LM-Netzen, die hier meist weniger Neuronen verwendet werden, und die Berechnung des Ausgangswerts der Neuronen weniger rechenintensiv ist als die Berechnung des Ausgangswerts von normierten Gaußfunktionen.
Gegen die Verwendung von MLP-Netzen spricht, dass die Parameteroptimierung hier aufwändiger ist. Dabei ist besonders das empfohlene Levenberg-Marquardt-Verfahren speicheraufwändig. Außerdem empfiehlt es sich hier, auf eine von einem spezialisierten Hersteller unterstützte Implementierung zurückzugreifen, da das Verfahren nicht einfach umzusetzen ist.
Die letzten Punkte treffen nicht auf die Verwendung von LM-Netzen zu. Hier können auch einfachere Verfahren zur Parameteroptimierung selbst implementiert werden (siehe Kapitel 4). Dabei sind der Rechenaufwand und somit ist auch die Zeit geringer, die für die Struktur- und Parameteroptimierung benötigt wird. Jedoch muss weiteres Wissen über die Schaltdynamik verwendet werden, bzw. es müssen Annahmen getroffen werden, damit auch LM-Netze bei hochdimensionalen Eingangsräumen verwendet werden können. Da die Struktur- und Parameteroptimierung automatisch abläuft, ist es vorteilhaft, sich für MLP-Netze zu entscheiden, da diese besser mit dem hochdimensionalen Eingangsraum umgehen können und auch eine größere numerische Vereinfachung ermöglichen. In diesem Kapitel wird jedoch gezeigt, wie auch LM-Netze auch bei hochdimensionalen Eingangsräumen zur Modellierung der Schaltdynamik verwendet werden können. Dies ist dann von Vorteil, wenn auf eine kommerzielle Implementierung des Levenberg-Marquardt-Verfahrens verzichtet werden soll.

6.3 Aufbau der neuronalen Netze zur Modellierung der Schaltdynamik

Nun wird der Aufbau der neuronalen Netze mit externen Laguerre-Filtern näher beschrieben, die zur Modellierung der Schaltdynamik verwendet werden. Dazu wird zunächst gezeigt, wie die Anzahl der Hyperparameter der externen Laguerre-Filter reduziert werden kann, damit die Anzahl der Suchräume für die spätere Liniensuche sinkt.

6.3.1 Reduktion der Hyperparameteranzahl

Durch die hohe Anzahl k von Ansteuersignalen bei Stufenautomaten entsteht eine hohe Anzahl von Hyperparametern, wenn individuelle Pole und Ordnungen in den Laguerre-Filtern verwendet werden, um die Eingangssignale zu filtern. Deshalb wird im Folgenden gezeigt, wie sich diese Anzahl der Hyperparameter reduzieren lässt. Dazu wird Wissen über die Schaltdynamik verwendet, um die Eingangssignale je nach statischem und dynamischem Einfluss zu unterteilen. Anschließend wird darauf eingegangen, wie die Eingangssignale mit dynamischem Einfluss zusammengefasst werden können, um bei diesen jeweils die gleiche Ordnung und Zeitkonstante zur Filterung durch Laguerre-Filter zu verwenden.

Eingangssignale mit statischem Einfluss

Im Folgenden wird begründet, dass die Steigung m_s und die Winkelgeschwindigkeit ω_{at} der Antriebswelle Eingangssignale mit statischem Einfluss sind.
Die Steigung m_s bewirkt eine im Vergleich zur Schaltdynamik sehr langsame Lastpunktverschiebung. Somit würde eine Filterung durch Laguerre-Filter wie ein Zeitverzögerungsoperator wirken und keine Vorteile für die Modellierung

bringen, sondern nur die Dimension des Eingangsraums unnötig erhöhen.
Ebenso ändert sich die Winkelgeschwindigkeit ω_{at} der Antriebswelle im Vergleich zur Schaltdynamik langsam. Dies liegt an der Trägheit des Fahrzeugs. Somit kann die Winkelgeschwindigkeit während eines Schaltvorgangs als statische Einflussgröße angenommen werden. Auch hier würde eine Filterung nur die Dimension des Eingangsraums unnötig erhöhen.
Die restlichen Eingangssignale haben einen dynamischen Einfluss.

Zusammenfassung von Eingangssignalen mit gleichem dynamischen Einfluss auf die Schaltdynamik

Im Folgenden wird motiviert, die Eingangssignale mit dynamischem Einfluss in drei Gruppen aufzuteilen:

1. Die erste Gruppe beinhaltet die Ansteuersignale $\mathbf{i}_{\mathrm{G,A,1}}$, die zur Ansteuerung der Schaltelemente dienen.
2. Die zweite Gruppe beinhaltet die restlichen Ansteuersignale $\mathbf{i}_{\mathrm{G,A,2}}$. Diese werden als zusätzliche Ansteuerungsmöglichkeit des Getriebes verwendet.
3. Die dritte Gruppe beinhaltet die Drehmomente, die auf das Getriebe einwirken.

Diese Einteilung führt dazu, dass drei Ordnungen und drei Zeitkonstanten bei der Liniensuche neben den Hyperparametern der Architektur des neuronalen Netzes optimiert werden müssen. Die Anzahl der dabei insgesamt zu optimierenden Hyperparameter ist unabhängig von der Anzahl der Ansteuersignale.
Die beschriebene Einteilung der Eingangssignale kann folgendermaßen begründet werden: Die erste Gruppe besteht aus sich im Vergleich zur Schaltdynamik schnell ändernden Signalen, deren Veränderung keine unmittelbare Auswirkung

zeigt, was durch das gedämpfte dynamische Übertragungsverhalten erklärt werden kann. Dieses Übertragungsverhalten kann daher durch Verwendung von Laguerre-Filtern approximiert werden.
Die zweite Gruppe besteht aus Ansteuersignalen, die die Schaltvorgänge zusätzlich unterstützen bzw. ändern. Beispiele für solche Ansteuersignale sind die Ansteuerung der WÜK oder die Anpassung des Arbeitsdrucks der elektrohydraulischen Ansteuerung, um das Schaltverhalten geringfügig zu modifizieren. Um dies entsprechend bei der Modellierung berücksichtigen zu können, werden diese Signale durch eine andere Ordnung und Zeitkonstante in den Laguerre-Filtern gefiltert.
Die dritte Gruppe besteht aus dem Drehmoment T_{m} des Motors und dem Drehmoment T_{brk} der Bremse. Durch diese beiden Drehmomente wird das dynamische Verhalten in den Schaltelementen und im Wandler beeinflusst, was sich entsprechend auf die Schaltdynamik auswirkt. Um dies zu approximieren, werden eine dritte Ordnung und eine dritte Zeitkonstante in den Laguerre-Filtern verwendet.
Bei der Auswahl von Eingangssignalen zur Modellierung der benötigten Ausgangssignale wird die beschriebene Einteilung beibehalten. Es wird jedoch zwischen der Modellierung von Winkelgeschwindigkeiten und Drehmomenten unterschieden, wofür unterschiedliche Eingangssignale verwendet werden. Die Auswahl der jeweiligen Eingangssignale wird im Folgenden behandelt.

6.3.2 Auswahl von Eingangssignalen zur Modellierung von Winkelgeschwindigkeiten

In diesem Abschnitt wird auf die Auswahl der Eingangssignale zur Modellierung von Winkelgeschwindigkeiten eingegangen. Dazu werden alle oben genannten Eingangssignale verwendet, und es wird die Einteilung in statische und dynamische Eingangssignale beibehalten.

Aufbau von MLP-Netzen zur Modellierung von Winkelgeschwindigkeiten

In Abbildung 6.1 ist die generische Architektur von MLP-Netzen mit externen Laguerre-Filtern zu sehen, die zur Modellierung von Winkelgeschwindigkeiten verwendet werden.

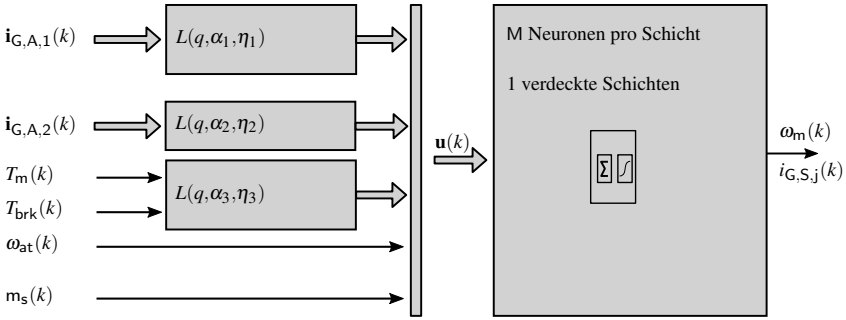


Abbildung 6.1: Eingangssignale zur Modellierung von Winkelgeschwindigkeiten mit MLP-Netzen und externen Laguerre-Filtern

Bei Verwendung eines MLP-Netzes werden die gefilterten und ungefilterten Signale als Eingänge verwendet, und es findet keine weitere Unterscheidung statt. Somit werden alle Signale nichtlinear bei der Modellierung berücksichtigt, da ein MLP-Netz weniger sensitiv auf den Fluch der Dimensionalität reagiert. Das heißt, es können hier auch höhere Ordnungen gefilterter Signale nichtlinear berücksichtigt werden.

Aufbau von LM-Netzen zur Modellierung von Winkelgeschwindigkeiten

In Abbildung 6.2 ist die generische Architektur von LM-Netzen mit externen Laguerre-Filtern zu sehen, die zur Modellierung von Winkelgeschwindigkeiten verwendet werden.

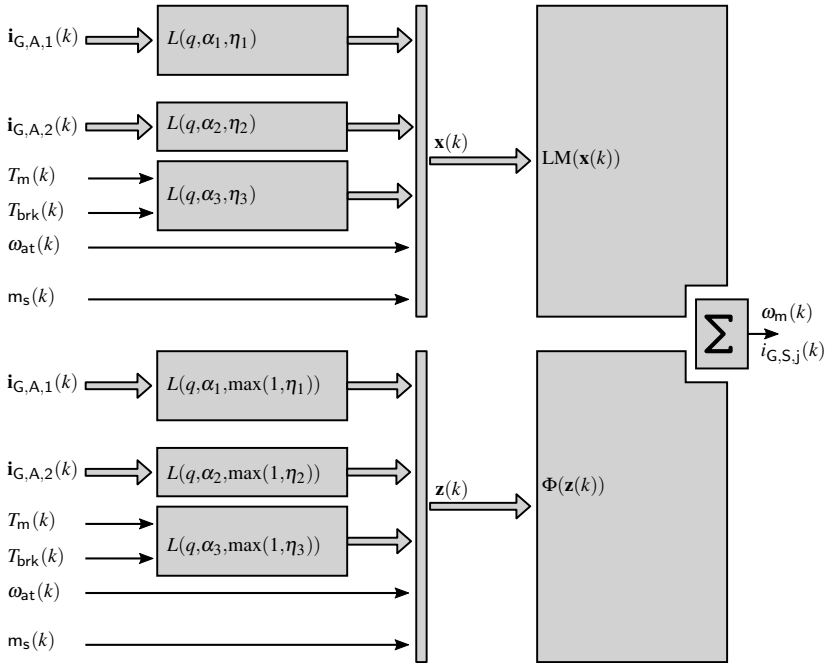


Abbildung 6.2: Eingangssignale zur Modellierung von Winkelgeschwindigkeiten mit LM-Netzen und externen Laguerre-Filtern

Dabei können als Modellausgang die Winkelgeschwindigkeit ω_{m} des Motors oder andere Winkelgeschwindigkeiten $i_{\mathsf{G,S,j}}$, die als Sensorgrößen benötigt werden, verwendet werden.

Zur Modellierung von Winkelgeschwindigkeiten wird bei Verwendung von LM-Netzen von der Möglichkeit Gebrauch gemacht, unterschiedliche Eingangssignale im $\mathbf{x}$-Regressor und im $\mathbf{z}$-Regressor zu verwenden. So werden in den lokalen Modellen die Ansteuerströme $\mathbf{i}_{\mathsf{G,A},1}$ nicht verwendet. Somit gehen diese ausschließlich nichtlinear in die Modellierung ein und werden nur bei der Partitionierung berücksichtigt. In den Anwendungen zur Modellierung der

Schaltdynamik hat sich gezeigt, dass diese Art der Aufteilung vorteilhaft ist. Um den Raum zur Partitionierung nicht zu groß werden zu lassen, werden im **z**-Regressor nur gefilterte Signale bis zur ersten Ordnung verwendet. Bei zu hohen Ordnungen steigt die Dimension zu sehr an und es kommt zu einer zu dünnen Belegung des Eingangsraums mit Daten (siehe Fluch der Dimensionalität in Kapitel 4).

6.3.3 Auswahl von Eingangssignalen zur Modellierung von Drehmomenten

Als Eingangssignale werden alle bis auf die Winkelgeschwindigkeit ω_{at} der Antriebswelle verwendet. Dies bringt für die Simulation deutlich mehr Stabilität und ein weniger fehleranfälliges Modell. Die Wirkzusammenhänge sind in Abbildung 6.3 zu sehen, wobei bereits auf die Darstellung der Rückkopplung der Winkelgeschwindigkeit verzichtet wurde.

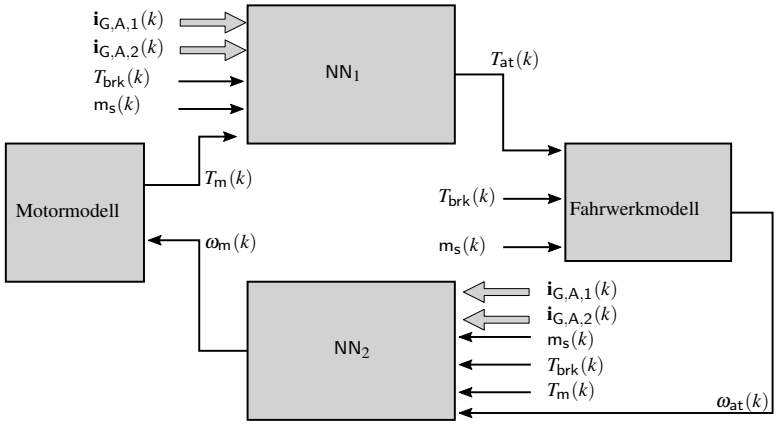


Abbildung 6.3: Blockschaltbild zur Simulation der Schaltdynamik mit neuronalen Netzen

Der höhere Grad an Stabilität kann folgendermaßen begründet werden:
In einer Simulation wird zunächst das neuronale Netz zur Modellierung des Drehmoments dazu verwendet, das dynamische Drehmoment der Antriebswelle zum Simulationszeitpunkt k zu berechnen. Dieses wird dann im Modell des Fahrwerks verwendet, um die Winkelgeschwindigkeit der Antriebswelle zum Simulationszeitpunkt k zu berechnen. Diese kann dann im zweiten neuronalen Netz zur Berechnung der Winkelgeschwindigkeit des Motors verwendet werden. Würde nun die Winkelgeschwindigkeit der Antriebswelle auch als Eingangssignal im neuronalen Netz verwendet, das das Drehmoment der Antriebswelle zum Simulationszeitpunkt k berechnet, läge ein implizites Gleichungssystem vor. Dies ließe sich durch Verwendung eines Zeitverzögerungsoperators vermeiden, allerdings entstünde dadurch implizit eine Rückführung zeitverzögerter Modellausgänge. Wie bereits in Kapitel 4 beschrieben, kann dies zu instabilem Verhalten in einer Simulation führen. Zudem wird der Aufwand der Parameteroptimierung erhöht. Daher wird auf die Verwendung der Winkelgeschwindigkeit ω_{at} als Eingangssignal bei der Modellierung des Drehmoments T_{at} verzichtet.

Aufbau von MLP-Netzen zur Modellierung von Drehmomenten

In Abbildung 6.4 ist die generische Modellstruktur der MLP-Netze zu sehen, die zur Modellierung des Drehmoments T_{at} der Antriebswelle bzw. anderer Drehmomente, die beispielsweise als Sensorsignale benötigt werden, verwendet wird.

Dabei unterscheidet sich der Aufbau des MLP-Netzes im Vergleich zum MLP-Netz zur Modellierung von Winkelgeschwindigkeiten nur darin, dass die Winkelgeschwindigkeit der Antriebswelle nicht verwendet wird. Jedoch können hier andere Ordnungen und Zeitkonstanten in den Laguerre-Filtern verwendet werden. Zudem kann auch eine andere Architektur des MLP-Netzes verwendet werden, wie zum Beispiel eine andere Anzahl von Neuronen.

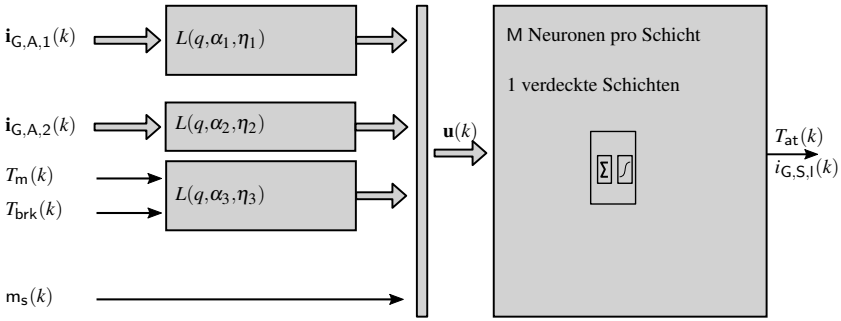


Abbildung 6.4: Eingangssignale zur Modellierung von Drehmomenten mit MLP-Netzen und externen Laguerre-Filtern

Aufbau von LM-Netzen zur Modellierung von Drehmomenten

In Abbildung 6.5 ist die generische Architektur lokaler Modellnetze mit externen Laguerre-Filtern zu sehen, die zur Modellierung von Drehmomenten verwendet werden.
Für die spätere Anwendung ist dies das Drehmoment T_{at}, das an der Antriebswelle anliegt. Eine andere Möglichkeit ist die Verwendung eines Drehmoments, das als Sensorsignal benötigt und entsprechend im Getriebesteuergerät verarbeitet wird. Für die Ordnungen und Zeitkonstanten können hier andere Werte in den Laguerre-Filtern verwendet werden, als es bei den LM-Netzen zur Modellierung von Winkelgeschwindigkeiten der Fall ist. Hier ist zu sehen, dass die Winkelgeschwindigkeit der Antriebswelle nicht als Eingangssignal verwendet wird. Zudem wird die gleiche Beschränkung der Ordnungen für die gefilterten Signale im $\mathbf{z}$-Regressor verwendet wie bei der Modellierung von Winkelgeschwindigkeiten.

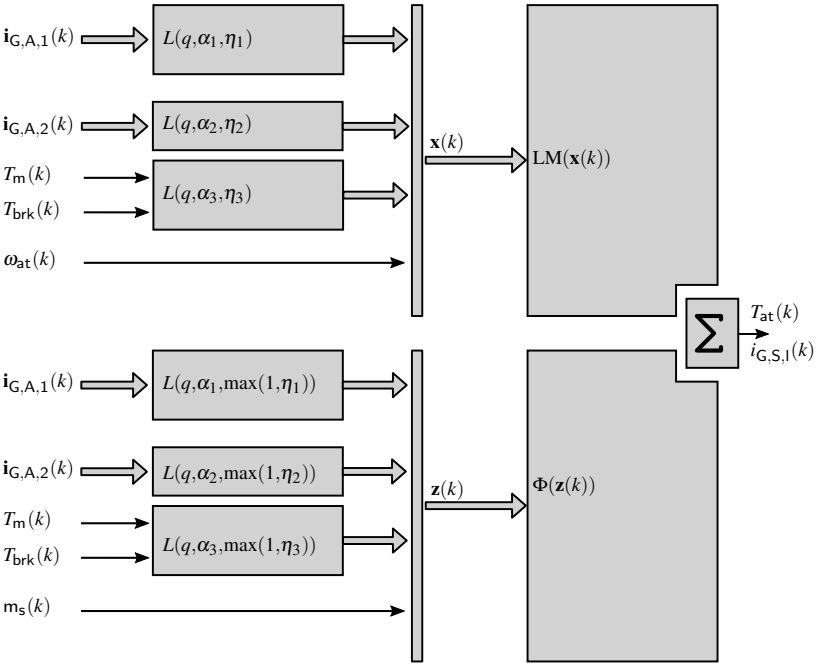


Abbildung 6.5: Eingangssignale zur Modellierung von Drehmomenten mit LM-Netzen und externen Laguerre-Filtern

6.4 Automatische Erstellung der neuronalen Netze

Nun wird darauf eingegangen, wie die zuvor vorgestellten generischen Modellstrukturen in Kombination mit der in Kapitel 5 beschriebenen Liniensuche zur automatischen Modellerstellung verwendet werden. Dazu wird als erstes auf die Suchräume eingegangen, die bei der Liniensuche verwendet werden.

6.4.1 Vorgabe der Suchräume für die Liniensuche

Wie im vorherigen Kapitel beschrieben, können die Suchräume, die für die Liniensuche benötigt werden, in folgende drei Kategorien aufgeteilt werden:

- ein oder mehrere Suchräume zur Optimierung der Architektur des ausgewählten neuronalen Netzes
- drei Suchräume zur Optimierung der drei Ordnungen
- drei Suchräume zur Optimierung der drei Zeitkonstanten

Die Suchräume zur Optimierung werden getrennt vorgestellt, je nach Auswahl des neuronalen Netzes. Die Suchräume zur Optimierung der Ordnungen und Zeitkonstanten werden jedoch unabhängig vom ausgewählten neuronalen Netz verwendet und im Folgenden beschrieben.
Für die drei Ordnungen wird die Verwendung folgender Suchräume empfohlen:

$$\begin{aligned} \mathsf{SP}_{\eta_1} &= \begin{pmatrix} 0 & 2 & 4 & 6 & 8 & 10 & 12 \end{pmatrix} \\ \mathsf{SP}_{\eta_2} &= \begin{pmatrix} 0 & 2 & 4 & 6 & 8 & 10 & 12 \end{pmatrix} \\ \mathsf{SP}_{\eta_3} &= \begin{pmatrix} 0 & 2 & 4 & 6 & 8 & 10 & 12 \end{pmatrix}. \end{aligned} \tag{6.4}$$

Die Auswahl der Ordnungen wird dadurch begründet, dass es sich bei Schaltvorgängen im Allgemeinen um ein gedämpftes Verhalten der ersten bzw. zweiten Ordnung handelt. Die beiden Beispiele aus Kapitel 4, bei denen Laguerre-Filter zur Modellierung eines gedämpften linearen dynamischen Systems zweiter Ordnung und eines nichtlinearen dynamischen Systems, das stark gedämpfte lineare dynamische Systeme zweiter Ordnung enthält, verwendet wurden, motivieren, dass eine maximale Ordnung von 12 für eine gute Approximationsgüte ausreicht. Das Auslassen der ungeraden Ordnungen heißt nicht, dass diese bei der Liniensuche nicht vorkommen, da stochastische Elemente verwendet werden.

Die Zeitkonstanten, die zur Bestimmung der Pole in den Laguerre-Filtern vorgegeben werden, orientieren sich an der Zeitdauer von Schaltvorgängen. Diese liegt üblicherweise zwischen $0,5[\mathrm{s}]$ und $2[\mathrm{s}]$. Wie in Kapitel 4 beschrieben, kann dies dazu verwendet werden, die Zeitkonstante abzuschätzen, indem ein Drittel dieser Zeitdauer als Zeitkonstante für Schaltvorgänge ansetzt wird. Deshalb wird die Verwendung folgender drei Suchräume für die Zeitkonstanten empfohlen:

$$\begin{aligned} \mathsf{SP}_{T_{\mathrm{K},1}} &= \mathrm{linspace}\left(\frac{3}{20}\,,\,\frac{7}{10}\,,\,7\right) \\ \mathsf{SP}_{T_{\mathrm{K},2}} &= \mathrm{linspace}\left(\frac{3}{20}\,,\,\frac{7}{10}\,,\,7\right) \\ \mathsf{SP}_{T_{\mathrm{K},3}} &= \mathrm{linspace}\left(\frac{3}{20}\,,\,\frac{7}{10}\,,\,7\right). \end{aligned} \tag{6.5}$$

Durch die Einteilung in sieben Intervalle wird die Zeitkonstante jeweils in etwa $0,1[\mathrm{s}]$-Schritten optimiert, wobei wieder durch die stochastischen Elemente Zwischenwerte gefunden werden können. Das Beispiel im letzten Kapitel, bei dem ein nichtlineares dynamisches System modelliert wurde, hat gezeigt, dass solch eine Vorgabe von Werten in den Suchräumen zur Optimierung der Zeitkonstanten ausreicht.
Für die Parameteroptimerung der neuronale Netzen kann eine Ausdünnung der gefilterten Trainingsdaten erfolgen, da die Abtastzeit der Simulationsdaten üblicherweise $T_{\mathsf{S}} = 0,01[\mathrm{s}]$ beträgt und in einer späteren Simulation mit den erstellten neuronalen Netzen beibehalten soll [Güh02; Güh03]. Diese ist jedoch höher als nötig, um die Schaltdynamik zu erfassen, da der kleinste Wert in den Suchräumen für diese Zeitkonstante $0,15[\mathrm{s}]$ ist. Daher reicht es, für die Parameteroptimierung eine Abtastzeit von $\frac{1}{15}[\mathrm{s}]$ zu verwenden, was zu einer erheblichen Verkürzung der benötigten Zeit für die Parameteroptimierung führt.

Als maximale Anzahl $\mathsf{N_O}$ von Suchdurchläufen wird $\mathsf{N_O} = 5$ empfohlen. Die Erfahrungen, die bei der Verwendung der Liniensuche gemacht wurden, zeigen,

dass nach zwei, jedoch spätestens drei Suchdurchläufen kaum signifikante Verbesserungen bei der Modellgüte erreicht werden. Dies heißt nicht, dass das globale Minimum gefunden wurde. In den meisten Fällen wird jedoch ein für die Anwendung akzeptables lokales Minimum gefunden. Somit stellt der empfohlene Wert von 5 Suchdurchläufen eine eher konservative Angabe dar.
Als nächstes wird auf die Suchräume zur Optimierung der Architektur von MLP-Netzen und LM-Netzen eingegangen. Da dann alle Suchräume und Vorgaben zur Liniensuche vorhanden sind, wird auch vorgestellt, wie die neuronalen Netze automatisch erstellt werden können.

6.4.2 Suchräume zur automatischen Erstellung von MLP-Netzen

Zur Optimierung der Architektur von MLP-Netzen wird der Suchraum der Anzahl M der Neuronen der verdeckten Schicht benötigt. Dabei wird zur Modellierung der Schaltdynamik von Stufenautomaten die Verwendung einer verdeckten Schicht empfohlen, da erstens die Gefahr einer Überanpassung mit mehreren verdeckten Schichten zu groß ist, zweitens der Zeitaufwand für die Parameteroptimierung stark ansteigt und drittens die Modellgüte sich nicht signifikant verbessert.
Zur Vorgabe, welche Anzahl M von Neuronen in der verdeckten Schicht verwendet werden soll, gibt es keine feste Regel [Nel01]. Architekturbedingt kann diese Anzahl auch schwer interpretiert und deshalb zum Beispiel nicht mit der Anzahl der Gangstufen in Verbindung gebracht werden. Es wird daher empfohlen, folgenden Suchraum standardmäßig für die Anzahl M zu verwenden:

$$SP_M = \begin{pmatrix} 5 & 7 & 10 & 12 & 15 & 17 & 20 & 22 & 25 \end{pmatrix}. \tag{6.6}$$

Für die meisten Anwendungen liegt die Anzahl der Neuronen der verdeckten Schicht in diesem Bereich. Durch die Liniensuche kann anschließend die genaue Anzahl festgelegt werden. Zur Parameteroptimierung wird das Levenberg-

Marquardt-Verfahren verwendet, für das keine weiteren Vorgaben notwendig sind. Damit sind alle Angaben zur Erstellung der MLP-Netze mit externen Laguerre-Filtern vollständig und es können die in Abbildung 6.1 bzw. Abbildung 6.4 gezeigten Netze zur Modellierung von Winkelgeschwindigkeiten bzw. Drehmomenten verwendet werden.

6.4.3 Suchräume automatischen Erstellung von LM-Netzen

Um die Architektur von LM-Netzen zu optimieren, wird jeweils ein Suchraum für den kleinsten Abstand r_{min} und mit der einheitlichen Skalierung σ benötigt, die beim inkrementellen Clustering-Verfahren verwendet werden. Dazu werden folgende Suchräume verwendet:

$$\begin{aligned} \mathsf{SP}_{r_{\mathsf{min}}} &= \text{linspace}\left(\frac{1}{10}\,,\,\frac{2}{5}\,,\,7\right) \\ \mathsf{SP}_{\sigma} &= \text{linspace}\left(\frac{3}{20}\,,\,\frac{2}{5}\,,\,7\right). \end{aligned} \tag{6.7}$$

Dabei zeigt die Erfahrung, dass durch die Verwendung dieser Suchräume gute Resultate mit dem inkrementellen Clustering-Verfahren erzielt werden. Dazu werden, wie schon bei der Modellierung des nichtlinearen dynamischen Systems, die lokalen Modelle in 12 Schritten hinzugefügt, bis die maximal erlaubte Anzahl von Modellparametern erreicht ist. Allerdings ist die Angabe zur maximalen Anzahl erlaubter Modellparameter abhängig davon, wie viele Simulationsdaten verwendet werden und wie komplex das zu modellierende Schaltverhalten ist. Bei einer Vielzahl von Ansteuerströmen ist zu erwarten, dass hier eine höhere Anzahl von Modellparametern benötigt wird. Als Orientierung kann dabei die Verwendung von etwa 300 Modellparametern pro Gangstufe dienen, wobei dieser Wert auf Erfahrungen beruht, die bei der Modellierung mit LM-Netzen und externen Laguerre-Filtern gesammelt wurden [CNG14]. Damit

sind alle Angaben vollständig, die für die Liniensuche benötigt werden, und somit können LM-Netze zur Modellierung der Schaltdynamik erstellt werden. Dazu werden die Trainings- und Validierungsdaten vorgegeben, und je nachdem, ob eine Winkelgeschwindigkeit oder ein Drehmoment modelliert werden soll, kommt das in Abbildung 6.2 bzw. Abbildung 6.5 gezeigte LM-Netz zum Einsatz.

6.5 Verwendung der erstellten neuronalen Netze in der Simulation

In diesem Abschnitt wird darauf eingegangen, wie die erstellten neuronalen Netze in einer Simulation der Schaltdynamik verwendet werden können. Das Simulationsmodell wird dabei entsprechend der zwei in diesem Kapitel beschriebenen generischen Modellstrukturen angelegt. Dies kann zum Beispiel als s-Function in Matlab/Simulink [Mat] oder als reine C-Funktion [KR88] implementiert werden. Um das Modell in einer Simulation verwenden zu können, müssen folgende Informationen übergeben werden:

- Normierungsparameter:
  Es müssen die Werte übergeben werden, die zur Normierung verwendet wurden. Diese werden intern dazu verwendet, die Eingangssignale zu normieren und die Rücktransformation des Ausgangssignals auf den ursprünglichen Wertebereich durchzuführen.
- Abtastzeit, Pol und optimierte Ordnung der Laguerre-Filter:
  Für die Berechnung der gefilterten Signale muss im Simulationsmodell extra Speicher angelegt werden, und die gefilterten Signale müssen auch pro Zeitschritt aktualisiert werden. Weiter ist es vorteilhaft, die rekursive Berechnung der gefilterten Signale durch Laguerre-Filter zu verwenden (siehe Abbildung 5.8). Die gefilterten Signale werden nach

der Normierung dazu verwendet, die Regressoren des neuronalen Netzes aufzubauen. Im Falle eines lokales Modellnetzes muss weiter zwischen dem **x**-Regressor und dem **z**-Regressor unterschieden werden, was etwas mehr Aufwand für die Implementierung bedeutet.

- Modellparameter des neuronalen Netzes:
  Es müssen die Modellparameter des neuronalen Netzes übergeben werden, damit der Modellausgang bestimmt werden kann. Im Falle von MLP-Netzen sind dies die Gewichte, die in den Neuronen verwendet werden. Im Falle von lokalen Modellnetzen sind dies die Parameter der lokalen Modelle, die Zentren der Gaußfunktionen und die Skalierung. Bei der Verwendung anderer Partitionierungsverfahren müssen evtl. mehr Parameter übergeben werden. Dies können zum Beispiel unterschiedliche Werte der Standardabweichungen auf der Diagonalen der Kovarianz-Matrix sein.

Zum Abschluss wird der Berechnungsablauf des Simulationsmodells während eines Simulationszeitschritts k beschrieben, der auch in Abbildung 6.3 zu sehen ist. Dabei wird zunächst eine rekursive Berechnung der gefilterten Eingangssignale durchgeführt, bei denen die durch die Liniensuche optimalen Ordnungen und Pole verwendet werden (siehe Abbildung 5.5). Wichtig ist dabei, die Werte für die Normierung der gefilterten Signale zu verwenden, die bei der Modellerstellung verwendet wurden. Außerdem ist es wichtig, dass die gleiche Abtastzeit zur Filterung verwendet wird. Diese Signale werden anschließend als Eingangssignale im MLP-Netz oder im LM-Netz verwendet. Für die erstellten neuronalen Netze kann nun mit den ermittelten Modellparametern die Berechnung des jeweiligen Modellausgangs mit anschließender Rücktransformation auf den ursprünglichen Wertebereich durchgeführt werden. Dabei werden die Werte verwendet, die zur Normierung des Ausgangssignals bei der Modellerstellung genutzt wurden.
Die Reihenfolge zur Bestimmung der Modellausgänge der neuronalen Netze ist dabei wie folgt:

Zunächst wird das neuronale Netz zur Bestimmung des Drehmoments $T_{\mathsf{at}}(k)$ der Antriebswelle aufgerufen. Dieses wird dann im Modell des Fahrwerks und in den darauf folgenden Modellen genutzt, um die Winkelgeschwindigkeit $\omega_{\mathsf{at}}(k)$ zu berechnen. Anschließend wird das neuronale Netz zur Modellierung der Winkelgeschwindigkeit $\omega_{\mathsf{m}}(k)$ des Motors verwendet.
Werden noch zusätzliche Sensorsignale in der Simulation benötigt, werden die entsprechenden neuronalen Netze verwendet, um die entsprechenden Größen zum Simulationsschritt k zu bestimmen. Sind alle nötigen Größen bestimmt, ist der Simulationsschritt k beendet und es folgt der nächste Simulationsschritt $k+1$. Diese wird solange wiederholt, bis die vorgegebene Anzahl von Simulationsschritten erreicht ist.

6.6 Fazit

Als Fazit dieses Kapitels werden die gezeigten vier Schritte zur automatischen numerischen Vereinfachung detaillierte Getriebemodelle zusammengefasst.

1. Schritt: Erstellung und Aufteilung von Simulationsdaten zur Erstellung von neuronalen Netzen

Bei der Aufbereitung der Simulationsdaten ist zu beachten, dass diese mit einer konstanten Abtastzeit vorliegen. Diese entspricht dann auch der Abtastzeit der zeitdiskreten Laguerre-Filter. Um anschließend die automatische Modellerstellung durchführen zu können, werden die Simulationsdaten in Eingangssignale und Ausgangssignale aufgeteilt. Zudem werden die ersten 85% der Simulationsdaten als Trainingsdaten und als Validierungsdaten verwendet. Diese werden dabei in 90% für das Training und in 10% für die Validierung unterteilt. Die restlichen 15% der Simulationsdaten werden als Testdaten verwendet.

2. Schritt: Auswahl des neuronalen Netzes

MLP-Netze erlauben eine Modellierung der Schaltdynamik mit weniger Modellparametern und ermöglichen somit eine etwas größere numerische Vereinfachung. Allerdings ist der Aufwand bei der Erstellung von MLP-Netzen größer, da ein aufwändigeres Parameteroptimierungsverfahren verwendet werden muss. Steht die numerische Vereinfachung im Vordergrund, sollten deshalb MLP-Netze verwendet werden. Soll jedoch die Modellstellung mit weniger zeitlichen Aufwand erfolgen, ist es besser LM-Netze zu verwenden.

3. Schritt: Automatische Erstellung von neuronalen Netzen unter Verwendung der vorgestellten generischen Modellstrukturen

Für die automatische Erstellung der neuronalen Netze zur Modellierung der Schaltdynamik wurden generische Modellstrukturen vorgestellt. Diese unterscheiden sich darin, ob ein Drehmoment oder eine Winkelgeschwindigkeit modelliert werden soll. Für MLP-Netze sind diese in Abbildung 6.1 und Abbildung 6.4 und für LM-Netze in Abbildung 6.2 und Abbildung 6.5 zur Modellierung von Winkelgeschwindigkeiten bzw. Drehmomenten zu sehen.
Dabei müssen drei Ordnungen und drei Zeitkonstanten neben den jeweiligen Hyperparametern der Architektur optimiert werden. Diese werden durch Verwendung der im letzten Kapitel vorgestellten Liniensuche optimiert, bei der $N_O = 5$ für die Anzahl von Suchdurchläufen verwendet wird. Dazu werden für beide Netztypen die gleichen Suchräume für die Ordnungen und Zeitkonstanten, die in den Laguerre-Filtern zum Einsatz kommen, verwendet.

Diese sind gegeben durch

$$\begin{aligned}
\mathsf{SP}_{\eta_1} &= \begin{pmatrix} 0 & 2 & 4 & 6 & 8 & 10 & 12 \end{pmatrix} \\
\mathsf{SP}_{\eta_2} &= \begin{pmatrix} 0 & 2 & 4 & 6 & 8 & 10 & 12 \end{pmatrix} \\
\mathsf{SP}_{\eta_3} &= \begin{pmatrix} 0 & 2 & 4 & 6 & 8 & 10 & 12 \end{pmatrix} \\
\mathsf{SP}_{T_{\mathsf{K},1}} &= \text{linspace}\left(\frac{3}{20}, \frac{7}{10}, 7 \right) \\
\mathsf{SP}_{T_{\mathsf{K},2}} &= \text{linspace}\left(\frac{3}{20}, \frac{7}{10}, 7 \right) \\
\mathsf{SP}_{T_{\mathsf{K},3}} &= \text{linspace}\left(\frac{3}{20}, \frac{7}{10}, 7 \right).
\end{aligned} \tag{6.8}$$

Für MLP-Netze kommt zusätzlich ein Suchraum für die Anzahl M der Neuronen in der verdeckten Schicht hinzu, der gegeben ist durch

$$\mathsf{SP}_{\mathsf{M}} = \begin{pmatrix} 5 & 7 & 10 & 12 & 15 & 17 & 20 & 22 & 25 \end{pmatrix}. \tag{6.9}$$

Somit müssen bei der Verwendung von MLP-Netzen sieben Hyperparameter durch die Liniensuche optimiert werden.
Für LM-Netze kommen zusätzlich zwei Suchräume hinzu, deren Werte beim inkrementellen Clustering-Verfahren verwendet werden. Diese sind der minimale Abstand r_{min} zwischen zwei Zentren und die einheitlich verwendete Skalierung σ. Die Suchräume sind dabei gegeben durch

$$\begin{aligned}
\mathsf{SP}_{r_{\mathsf{min}}} &= \text{linspace}\left(\frac{1}{10}, \frac{2}{5}, 7 \right) \\
\mathsf{SP}_{\sigma} &= \text{linspace}\left(\frac{3}{20}, \frac{2}{5}, 7 \right).
\end{aligned} \tag{6.10}$$

Für das inkrementelle Clustering-Verfahren orientiert sich die maximal zulässige Anzahl von Modellparametern, die vorgegeben wird, an der Anzahl der Gangstufen. Es wird empfohlen, pro Gangstufe maximal etwa 300 Modellparameter zu verwenden. Weiter werden beim inkrementellen Clustering-Verfahren 12 Schritte verwendet, bis die maximale Anzahl von Modellparametern erreicht ist.
Zudem reicht es aus, für die Parameteroptimierung die Trainingsdaten mit einer Abtastzeit von $0,15[\mathrm{s}]$ zu verwenden. Dadurch kann die Dauer der Liniensuche erheblich verkürzt werden, wenn in der Simulation deutlich kleinere Abtastzeiten als $0,15[\mathrm{s}]$ verwendet werden sollen.

4. Schritt: Übergabe von Parametern an ein Simulationsmodell

Zur Verwendung der erstellten neuronalen Netze in einem Simulationsmodell werden folgende Parameter benötigt:

- Die Modellparameter $\mathbf{w}$ des statischen neuronalen Netzes
- Die bestimmten Hyperparameter, also die Ordnungen und Pole bzw. Zeitkonstanten der Laguerre-Filter
- Die Architekturparameter:
  - Bei LM-Netzen sind dies die Zentren der Aktivierungsfunktionen und die einheitliche Skalierung
  - Bei MLP-Netzen ist dies die Anzahl der Neuronen der verdeckten Schicht
- Die Information über die verwendete Abtastzeit in den Laguerre-Filtern, um die gefilterten Signale in jedem Simulationsschritt korrekt berechnen zu können
- Die Normierungsparameter, um die Eingangssignale auf Werte zwischen null und eins zu normieren und auf die ursprünglichen Werte zurückzutransformieren

7 Anwendung der Methode zur numerischen Vereinfachung an einem Beispiel

In diesem Kapitel wird gezeigt, wie die zuvor beschriebene Methode angewendet werden kann, um aus einem energieflussbasierten Getriebemodell ein signalflussorientiertes und numerisch vereinfachtes Getriebemodell abzuleiten. Dazu wird ein Modell eines Stufenautomaten mit neun Gängen verwendet, das in Software-in-the-Loop-Simulationen zur Funktionsentwicklung und Funktionsabsicherung verwendet wird (siehe Kapitel 3). Die Anwendung der Methode zur numerischen Vereinfachung an diesem Beispiel bietet dabei die Möglichkeit zur Validierung, da durch die hohe Anzahl von Gängen und damit auch hohe Anzahl von Schaltelementen einer der aktuell komplexesten Stufenautomaten vorliegt.

7.1 Bearbeitung von Simulationsdaten zur Modellerstellung

Der erste Schritt der Methode zur numerischen Vereinfachung besteht in der Erstellung von Simulationsdaten, die zur Modellerstellung verwendet werden. Zur Erzeugung der Simulationsdaten wird ein energieflussbasiertes Fahrzeugmodell verwendet, das ein detailliertes Modell eines Stufenautomaten mit neun Gängen enthält. Solch ein Modell wird zum Beispiel in einer SiL-Simulation verwendet, um die Funktionssoftware abzusichern, wie es in Kapitel 3 beschrieben wurde. Das dabei verwendete detaillierte Getriebemodell modelliert dabei

die Hydraulik und Mechanik, wie es in Kapitel 3 beschrieben ist. Um zwischen den neun Gängen wechseln zu können, werden sechs Schaltelemente verwendet. Zudem besitzt der Stufenautomat eine Wandlerüberbrückungskupplung und eine Möglichkeit zur Regelung des Arbeitsdrucks. Somit wird in der Simulation von Schaltvorgängen dieses Stufenautomaten der Verlauf der acht Ansteuerströme benötigt. Das Fahrzeugmodell besteht dabei aus einem Motormodell, einem Fahrwerkmodell und einem Fahrermodell und ist in Abbildung 7.1 zu sehen.

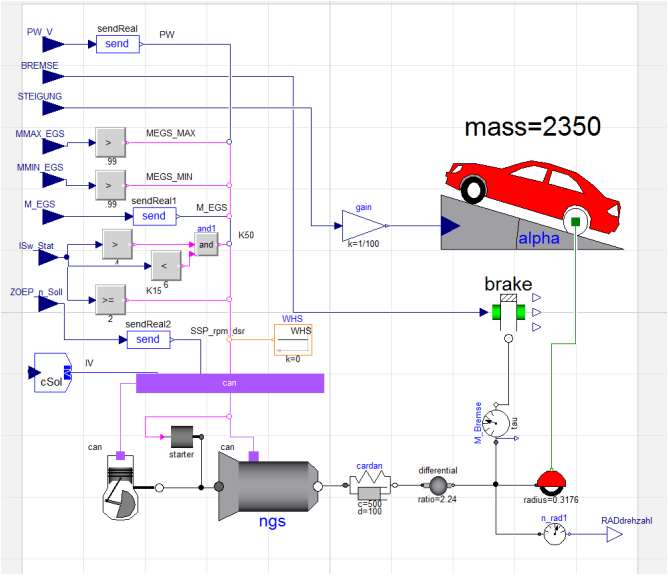


Abbildung 7.1: Dymola Modell zur Simulation von Schaltvorgängen eines Stufenautomaten mit neun Gängen

Diese Teilmodelle sind dabei folgendermaßen implementiert:[1]
Das Motormodell ist als kennfeldbasiertes Modell implementiert. Das Kennfeld gibt dabei den Zusammenhang zwischen Gaspedalstellung, Winkelgeschwin-

[1] Zu weiteren Einzelheiten zur Modellierung der Teilmodelle des Fahrzeugmodells wird auf Kapitel 2 verwiesen, in dem verschiedene Ansätze zur Modellierung beschrieben sind.

digkeit des Motors ω_m und Drehmoment T_m des Motors an. Zur Modellierung des Fahrwerks wird das Fahrzeug als Massenpunkt angenommen. Dieses Modell kann daher nur zur Untersuchung der Längsdynamik verwendet werden. Im Antriebsstrang wird ein Schwingungsfreiheitsgrad betrachtet, wozu ein Feder-Dämpfer-Element verwendet wird. Das Drehmoment an der Antriebswelle wird über einen starren Reifen auf die Straße übertragen. Außerdem kann der Fahrer über die Bremspedalstellung das Bremsmoment T_{brk} beeinflussen. Weiter werden der Rollwiderstand, die Hangabtriebskraft des Fahrzeugs und der Luftwiderstand berücksichtigt, die dem beschleunigenden Drehmoment entgegenwirken. Zur Bestimmung der Hangabtriebskraft ist die Steigung m_s der Strecke vorgegeben. Durch Bildung der Kräftebilanz wird dann das Fahrverhalten bestimmt. Das Fahrermodell besteht aus einem fest vorgeschriebenen zeitlichen Verlauf der Gaspedalstellung und der Bremspedalstellung. Auf diese Weise können unterschiedliche Fahrsituationen durchfahren werden, um das Schaltverhalten des Stufenautomaten in verschiedenen Betriebsmodi zu untersuchen.

Um die Schaltvorgänge zu untersuchen, wurde ein 3000[s] langes Fahrszenario verwendet, bei dem eine hohe Anzahl unterschiedlicher Schaltvorgänge enthalten ist. Wie in den Kapiteln 2 und 3 beschrieben ist, können dabei zur Simulation nur bestimmte numerische Simulationsverfahren verwendet werden. Eine Simulation dieses Fahrszenarios ist dabei zum Beispiel mit einer festen Schrittweite nur möglich, wenn diese kleiner 10^{-7}[s] ist, wodurch die Rechenzeit stark ansteigt und somit keine Echtzeitfähigkeit gegeben ist. Deshalb wird zur Simulation in Dymola [Dym] das Simulationsverfahren DASSL verwendet, das eine deutlich effizientere Simulation ermöglicht. Dieses eignet sich sowohl für das steife Differentialgleichungssystem, das durch die Modellierung der Mechanik und Hydraulik entsteht, als auch für die Unstetigkeiten, die durch die Modellierung der Reibung in den Schaltelementen entstehen. Das Verfahren verwendet dabei eine variable Schrittweite, wobei die Simulations-

schrittweite kurzzeitig auf etwa 10^{-12}[s] reduziert wird, um die Unstetigkeiten, die bei der Modellierung der Schaltelemente enthalten sind, aufzulösen. Mit der Auswahl des DASSL-Verfahrens zur Simulation beträgt die Rechenzeit etwa 500[s]. Allerdings ist durch die kurzzeitig sehr kleinen Simulationsschrittweiten keine direkte Verwendung in einer Echtzeitsimulation möglich. Die erstellten Simulationsdaten können für die zuvor beschriebene Methode zur numerischen Vereinfachung verwendet werden, da diese viele unterschiedliche Fahrmanöver enthalten, durch die das statische und dynamische Schaltverhalten des Stufenautomaten in vielen unterschiedlichen Betriebsmodi vorliegt. Zudem ist der gesamte Arbeitsbereich der am Stufenautomaten wirkenden Drehmomente und resultierenden Winkelgeschwindigkeiten in den Daten enthalten. Das bedeutet, dass der gesamte Bereich an möglichen Drehmomenten des Motors abgefahren wurde und somit das Fahrzeug möglichst viele Arbeitspunkte durchlaufen hat. Die Daten lagen dabei mit einer festen Abtastzeit von $T_\mathsf{S} = 0,01$[s] vor. Unter Verwendung der Dauer des Fahrszenarios ergaben sich also $\mathsf{N} = 300001$ Datenpunkte.
Zur Modellerstellung wurden nun diese Simulationsdaten in Eingangssignale und Ausgangssignale unterteilt, wie es in Gleichung (6.1) und Gleichung (6.2) beschrieben ist. Die Ansteuerströme wurden weiter in zwei Gruppen unterteilt, wie es in Kapitel 6 beschrieben ist. Die erste Gruppe $\mathbf{i}_{\mathsf{G,A},1}$ beinhaltet die Ströme zur Ansteuerung der Schaltelemente und besteht aus sechs Signalen. Die zweite Gruppe $\mathbf{i}_{\mathsf{G,A},2}$ besteht aus den Ansteuerströmen zum Öffnen und Schließen der WÜK und zur Regelung des Arbeitsdrucks und besteht daher aus zwei Signalen. Somit ergaben sich folgende Eingangssignale $\mathbf{u}(k)$ zur Erstellung der neuronalen Netze:

- Ansteuerströme der Schaltelemente

$$\mathbf{i}_{\mathsf{G,A1}}(k) = \begin{pmatrix} i_{\mathsf{G,A},1,1}(k) & i_{\mathsf{G,A},1,2}(k) & \dots & i_{\mathsf{G,A},1,6}(k) \end{pmatrix}$$

- Ansteuerströme der WÜK und zur Regelung des Arbeitsdrucks

$$\mathbf{i}_{\mathsf{G,A2}}(k) = \begin{pmatrix} i_{\mathsf{G,A,2,1}}(k) & i_{\mathsf{G,A,2,2}}(k) \end{pmatrix}$$

- Drehmoment $T_{\mathsf{m}}(k)$ des Motors
- Winkelgeschwindigkeit $\omega_{\mathsf{at}}(k)$ der Antriebswelle
- Drehmoment $T_{\mathsf{brk}}(k)$ der Bremse
- Steigung $\mathsf{m_s}(k)$ der Straße

Als Ausgangssignale werden hier nur das Drehmoment T_{at} der Antriebswelle und die Winkelgeschwindigkeit ω_{m} verwendet. Um die Struktur- und Parameteroptimierung zu ermöglichen, wurden die Simulationsdaten entsprechende der vorgestellten Methode aufgeteilt.
Im Folgenden werden die Ergebnisse der automatischen Modellerstellung vorgestellt, wobei zunächst die Modellierung mit MLP-Netzen und anschließend mit LM-Netzen vorgestellt wird. Bei der Vorstellung der Ergebnisse wird auf den Verlauf der Liniensuche eingegangen. Dazu wird der Verlauf der Fehler, die Anzahl von Modellparametern und der Verlauf der Hyperparameter betrachtet. Dabei wird neben dem Gesamtverlauf auch der Verlauf des jeweils neu ermittelten Minimums der Liniensuche dargestellt. Wie im letzten Kapitel beschrieben, stellen die drei Ordnungen und drei Zeitkonstanten der externen Laguerre-Filter der generischen Modellstrukturen zu optimierende Hyperparameter dar. Hinzu kommen die Hyperparameter der Architektur des verwendeten neuronalen Netzes (siehe Fazit aus Kapitel 6). Außerdem wird betrachtet, wie viel Zeit für die Modellerstellung benötigt wurde.

7.2 Modellierung der Schaltdynamik mit MLP-Netzen

In diesem Abschnitt wird auf die Modellierung mit MLP-Netzen eingegangen. Dazu werden zunächst die Ergebnisse der Liniensuche zur Modellierung des

Drehmoments der Antriebswelle behandelt und anschließend die Ergebnisse der Liniensuche zur Modellierung der Winkelgeschwindigkeit des Motors.

7.2.1 Modellierung des Drehmoments der Antriebswelle

Es wird zunächst auf die Ergebnisse der Liniensuche zur Modellierung des Drehmoments der Antriebswelle mit einem MLP-Netz eingegangen. In Abbildung 7.2 ist der Verlauf der Modellgüte zu sehen.

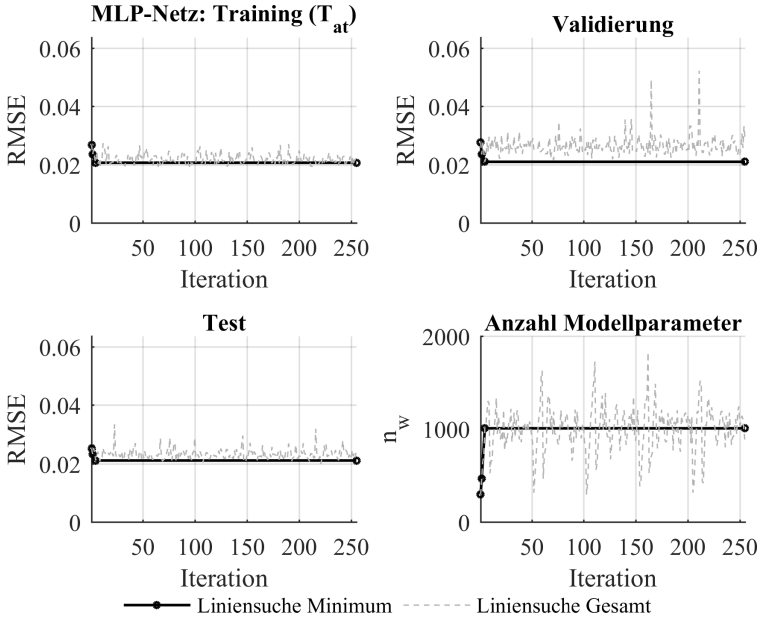


Abbildung 7.2: Modellgüte während der Liniensuche zur Modellierung des Drehmoments der Antriebswelle mit einem MLP-Netz

Dabei wird die Modellgüte durch den Verlauf des Trainings-, des Validierungs- und des Testfehlers beschrieben. Zusätzlich ist der Verlauf der Anzahl der Mo-

dellparameter als Maß für die Modellkomplexität zu sehen. Dieser Abbildung ist zu entnehmen, dass das Minimum der Liniensuche bereits nach 5 Iterationen gefunden wird und, dass hier die Wahl der Hyperparameter die Modellgüte nicht so stark beeinflusst. Um dies weiter zu untersuchen, ist in Abbildung 7.3 der Verlauf der Hyperparameter während der Liniensuche dargestellt. Dabei ist zu sehen, welche Werte der Hyperparameter im Gesamtverlauf bei der jeweiligen Iteration der Liniensuche gewählt wurden und welche dieser Werte zu einem neuen Minimum der Liniensuche führten. Es zeigt sich, dass das optimale Modell zur Modellierung des Drehmoments der Antriebswelle im Wesentlichen durch die Optimierung der Anzahl der Neuronen der verdeckten Schicht geschieht. Zudem sind in diesem Beispiel die gewählten Startwerte der Hyperparameter der Laguerre-Filter sehr nah an den optimalen Werten, was sich zusätzlich positiv auf das schnelle Konvergenz-Verhalten der Liniensuche auswirkt.
Die Ergebnisse der Liniensuche sind in Tabelle 7.1 zusammengefasst. Der Tabelle ist zu entnehmen, dass für den gesamten Durchlauf der Liniensuche knapp 14[h] benötigt wurden, was vergleichsweise hoch ist und auf die nichtlineare Optimierung der Modellparameter der MLP-Netze durch das Levenberg-Marquardt-Verfahren zurückzuführen ist. Dabei wurde durch die Ausdünnungen der Trainingsdaten die Dauer der Liniensuche um den Faktor drei verkürzt, ohne dass eine Verschlechterung der Modellgüte zu verzeichnen war. Es fällt außerdem bei Betrachtung der Modellgüte auf, dass die ermittelten Fehler sehr nah beieinander liegen und somit eine Überanpassung ausgeschlossen werden kann. Somit liefert die Strukturoptimierung durch die verwendete in diesem Fall ein valides neuronales Netz.

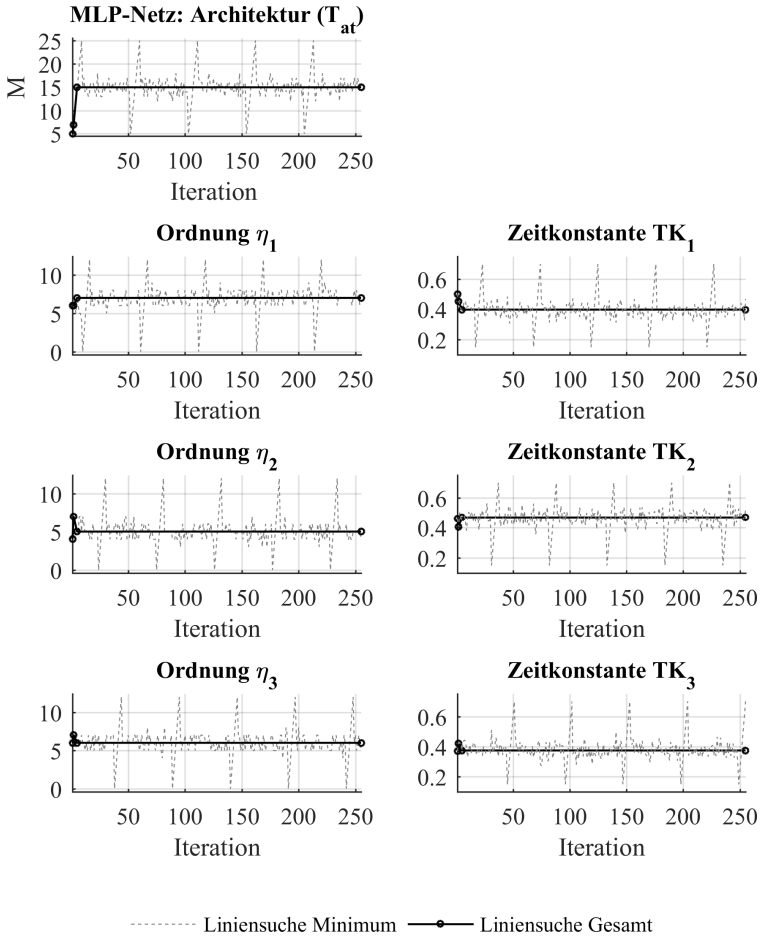


Abbildung 7.3: Hyperparameter während der Liniensuche zur Modellierung des Drehmoments der Antriebswelle mit einem MLP-Netz

Iterationen Liniensuche	255	Ermittelte Hyperparameter	
Dauer Liniensuche	$t = 13,7$[h]	Ordnungen	$\mathsf{BP}_\eta = \begin{pmatrix} 7 & 5 & 6 \end{pmatrix}$
Trainingsfehler	$\mathrm{RMSE}_\mathsf{T} = 0,0207$	Zeitkonstanten	$\mathsf{BP}_{T_\mathsf{K}} = \begin{pmatrix} 0,398 & 0,469 & 0,374 \end{pmatrix}$ [s]
Validierungsfehler	$\mathrm{RMSE}_\mathsf{V} = 0,0210$	Anzahl Neuronen	$\mathsf{BP}_\mathsf{M} = 15$
Testfehler	$\mathrm{RMSE}_\mathsf{TS} = 0,0211$	Anzahl Modellparameter	$n_\mathsf{w} = 1006$

Tabelle 7.1: Ergebnisse der Liniensuche mit einem MLP-Netz zur Modellierung des Drehmoments der Antriebswelle

7.2.2 Modellierung der Winkelgeschwindigkeit des Motors

In diesem Abschnitt wird das Ergebnis der Liniensuche zur Modellierung der Winkelgeschwindigkeit des Motors mit einem MLP-Netz vorgestellt. In Abbildung 7.4 ist der Verlauf der Modellgüte zu sehen. Wie zuvor ist der Verlauf der Anzahl der Modellparameter als Maß für die Modellkomplexität zu sehen. Hier wird bei Iteration 174 der Liniensuche das Modell mit dem kleinsten Validierungsfehler gefunden. Weiter ist zu erkennen, dass nach Iteration 24 nur noch kleine Verbesserungen der Modellgüte möglich sind. Bei Betrachtung des Testfehlers ist zu sehen, dass dieser nicht mit dem gefundenen Minimum der Liniensuche ansteigt, so dass hier keine Überanpassung vorliegt. Beim Gesamtverlauf der Liniensuche fallen immer wieder vereinzelte, vergleichsweise schlechte Modellgüten ($\mathrm{RMSE}_\mathsf{V} > 0.07$) beim Validierungs- und Testfehler auf. Allerdings werden diese nicht durch eine zu hohe Anzahl von Modellparametern hervorgerufen.

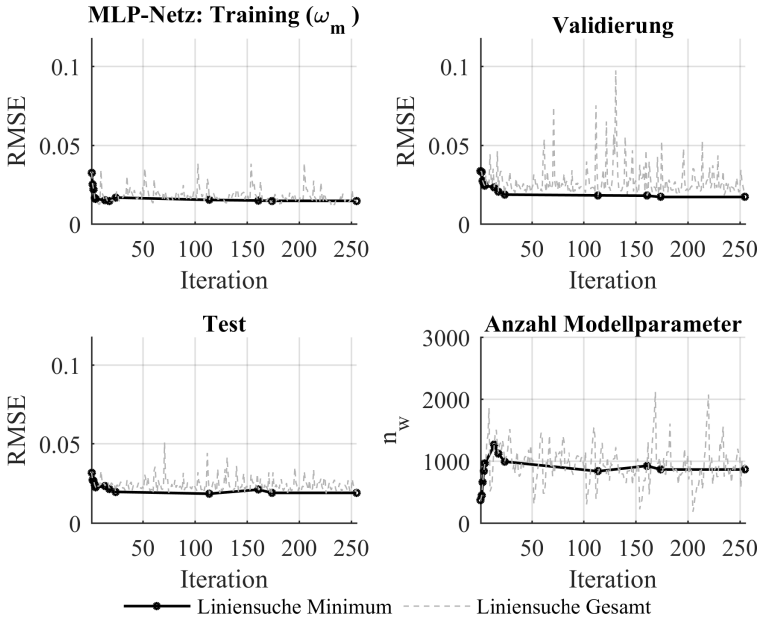


Abbildung 7.4: Modellgüte während der Liniensuche zur Modellierung der Winkelgeschwindigkeit des Motors mit einem MLP-Netz

Dies zeigt, dass hier die Wahl der Ordnungen und Zeitkonstanten einen stärkeren Einfluss auf die Modellgüte hat als bei der Modellierung des Drehmoments der Antriebswelle.

Um dies weiter zu untersuchen, ist in Abbildung 7.5 der Verlauf der Hyperparameter während der Liniensuche zu sehen. Bei Betrachtung des Verlaufs der Hyperparameter während der Liniensuche fällt auf, dass kleinere Ordnungen der Laguerre-Filter zu einer besseren Modellgüte führen. Dadurch wird die Anzahl der Eingangssignale des MLP-Netzes geringer, und somit kann eine höhere Anzahl von Neuronen in der verdeckten Schicht verwendet werden, so dass dabei sogar die Anzahl der Modellparameter des MLP-Netzes etwas verkleinert wird. Besonders auffällig ist dabei, dass hohe Ordnungen η_2 die

Modellgüte stark verschlechtern, wodurch die zuvor genannte Verschlechterung der Modellgüte erklärt werden kann, die teilweise auftritt. Bei der Ordnung η_1 wird beim letzten Schritt durch eine Verdopplung der Zeitkonstanten $T_{\mathrm{K},1}$ eine bessere Modellgüte erzielt. Gleichzeitig wird dabei die Ordnung η_1 von 4 auf 3 reduziert. Weiter fällt auf, dass erst am Ende die optimalen Zeitkonstanten alle in der gleichen Größenordnung von etwa $0,5[\mathrm{s}]$ liegen.
Die Ergebnisse der Liniensuche sind in Tabelle 7.2 zusammengefasst. Der Tabelle ist zu entnehmen, dass die Dauer der Liniensuche hier deutlich höher ist als zur Erstellung der MLP-Netze zur Modellierung des Drehmoments der Antriebswelle. Der Grund ist dabei im Konvergenz-Verhalten des Levenberg-Marquardt-Verfahrens zu finden, da hier pro Iteration der Liniensuche etwas mehr Zeit benötigt wurde. Dies summiert sich entsprechend über den Gesamtverlauf der Liniensuche auf. Insgesamt hat das erstellte MLP-Netz weniger Modellparameter (etwa 150) als das MLP-Netz zur Modellierung des Drehmoments der Antriebswelle.

Iterationen Liniensuche	255	Ermittelte Hyperparameter	
Dauer Liniensuche	$t = 19,8[\mathrm{h}]$	Ordnungen	$\mathsf{BP}_{\eta} = \begin{pmatrix} 3 & 0 & 6 \end{pmatrix}$
Trainingsfehler	$\mathrm{RMSE}_{\mathsf{T}} = 0,0147$	Zeitkonstanten	$\mathsf{BP}_{T_{\mathsf{K}}} = \begin{pmatrix} 0,517 & 0,417 & 0,509 \end{pmatrix}[\mathrm{s}]$
Validierungsfehler	$\mathrm{RMSE}_{\mathsf{V}} = 0,0171$	Anzahl Neuronen	$\mathsf{BP}_{\mathsf{M}} = 24$
Testfehler	$\mathrm{RMSE}_{\mathsf{TS}} = 0,191$	Anzahl Modellparameter	$n_{\mathsf{w}} = 865$

Tabelle 7.2: Ergebnisse der Liniensuche mit einem MLP-Netz zur Modellierung der Winkelgeschwindigkeit des Motors

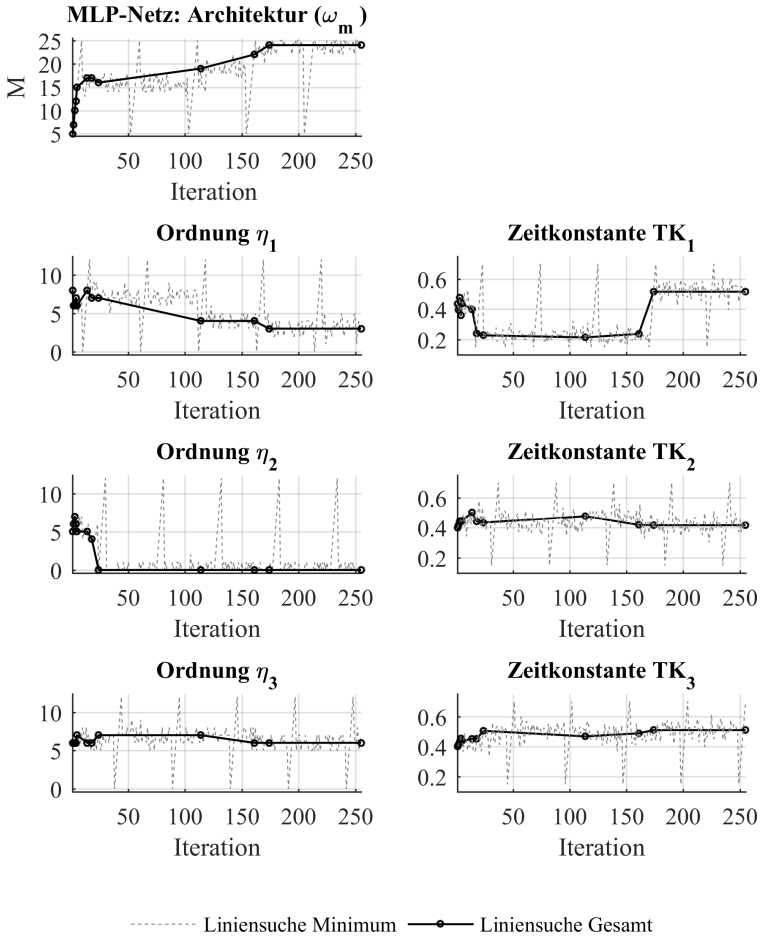


Abbildung 7.5: Hyperparameter während der Liniensuche zur Modellierung der Winkelgeschwindigkeit des Motors mit einem MLP-Netz

7.3 Modellierung der Schaltdynamik mit LM-Netzen

In diesem Abschnitt werden die Ergebnisse der Liniensuche vorgestellt, die bei Verwendung von LM-Netzen erhalten wurden. Dabei wird zunächst die Modellierung des Drehmoments der Antriebswelle und anschließend die Modellierung der Winkelgeschwindigkeit des Motors betrachtet.

7.3.1 Modellierung des Drehmoments der Antriebswelle

Im Folgenden wird auf das Ergebnis der Liniensuche zur Modellierung des Drehmoments der Antriebswelle mit einem LM-Netz eingegangen. In Abbildung 7.6 ist der Verlauf der Modellgüte zu sehen. Es zeigt sich, dass bei der Liniensuche nach etwa 20 Iterationen kaum eine weitere Verbesserung beim Validierungsfehler eintritt. Zudem wurden wie zuvor zu Beginn die größten Verbesserungen beim Validierungsfehler erzielt. Bei Betrachtung der Anzahl der Modellparameter wird klar, dass diese bis zu Iteration 230 reduziert werden kann. Vom vorletzten gefundenen zum letzten gefunden Minimum steigt die Anzahl der Modellparameter jedoch wieder deutlich an. Allerdings wurde dadurch der Testfehler verbessert, so dass keine Überanpassung vorliegt. Weiter fällt bei Betrachtung des Gesamtverlaufs der Anzahl der Modellparameter auf, dass diese stark streuen. Dies liegt daran, dass das inkrementelle Clustering-Verfahren früher abgebrochen wurde, weil keine lokalen Modelle mehr hinzugefügt werden konnten oder weil der Validierungsfehler durch Hinzufügen lokaler Modelle wieder angestiegen ist.

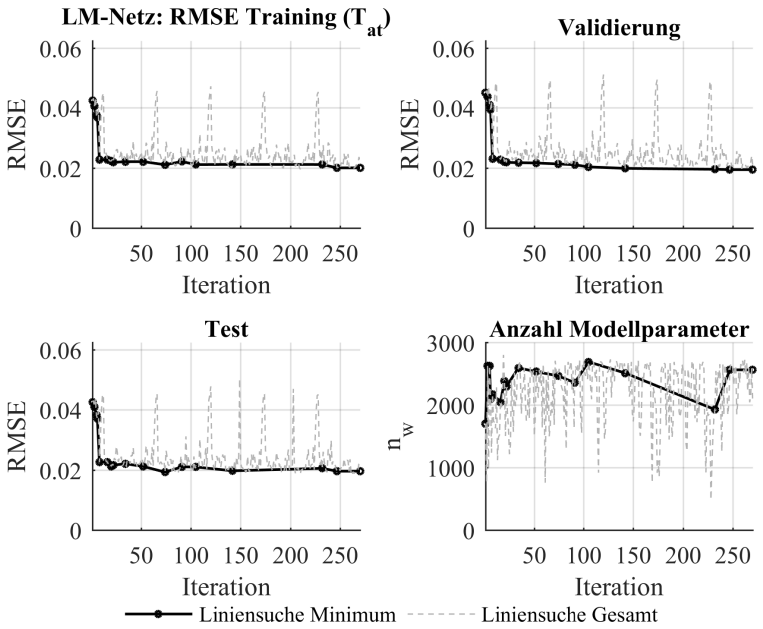


Abbildung 7.6: Verlauf der Modellgüte während der Liniensuche zur Modellierung des Drehmoments der Antriebswelle mit einem LM-Netz

In Abbildung 7.7 ist der Verlauf der Hyperparameter zur Modellierung des Drehmoments der Antriebswelle mit einem LM-Netz zu sehen. Dabei ist festzustellen, dass für die beiden Hyperparameter r_{min} und σ des inkrementellen Clustering-Verfahrens bereits nach wenigen Iterationen die optimalen Werte gefunden wurden und sich anschließend kaum verändern. Weiter muss beachtet werden, dass nur die erste Ordnung der gefilterten Signale zur Partitionierung verwendet wurden und die restlichen Ordnungen nur in den linearen Modellen berücksichtigt werden.

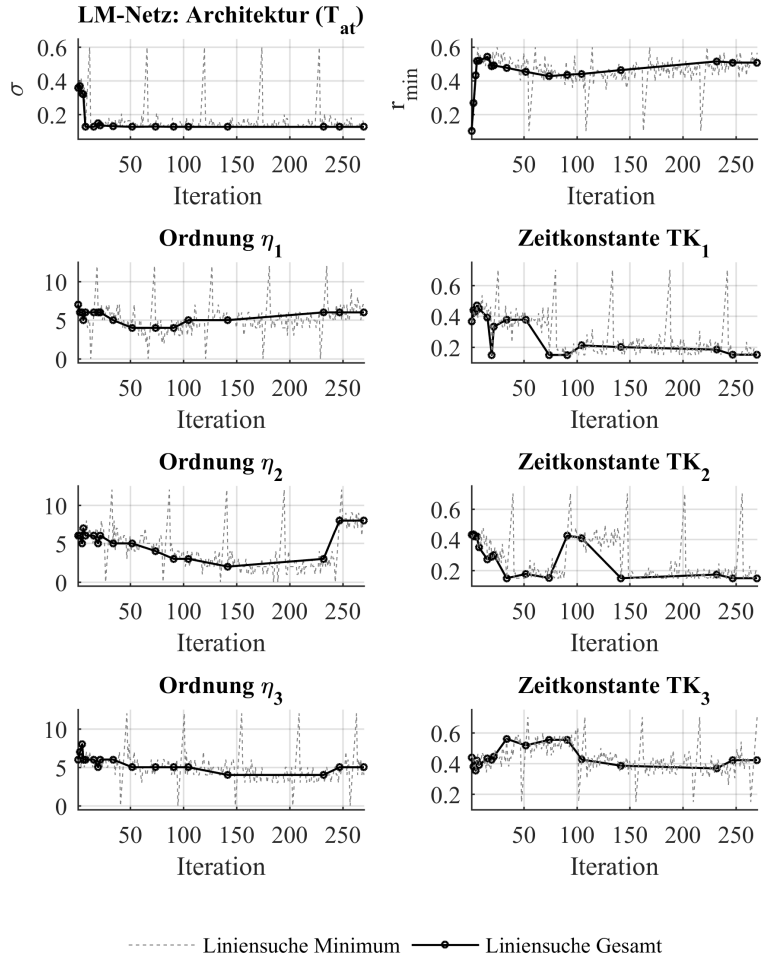


Abbildung 7.7: Hyperparameter während der Liniensuche zur Modellierung des Drehmoments der Antriebswelle mit einem LM-Netz

Beim Verlauf der Hyperparameter der Laguerre-Filter fällt auf, dass die optimalen Werte immer wieder stark springen. Allerdings führt dies nach Iteration 20 kaum zu weiteren Verbesserungen. Dies bedeutet, dass es in diesem Beispiel mehrere lokale Minima gibt, die von der Liniensuche gefunden wurden.
Die Ergebnisse der Liniensuche zur Erstellung eines LM-Netzes zur Modellierung des Drehmoments der Antriebswelle sind in Tabelle 7.3 zusammengefasst.

Iterationen Liniensuche	270	Ermittelte Hyperparameter	
Dauer Liniensuche	$t = 2,0[\mathrm{h}]$	Ordnungen	$\mathrm{BP}_{\eta} = \begin{pmatrix} 6 & 8 & 5 \end{pmatrix}$
Trainingsfehler	$\mathrm{RMSE}_{\mathrm{T}} = 0,0201$	Zeitkonstanten	$\mathrm{BP}_{T_{\mathrm{K}}} = \begin{pmatrix} 0,151 & 0,150 & 0,421 \end{pmatrix}$ [s]
Validierungsfehler	$\mathrm{RMSE}_{\mathrm{V}} = 0,0195$	Anzahl lokale Modele	$\mathrm{BP}_{\mathrm{M}} = 40$
Testfehler	$\mathrm{RMSE}_{\mathrm{TS}} = 0,0196$	Anzahl Modellparameter	$n_{\mathrm{w}} = 2560$
		Clustering	$\mathrm{BP}_{r_{\mathrm{min}}} = 0,51$ $\mathrm{BP}_{\sigma} = 0,13$

Tabelle 7.3: Ergebnisse der Liniensuche mit einem LM-Netz zur Modellierung des Drehmoments der Antriebswelle

Bei Betrachtung der in der Tabelle gegebenen Werte der Modellgüte fällt auf, dass Trainings-, Validierungs- und Testfehler nahezu identisch sind und nahe bei $0,02$ liegen. Dies lässt den Schluss zu, dass hier die Strukturoptimierung besonders gut geklappt hat. Die Dauer der Liniensuche ist mit $2,4[\mathrm{h}]$ deutlich kürzer als die zur Erstellung eines MLP-Netzes, obwohl hier sogar die Anzahl der Iterationen der Liniensuche etwas höher ist. Der Grund dafür ist, dass das inkrementelle Clustering-Verfahren weniger rechenintensiv ist als

das Levenberg-Marquardt-Verfahren. Weiter fällt auf, dass die Ordnungen der Laguerre-Filter hier in der gleichen Größenordnung wie zuvor liegen. Allerdings werden hier unterschiedliche optimale Zeitkonstanten ermittelt.

7.3.2 Modellierung der Winkelgeschwindigkeit des Motors

Im Folgenden wird das Ergebnis der Liniensuche zur Modellierung der Winkelgeschwindigkeit des Motors mit einem LM-Netz vorgestellt. In Abbildung 7.8 ist dazu der Verlauf der Modellgüte zu sehen. Hier ist nach etwa Iteration 90 kaum eine weitere Verbesserung des Validierungsfehlers zu beobachten. Der

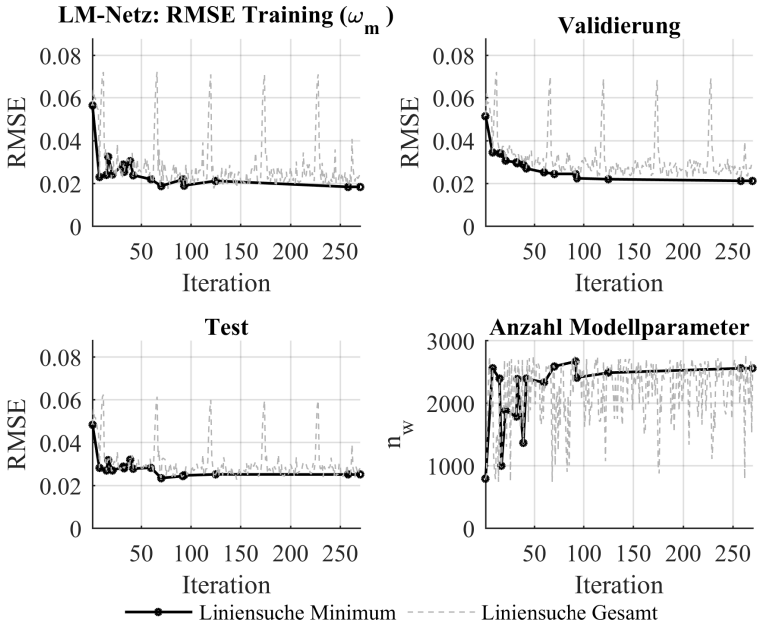


Abbildung 7.8: Modellgüte während der Liniensuche zur Modellierung der Winkelgeschwindigkeit des Motors mit einem LM-Netz

Verlauf des Testfehlers zeigt, dass dieser beim letzten gefundenen Minimum der Liniensuche nicht ansteigt, somit ist hier keine Überanpassung gegeben. Es fällt jedoch auf, dass der Testfehler etwas höher ist als die Trainings- und Validierungsfehler. Zudem fällt beim Verlauf aller Fehler auf, dass es immer wieder Ausreißer gibt, bei denen sich eine deutlich schlechtere Modellgüte ergibt. Weiter ist bei Betrachtung der Anzahl der Modellparameter zu erkennen, dass diese bei den anfangs gefundenen Minima der Liniensuche sehr stark variieren.

Um dies weiter untersuchen zu können, ist in Abbildung 7.9 ist der Verlauf der Hyperparameter während der Liniensuche zu sehen. Aus dem Verlauf ist zu erkennen, dass am Anfang der Liniensuche ein deutlich anderer Wert für den besten kleinsten Abstand r_{min} für das inkrementelle Clustering-Verfahren ermittelt wurde. Erst bei etwa Iteration 100 wurde die Größenordnung der am Ende der Liniensuche gefundenen optimalen Werte erreicht. Im Gegensatz zur Modellierung des Drehmoments der Antriebswelle konvergieren hier auch die Hyperparameter der Laguerre-Filter auf bestimmte Werte. Das bedeutet, dass hier in ein lokales Minimum vorliegt, das durch die Liniensuche gefunden wurde. Allerdings wurden hierdurch kaum signifikante Verbesserungen der Modellgüte erzielt.

Die Ergebnisse der Liniensuche zur Erstellung des LM-Netzes zur Modellierung der Winkelgeschwindigkeit des Motors sind in Tabelle 7.4 zusammengefasst. Zunächst fällt auf, dass hier die Anzahl der lokalen Modelle etwa so groß ist wie bei der Modellierung des Drehmoments der Antriebswelle mit einem LM-Netz. Die Dauer der Liniensuche ist hier etwas geringer. Zudem sind hier für das inkrementelle Clustering-Verfahren nahezu die gleichen optimalen Werte für r_{min} und σ gefunden worden wie zuvor.

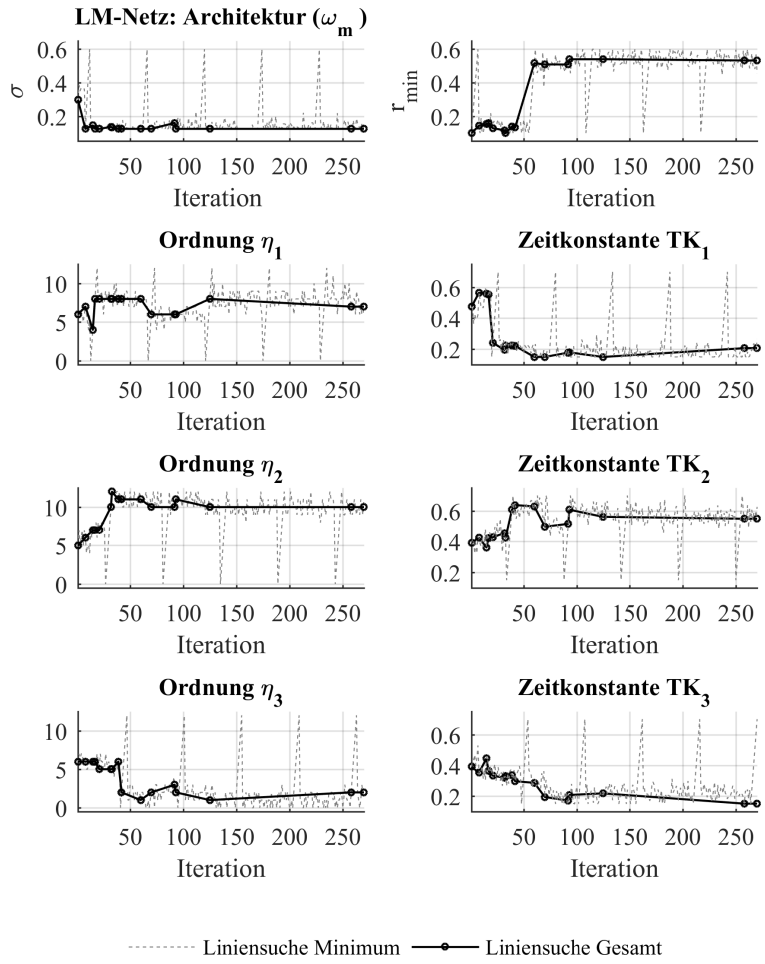


Abbildung 7.9: Hyperparameter während der Liniensuche zur Modellierung der Winkelgeschwindigkeit des Motors mit einem LM-Netz

Iterationen Liniensuche	270	Ermittelte Hyperparameter	
Dauer Liniensuche	$t = 2,0$[h]	Ordnungen	$\mathsf{BP}_{\eta} = \begin{pmatrix} 7 & 10 & 2 \end{pmatrix}$
Trainingsfehler	$\mathrm{RMSE}_{\mathsf{T}} = 0,0184$	Zeitkonstanten	$\mathsf{BP}_{T_{\mathsf{K}}} = \begin{pmatrix} 0,207 & 0,550 & 0,152 \end{pmatrix}$ [s]
Validierungsfehler	$\mathrm{RMSE}_{\mathsf{V}} = 0,0212$	Anzahl lokale Modele	$\mathsf{BP}_{\mathsf{M}} = 37$
Testfehler	$\mathrm{RMSE}_{\mathsf{TS}} = 0,0251$	Anzahl Modellparameter	$n_{\mathsf{w}} = 2553$
		Clustering	$\mathsf{BP}_{r_{\mathsf{min}}} = 0,53$ $\mathsf{BP}_{\sigma} = 0,13$

Tabelle 7.4: Ergebnisse der Liniensuche mit einem LM-Netz zur Modellierung der Winkelgeschwindigkeit des Motors

7.4 Vergleich der erstellten neuronalen Netze

In diesem Abschnitt werden die Ergebnisse der Erstellung der beiden Typen neuronaler Netze verglichen. Dazu wird zunächst auf die Modellgüte und den Verlauf der Liniensuche eingegangen. Der Vergleich von LM-Netz und MLP-Netz zur Modellierung der jeweiligen Ausgangssignale ergibt folgende Ergebnisse:

- Die Dauer der Liniensuche zur Erstellung von MLP-Netzen ist deutlich größer. Das liegt an dem rechenaufwändigeren Levenberg-Marquardt-Verfahren zur Parameteroptimierung der MLP-Netze. Allerdings konvergiert bei beiden Netzen die Liniensuche sehr schnell, so dass nach wenigen Iterationen bereits die Größenordnungen des Minimums des am Ende gefundenen Validierungsfehlers erreicht werden. Bei späteren Iterationen werden nur noch kleinere Modellverbesserungen erzielt.

- Die Anzahl der benötigten Modellparameter von MLP-Netzen ist geringer (etwa um den Faktor 2). Dies liegt daran, dass durch MLP-Netze eine stärkere Komprimierung hochdimensionaler Eingangsräume stattfindet. Dies macht sich hier bei der Anzahl der benötigten Modellparameter bemerkbar.
- Die Modellgüte ist bei beiden Netzen vergleichbar, wobei der RMSE in der Größenordnung von $0,02$ liegt. Bei der Modellierung der Winkelgeschwindigkeit des Motors schneidet jedoch das MLP-Netz etwas besser ab, da sowohl ein kleinerer Testfehler als auch ein kleinerer Validierungsfehler ermittelt wurde.
- Es werden bei der Liniensuche unterschiedliche optimale Hyperparameter für die Laguerre-Filter ermittelt. Dies liegt daran, dass bei den LM-Netzen nur gefilterte Signale bis zur ersten Ordnung der Laguerre-Filter im $\mathbf{z}$-Regressor verwendet werden und somit höhere Ordnungen nur linear berücksichtigt werden. Somit kommt es bei der Liniensuche zu unterschiedlichen optimalen Hyperparametern.

Als nächstes werden die Signalverläufe der erstellten neuronalen Netze diskutiert. Dazu wird jeweils ein Ausschnitt aus den Trainings-, den Validierungs- und den Testdaten gezeigt und mit dem Referenzverlauf verglichen. Dabei wird im Vergleich zur Gesamtdauer der Simulation nur eine kleine Anzahl von Schaltvorgängen vorgestellt. Bei der Auswahl der Ausschnitte wurde deshalb darauf geachtet, dass eine möglichst repräsentative Auswahl verschiedener Schaltvorgänge zu sehen ist.

7.4.1 Modellierung des Drehmoments der Antriebswelle

In Abbildung 7.10 ist der Verlauf des Drehmoments der Antriebswelle zu sehen. Der Referenzverlauf ist gestrichelt gezeichnet und der Verlauf des erstellten MLP-Netzes und LM-Netzes mit durchgezogenen Linien.

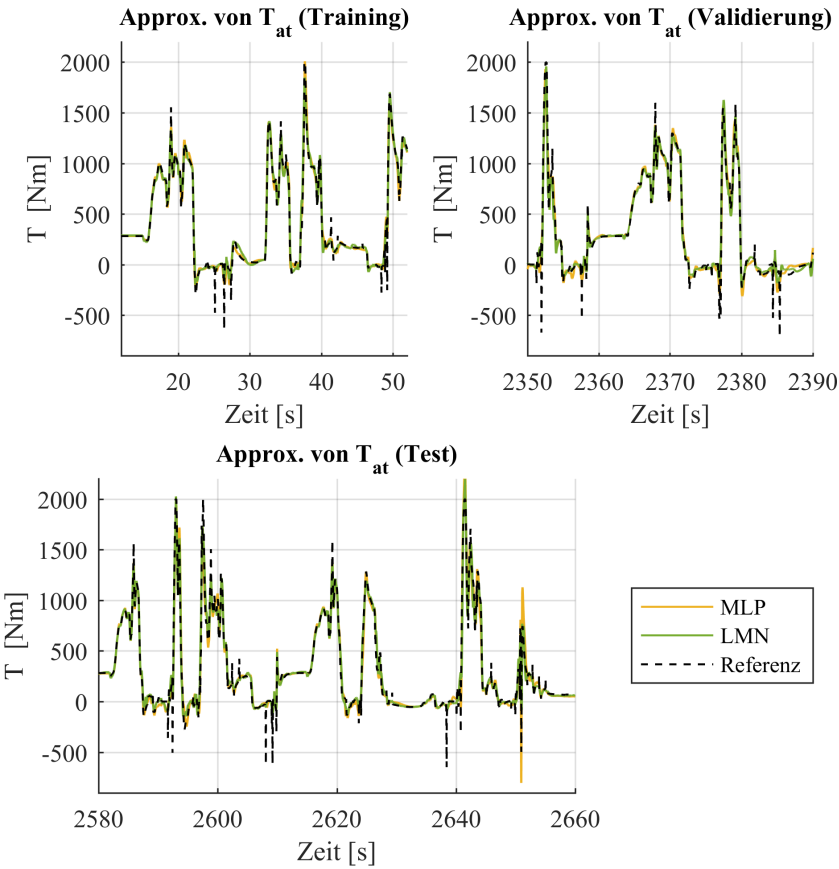


Abbildung 7.10: Drehmoments der Antriebswelle mit dem MLP-Netz und dem LM-Netz

Es zeigt sich, dass das Drehmoment der Antriebswelle durch beide neuronale Netze sehr gut approximiert wird. Abweichungen gibt es bei beiden neuronalen Netzen, wenn das Drehmoment der Antriebswelle kurzzeitig stark negativ wird und bremsend auf das Fahrzeug wirkt. Der dabei entstehende Ruck kann in einer Simulation daher nur durch das detaillierte Modell berücksichtigt werden.

Allerdings ist zu erwarten, dass die Auswirkungen dieser Spitzen beim Drehmoment der Antriebswelle in einem Fahrzeugmodell, wie es oben beschrieben ist, kaum bemerkbar sind, da das Schwingungsverhalten nicht bewertet wird. Zudem besitzt das Fahrzeug eine hohe Trägheit, so dass schnelle Änderungen des Drehmoments sich nur stark gedämpft bemerkbar machen.
Bei den Testdaten fällt auf, dass es beim MLP-Netz bei etwa 2650[s] zu einem kurzzeitigen Überschwingen kommt. Dies sollte sich jedoch wegen des Dämpfungsverhaltens des Fahrzeugs kaum bemerkbar machen. Daher ist zu erwarten, dass diese Approximation kaum Abweichungen in einem Simulationsmodell zur Folge hat.
Auf die Verwendung der erstellten neuronalen Netze in einer Simulation wird später in diesem Kapitel noch näher eingegangen.

7.4.2 Modellierung der Winkelgeschwindigkeit des Motors

In Abbildung 7.11 ist der Verlauf der Winkelgeschwindigkeit des Motors zu sehen. Dabei ist der Referenzverlauf wie zuvor gestrichelt gezeichnet und der Verlauf des erstellten MLP-Netzes und des LM-Netzes mit durchgezogenen Linien. Die gezeigte Winkelgeschwindigkeit des Motors ergibt sich aus der Winkelgeschwindigkeit der Antriebswelle und wird entsprechend übersetzt. Ein Vergleich der Referenz mit den beiden erstellten neuronalen Netzen der Liniensuche zeigt, dass diese den Verlauf sehr gut approximieren. Abweichungen gibt es, wenn der Motor niedrige Winkelgeschwindigkeiten annimmt. Zudem wird deutlich, dass in den gezeigten Ausschnitten der Daten beim MLP-Netz weniger Abweichungen von der Referenz zu sehen sind als beim LM-Netz. Somit machen sich die kleineren Modellfehler auch optisch bei Betrachtung der Signalverläufe bemerkbar.

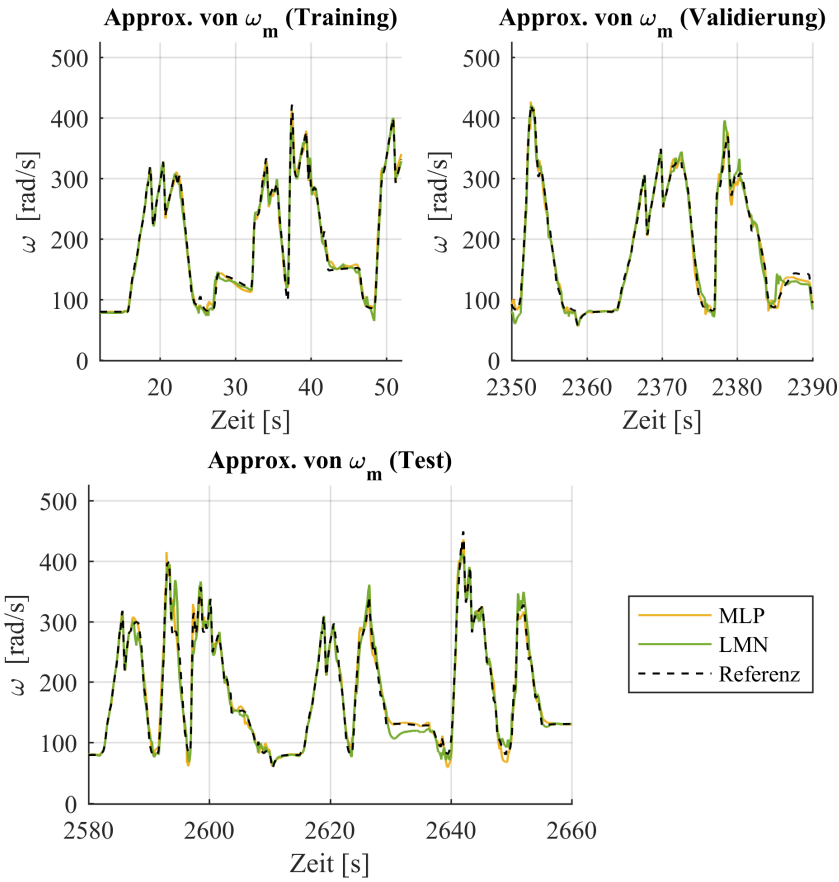


Abbildung 7.11: Winkelgeschwindigkeit des Motors mit dem MLP-Netz und dem LM-Netz

Da bei alleiniger Betrachtung der Winkelgeschwindigkeit des Motors die Information darüber fehlt, welche Winkelgeschwindigkeit die Antriebswelle hat, ist in Abbildung 7.12 der Verlauf der Übersetzung zwischen der Winkelgeschwindigkeit des Motors und der Antriebswelle zu sehen. Der dargestellte Verlauf

der Übersetzung zeigt an, wann das Fahrzeug im Stillstand ist. Dies ist der Fall, wenn die Übersetzung null ist, wobei sich der Motor im Leerlauf befinden kann. Weiter ist zu erkennen, dass die Übersetzungen überwiegend gut approximiert werden.

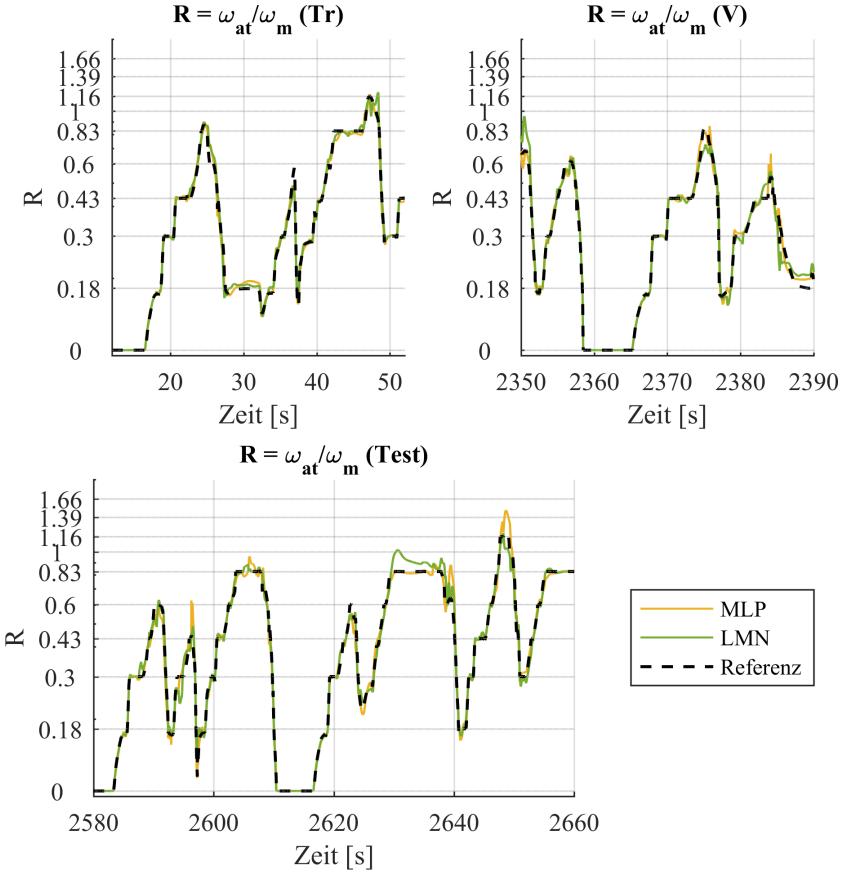


Abbildung 7.12: Übersetzung bei dem MLP-Netz und LM-Netz

Allerdings gibt es bei einer statischen Übersetzung zwischen etwa 2630[s] und 2640[s] eine größere Abweichung vor allem beim LM-Netz. Zudem gibt es Abweichungen, wenn sich die Übersetzungsverhältnisse schnell ändern. Dabei ist auch zu sehen, dass das MLP-Netz zum Überschwingen neigt, wie beispielsweise die Testdaten bei etwa 2595[s] und 2645[s] zeigen. Die Auswirkungen solcher Abweichungen werden im nächsten Abschnitt diskutiert, wenn die Ergebnisse der Simulation der Schaltdynamik mit den erstellten neuronalen Netzen vorgestellt werden.

7.5 Simulation der Schaltvorgänge des Stufenautomaten

In diesem Abschnitt werden die Ergebnisse vorgestellt, die bei der Verwendung der erstellten neuronalen Netze in einem signalflussorientierten Modell des Fahrzeugs verwendet wurden. Dazu wird zunächst auf den Aufbau des dazu verwendeten signalflussorientierten Fahrzeugmodells eingegangen und wie die erstellten neuronalen Netze in diesem verwendet wurden.

7.5.1 Signalflussorientiertes Fahrzeugmodell zur Validierung der erstellten neuronalen Netze

Zur Validierung der erstellten neuronalen Netze wird ein in Matlab/Simulink [Mat] implementiertes signalflussorientiertes Modell des Fahrzeugs verwendet, das einen möglichst ähnlichen Detaillierungsgrad hat, wie das in Dymola. Dies bedeutet, dass das Fahrermodell, das Motormodell und das Modell des Fahrwerks einen ähnlichen Detaillierungsgrad haben. Dazu musste im Wesentlichen ein geeignetes Fahrwerkmodell erstellt werden, das in Abbildung 7.13 zu sehen ist. Dieses wird dann in Kombination mit dem in Abbildung 6.3 gezeigten Blockschaltbild verwendet. Im Fahrwerkmodell werden Hangabtriebskraft, Luftreibung und Reibung auf der Straße berücksichtigt.

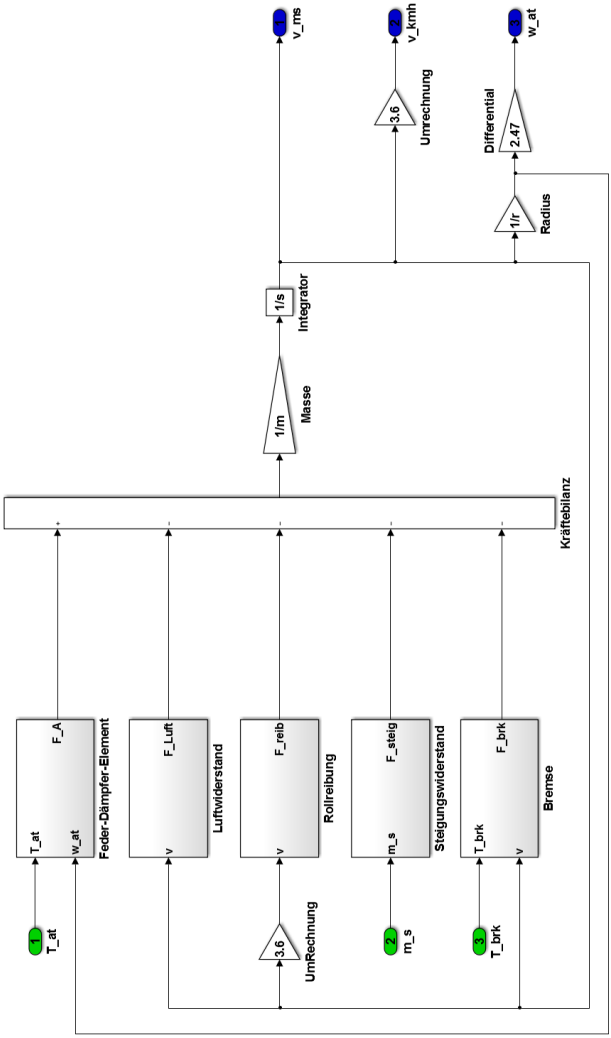


Abbildung 7.13: Aufbau des verwendeten Fahrwerkmodells in Simulink

Zusätzlich wird ein Schwingungsfreiheitsgrad des Fahrzeugs durch ein Feder-Dämpfer-Element abgebildet, das sich zwischen Ersatzträgheit des gesamten Getriebes und der Trägheit der Antriebswelle befindet. Zusammenfassend wurde das Fahrzeugmodell folgendermaßen implementiert:

- Das Modell berücksichtigt nur die Längsdynamik, und das Fahrzeug wird als Punktmasse angenommen.
- Die Reifen werden starr modelliert, somit wird das am Reifen angreifende Drehmoment ideal auf die Straße übertragen.
- Es wird die statische Übersetzung der Winkelgeschwindigkeit und des Drehmoments durch das Differential zwischen Getriebe und Antriebswelle abgebildet.
- Es wird das Schwingungsverhalten durch ein Feder-Dämpfer-Element zwischen Getriebe und Fahrzeug berücksichtigt. Dazu wird eine Ersatzträgheit des Getriebes angenommen, die sich am vorgegebenen energieflussbasierten Modell orientiert.

Das so implementierte Fahrzeugmodell wird dann in Kombination mit den erstellten neuronalen Netzen verwendet, wie es in Kapitel 6 beschrieben ist. Auf diese Weise können die numerischen Eigenschaften der erstellten neuronalen Netze bewertet werden. Zudem können die Modellgüte und die Fortpflanzung von Modellabweichungen in einer geschlossenen Simulation betrachtet werden. Dabei ist zu erwarten, dass durch Abweichungen bei der Modellierung des Drehmoments an der Antriebswelle auch Abweichungen bei den Winkelgeschwindigkeiten der Antriebswelle auftreten, die sich wiederum auf die Berechnung der Winkelgeschwindigkeit des Motors auswirken.

7.5.2 Numerische Eigenschaften der LM-Netze

Für die Untersuchung wurde das dynamische lokale Modellnetz als s-Function implementiert. Für weitere Information zu s-Functions wird auf [Mat] oder [Dro03] verwiesen. Wie in Kapitel 6 werden dabei über eine definierte Schnittstelle die aus der Liniensuche gewonnenen Modellparameter, Hyperparameter und Normierungsparameter übergeben, damit anschließend die erstellten LM-Netze in der Simulation verwendet werden können.
Für die Untersuchung der numerischen Eigenschaften wurde exakt das gleiche Fahrszenario abgefahren, wie es beim energieflussbasierten Modell verwendet wurde. Zudem wurde ein numerisches Verfahren mit fester Schrittweite ausgewählt, wobei die Schrittweite $0,01$[s] betrug. Die gesamte Rechendauer, um die Zeitdauer der Fahrszenarien und Schaltvorgänge von 3000[s] zu simulieren, betrug mit einer festen Schrittweite von $0,01$[s] etwa 45[s].
Die Berechnung des gleichen Fahrszenarios hat auf dem gleichen Simulationsrechner in Dymola etwa 500[s] gedauert, wie oben beschrieben wurde. Somit wurde eine Verkürzung der Simulationsdauer um etwa den Faktor 10 erreicht. Zudem kann nun ein Simulationsverfahren mit deutlich größerer fester Schrittweite als zuvor (von etwa 10^{-7}[s] auf 10^{-2}[s]) verwendet werden, wodurch die gewünschte numerische Vereinfachung erzielt wurde. Dieses Ergebnis zeigt, dass durch die Verwendung von LM-Netzen das Schaltverhalten nun deutlich schneller simuliert werden kann und somit eine deutliche numerische Vereinfachung erzielt wurde. In diesem Beispiel ist auch eine Echtzeitsimulation möglich.

7.5.3 Numerische Eigenschaften der MLP-Netze

Dazu wurde die Möglichkeit genutzt, das erstellte MLP-Netz mit Hilfe der Toolbox von Mathworks [Mat] nach Simulink zu exportieren und anschließend

wie die lokalen Modellnetze in der Simulation zu verwenden. Dazu wurden zusätzlich die Filterung durch die Laguerre-Filter als s-Function implementiert. Da die dynamischen MLP-Netze etwas weniger Modellparameter besitzen, war die Berechnungsdauert hier etwas kürzer als bei der Verwendung von LM-Netzen und betrug etwa 40[s]. Die Verkürzung der Simulationsdauer erfolgt hier um etwa einen Faktor 13. Dieses Ergebnis zeigt, dass auch durch die Verwendung von MLP-Netzen das Schaltverhalten deutlich schneller simuliert werden kann und somit eine deutliche numerische Vereinfachung erzielt wurde und eine Echtzeitsimulation möglich ist. Im nächsten Abschnitt wird auf die Modellgüte der beiden Ansätze eingegangen.

7.5.4 Bewertung der Modellgüte

In diesem Abschnitt wird auf die Fehlerfortpflanzung eingegangen, die in der Simulation entsteht. Dabei muss folgende Ausbreitung des Fehler betrachtet werden:

- Das dynamische neuronale Netz, das das Drehmoment T_{at} der Antriebswelle approximiert, wird im Fahrzeugmodell als beschleunigendes Drehmoment verwendet. Durch die Approximation entstehen Fehler bei der Berechnung der Winkelgeschwindigkeit ω_{at} der Antriebswelle. Da diese Winkelgeschwindigkeit als Eingangssignal im dynamischen neuronalen Netz verwendet wird, das die Winkelgeschwindigkeit ω_{m} des Motors bestimmt, pflanzt sich der Fehler fort.
- Das Motormodell benötigt die Winkelgeschwindigkeit ω_{m} als Eingangssignal zur Berechnung des Drehmoments T_{m} des Motors. Somit kann sich hier der Fehler weiter fortpflanzen, und es kann über die Zeit zu größeren Abweichungen kommen.

Eine vorteilhafte Eigenschaft des Fahrzeugmodells und des Motormodells ist, dass diese ein Tiefpass-Verhalten besitzen. Das hat zur Folge, dass kurzzeitige Abweichungen bei den Drehmomenten gedämpft werden und diese somit nur zu kleineren weiteren Abweichungen führen. Die Betrachtung der Simulationsergebnisse zeigt, dass die Auswirkungen dieser Fehlerkette tatsächlich gering ist. Die Simulation ergab folgende RMSE unter Verwendung der normierten Signale, wobei jedoch nur die Testdaten betrachtet wurden:

- MLP-Netz: $\mathrm{RMSE}_{\mathsf{TS}} = 0,0189$
- LM-Netz: $\mathrm{RMSE}_{\mathsf{TS}} = 0,0216$

Es ist zu erkennen, dass beim LM-Netz die Auswirkungen der Abweichungen etwas größer sind als beim MLP-Netz. Allerdings bleibt die Modellgüte der oben erstellten Netze durch die Liniensuche in der Simulation erhalten. Dies ist eine Folge der oben beschriebenen Tendenz, dass auch die Modellgüte der erstellten MLP-Netze etwas höher ist, bei denen zunächst keine Fehlerfortpflanzung betrachtet wurde. In Abbildung 7.14 ist der Verlauf der Winkelgeschwindigkeit ω_{at} der Antriebswelle in einer Simulation dargestellt. Dort ist im Vergleich zu sehen, welche Winkelgeschwindigkeit unter Verwendung des Referenzsignals T_{at} resultiert und das Ergebnis, wenn die beiden erstellten neuronalen Netze verwendet werden. Dabei ist zu erkennen, dass beide neuronalen Netze etwa gleich gut abschneiden, und dass sich die Abweichungen bei der Approximation des Drehmoments der Antriebswelle kaum bemerkbar machen. Die so berechneten Winkelgeschwindigkeiten können nun in den neuronalen Netzen zur Modellierung der Winkelgeschwindigkeit des Motors verwendet werden.
In Abbildung 7.15 ist der Verlauf der Winkelgeschwindigkeit ω_{m} des Motors in einer Simulation dargestellt. Dort sind die Auswirkung der Abweichungen bei der Berechnung der Winkelgeschwindigkeit ω_{at} direkt ersichtlich.

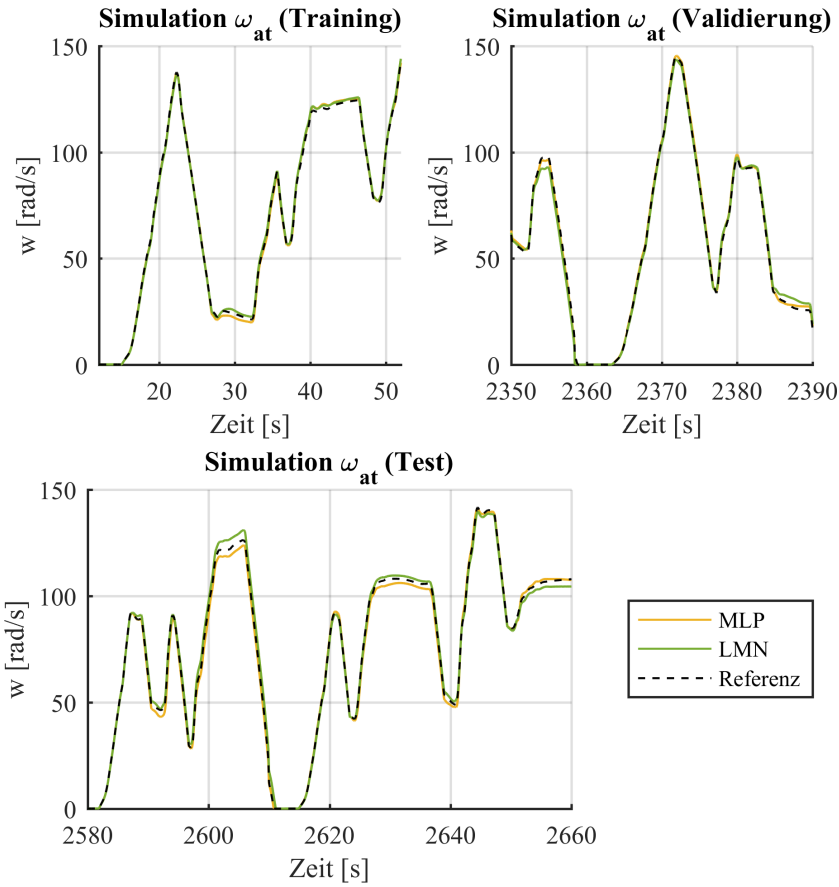


Abbildung 7.14: Winkelgeschwindigkeit der Antriebswelle in der Simulation

Dabei ist zudem zu sehen, dass sich die Abweichungen auf das lokale Modellnetz kaum auswirken und die Ergebnisse von MLP-Netz und LM-Netz ähnlich sind. Allerdings lässt der oben genannte RMSE den Schluss zu, dass das MLP-Netz insgesamt etwas besser abschneidet.

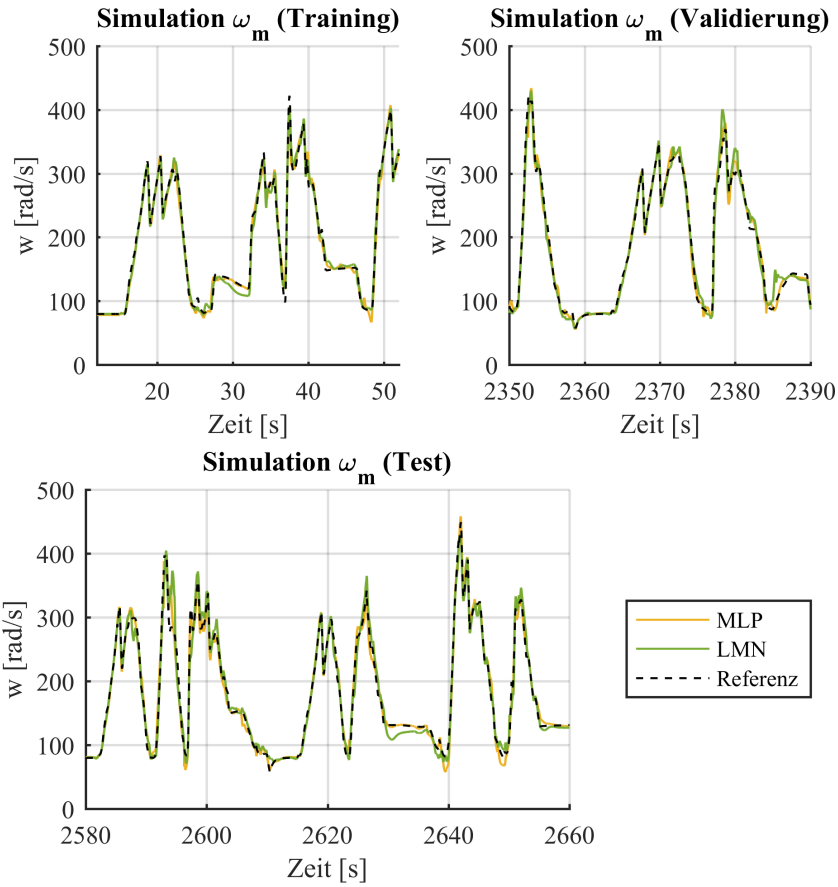


Abbildung 7.15: Winkelgeschwindigkeit des Motors in der Simulation

7.6 Fazit

In diesem Kapitel wurde gezeigt, wie aus einem energieflussbasierten Modell eines Stufenautomaten ein numerisch vereinfachtes, signalflussorientiertes Modell

erstellt werden kann. Die dabei erstellten Simulationsdaten dienten als Grundlage zur Anwendung der in dieser Arbeit entwickelten Methode aus Kapitel 6 zur automatischen numerischen Vereinfachung. Dabei war die Dauer zur Erstellung der MLP-Netze deutlich höher als die zur Erstellung der LM-Netze. Während die Liniensuche zur Erstellung von LM-Netzen nach etwa $2[\mathrm{h}]$ abgeschlossen war, hat die Liniensuche zur Erstellung von MLP-Netzen bis zu $20[\mathrm{h}]$ gedauert. Ein Grund dafür ist die Verwendung des Levenberg-Marquardt-Verfahrens zur Parameteroptimierung. Bei der Betrachtung der Modellgüte wurde gezeigt, dass diese bei beiden neuronalen Netzen gut ist, und dass die MLP-Netze etwas besser abschneiden. Die Modellgüte wurde dabei an Hand des RMSE und der Signalverläufe diskutiert, wobei der RMSE sich in der Größenordnung von $0,02$ bei den Trainings-, Validierungs- und Testdaten befand. Bei Betrachtung der Signalverläufe konnte gezeigt werden, dass es bei schnell aufeinander folgenden Gangwechseln zu größeren Abweichungen bei der Übersetzung und beim Drehmoment an der Antriebswelle kommt.
Anschließend wurde auf die Verwendung der erstellten neuronalen Netze in einer Simulation eingegangen. Dazu wurde ein signalflussorientiertes Modell des Fahrzeugs nachgebildet, das dem in Dymola entspricht. Es wurde dabei gezeigt, dass die Fehlerfortpflanzung in der Simulation bei beiden neuronalen Netzen gering ist und die Modellgüte der Liniensuche beibehalten wird. So hat auch hier das MLP-Netz etwas besser abgeschnitten hat. Die Simulation mit den beiden neuronalen Netzen war um etwa den Faktor 10 bis 13 schneller als die Simulation des ursprünglichen Dymola-Modells, wobei das gleiche Fahrszenario verwendet wurde. Zudem konnte ein Simulationsverfahren mit einer festen Schrittweite von $0,01[\mathrm{s}]$ verwendet werden, so dass in diesem Beispiel sogar eine Echtzeitsimulation möglich ist.

8 Zusammenfassung und Ausblick

In dieser Arbeit wurde behandelt, wie detaillierte Getriebemodelle automatisch numerisch vereinfacht werden können. Um dies zu erreichen, können entweder mathematische Verfahren zur Modellvereinfachung verwendet werden, oder es wird ein numerisch vereinfachtes Ersatzmodell erstellt. Dabei wurde diskutiert, welcher der beiden Ansätze zur numerischen Vereinfachung sich besser eignet. Das Ergebnis war dabei, dass die Erstellung eines numerisch vereinfachten Ersatzmodells der geeignetere Ansatz ist. Da die Schaltdynamik von Stufenautomaten durch eine Vielzahl von Eingangssignalen beeinflusst wird und zudem ein nichtlineares Verhalten aufweist, eignet sich die Verwendung neuronaler Netze als Ersatzmodell. Um eine automatische numerische Vereinfachung zu ermöglichen, wurde gezeigt, wie die benötigten neuronalen Netze automatisch erstellt werden können. Da es jedoch eine Vielzahl von Möglichkeiten gibt, neuronale Netze zur Modellierung zu verwenden, wurde in Kapitel 4 diskutiert, welche dieser Möglichkeiten sich am besten eignet. Zur Bewertung dieser verschiedenen Möglichkeiten wurden die Stabilität in einer Simulation, der Aufwand zur Modellerstellung und das Potential zur numerischen Vereinfachung als Kriterien verwendet. Dabei erwiesen sich neuronale Netze mit externen Laguerre-Filtern als beste Möglichkeit, um diese drei Kriterien zu erfüllen. Als neuronale Netze können dabei MLP-Netze oder LM-Netze verwendet werden.

Um eine automatische numerische Vereinfachung zu ermöglichen, ist ein Verfahren nötig, durch das sowohl die Struktur als auch die Parameter von neuronalen Netzen optimiert werden können. Dabei wird die Struktur durch Hyperparameter beschrieben, die zu einer unterschiedlichen Anzahl von Modellparametern

führen. Bei neuronalen Netzen mit externen Laguerre-Filtern sind dabei die Pole, die Ordnungen und die Architektur des neuronalen Netzes Hyperparameter, die optimiert werden. Um die Hyperparameter zu optimieren, wurde in Kapitel 5 eine Liniensuche mit stochastischen Elementen vorgestellt. Weiter wurde beschrieben, wie die Anzahl von Ordnungen und Polen reduziert werden kann, um den Aufwand der Liniensuche zu senken. Die dabei vorgestellten Ansätze wurden verwendet, um generische Modellstrukturen abzuleiten, die zur Modellierung der Schaltdynamik von Stufenautomaten verwendet werden. Diese können dabei unabhängig von der Anzahl der Ansteuerströme von Stufenautomaten verwendet werden. Das Vorgehen zur automatischen numerischen Vereinfachung ist in Kapitel 6 beschrieben. Dabei wurde gezeigt, dass die Verwendung von MLP-Netzen zur numerischen Vereinfachung zu präferieren ist, obwohl sie aufwändiger zu erstellen sind.

In Kapitel 7 wurde dann das vorgestellte Vorgehen zur numerischen Vereinfachung an einem detaillierten Getriebemodell mit neun Gangstufen demonstriert. Dazu wurde ein 3000[s] langes Fahrszenario verwendet. Die dabei entstehenden Simulationsdaten wurden dann zur automatischen Erstellung neuronaler Netze mit externen Laguerre-Filtern verwendet. Dabei wurden zunächst MLP-Netze und anschließend LM-Netze als neuronales Netz zur Modellierung der Schaltdynamik ausgewählt. Das Ergebnis war, dass die Erstellung der MLP-Netze, wie erwartet, deutlich länger gedauert hat. Jedoch war die Modellgüte der erstellten neuronalen Netze in einer ähnlichen Größenordnung. Zur Bewertung der Modellgüte wurde der RMSE verwendet, der bei Trainings-, Validierungs- und Testdaten bei etwa $0,02$ lag[1]. Die Modellierung des Drehmoments des Motors war überwiegend sehr gut. Abweichungen gab es, wenn schnell hintereinander Gangstufen gewechselt wurden. Bei der Modellierung der Winkelgeschwindigkeit war das MLP-Netz etwas besser und hatte geringere Abweichungen. Der Vergleich der Anzahl der Modellparameter der beiden Netztypen ergab, dass

[1] Zur Berechnung des RMSE wurden die auf zwischen 0 und 1 normierten Signale verwendet.

das MLP-Netz auch etwa halb so viele Modellparameter benötigt. Dies ist vor allem auf die unterschiedlichen Konstruktionsmechanismen zurückzuführen.

Zum Abschluss wurden die erstellten Netze in einer Simulation verwendet, um die numerischen Eigenschaften zu bewerten. Dabei wurde gezeigt, dass sich mit beiden Netzen eine deutliche Verkürzung der Simulationszeit erzielen lässt. Bei MLP-Netzen war der Faktor mit 13 etwas größer als bei LM-Netzen mit einem Faktor von 10. Dies ließ sich auf die geringere Anzahl der Modellparameter von MLP-Netzen zurückführen. Zudem war es nun möglich, die Simulation mit einer festen Schrittweite von $0,01\,[\mathrm{s}]$ durchzuführen. Dies führte dazu, dass in diesem Beispiel eine Echtzeitsimulation möglich war.

Bei Betrachtung der Modellgüte wurde gezeigt, dass die Auswirkungen der Abweichungen bei der Modellierung des Drehmoments der Antriebswelle gering waren und die zuvor erhaltene Modellgüte bei der Modellerstellung in der Simulation beibehalten werden konnte. Es muss jedoch im Einzelfall überprüft werden, ob die gezeigten Abweichungen bei der Modellierung mit neuronalen Netzen für die zu Grunde liegende Simulationsfragestellung akzeptabel sind. Für viele Simulationsfragestellungen können jedoch die so erstellten Modelle verwendet werden, wobei die numerische Vereinfachung sich vorteilhaft durch kürzere Simulationszeiten bemerkbar macht.

In weiteren Arbeiten kann überprüft werden, inwiefern die verwendete Liniensuche mit stochastischen Elementen zur numerischen Vereinfachung von weiteren Teilmodellen des Fahrzeugs verwendet werden kann. Eine ähnliche Problematik, wie in dieser Arbeit beschrieben, tritt bei der Modellierung hydraulischer Bremsen und Doppelkupplungsgetrieben auf.

Verbesserungen bei der Modellerstellung können erzielt werden, indem eine Verwendung individueller Ordnungen und Pole ermöglicht wird. Dabei steigt

jedoch der Aufwand der Optimierung. Deshalb könnte in zukünftigen Arbeiten untersucht werden, inwiefern erstens durch die Verwendung individueller Ordnungen und Pole höhere Modellgüten erzielt werden, und zweitens, inwiefern der Aufwand zur Strukturoptimierung bzw. der Modellerstellung reduziert werden kann.

Anhang

A Liniensuche und Verfahren zur Erstellung von LM-Netzen

A.1 Liniensuche mit Rauschen

A.1.1 Untersuchung der Rauschamplitude

Um den Einfluss der Rauschamplitude zu untersuchen, wird eine fiktive Kostenfunktion $f(\mathsf{P}_1, \mathsf{P}_2)$ mit zwei Parametern verwendet, deren Minimum gefunden werden soll. Bei dieser Kostenfunktion kann die Abhängigkeit der zu optimierenden Parameter eingestellt werden, um den Einfluss auf die Suche des Minimums zu untersuchen. Zur Suche des Minimums kommt das Verfahren aus Kapitel 5 zum Einsatz (siehe Abbildung 5.6). Die Kostenfunktion, deren Minimum gesucht wird, ist dabei gegeben durch

$$\begin{aligned} &f(\mathsf{P}_1, \mathsf{P}_2) = \\ &1 - \exp\left(-(2 \cdot (\mathsf{P}_1 - a)^2 + 2 \cdot (\mathsf{P}_2 - b)^2 - c \cdot (\mathsf{P}_1 - a) \cdot (\mathsf{P}_2 - b))\right). \end{aligned} \tag{A.1}$$

Mit Hilfe der Parameter a und b wird die Lage des Minimums bestimmt. In diesem Beispiel wurde das Minimum bei $a = 0,25$ und $b = 0,15$ festgelegt. Durch den Parameter c wird die Stärke der Abhängigkeit zwischen den beiden Parametern a und b eingestellt. Dabei bedeutet $c = 0$ keine und $c = 3,8$ eine sehr starke Abhängigkeit. Um das in Kapitel 5 beschriebene Verfahren zu untersuchen, werden folgende Suchräume für die beiden Parameter P_1 und P_2 verwendet und gegeben sind durch

$$\begin{aligned} \mathsf{SP}_1 &= \text{linspace}\left(-\frac{3}{2}, \frac{3}{2}, \mathsf{NP}\right) \\ \mathsf{SP}_2 &= \text{linspace}\left(-\frac{3}{2}, \frac{3}{2}, \mathsf{NP}\right). \end{aligned} \tag{A.2}$$

In diesem Fall wird für die Suche in beiden Räumen die gleiche Anzahl von Werten verwendet. Für diese muss noch die Rauschamplitude $\Delta\mathsf{P}$ vorgegeben werden. In diesem Fall wird diese für die beiden Parameter gleich gewählt. Um die Vorgaben komplett zu machen, muss die Anzahl $\mathsf{N_O}$ der Suchdurchläufe vorgegeben werden. Die Anzahl der Iterationen zur Bestimmung des Minimums beträgt $2 \cdot \mathsf{N_O} \cdot \mathsf{NP}$. In Abbildung A.1 ist die Suche des Minimums für $c = 3,5$ der Kostenfunktion zu sehen. Dabei wurde $\mathsf{N_O} = 10$ gewählt.
Die starke Abhängigkeit der Parameter P_1 und P_2 ist am Verlauf der eingezeichneten Höhenlinien zu erkennen. In jedem Suchraum wurden 11 Werte verwendet. In der Abbildung ist zusätzlich der Verlauf der Suche nach dem Minimum eingezeichnet, wobei nur die jeweils besten Werte der beiden Parameter zu sehen sind. Im Fall ohne Rauschen ($\Delta\mathsf{P} = 0$) ist zu erkennen, dass bei der Suche genau entlang der Achsen gesucht wird. Im Fall mit Rauschen ($\Delta\mathsf{P} = 0,075$) wird deutlich, dass durch das Rauschen nicht nur entlang der Achsen gesucht wird, wodurch sich hier die gefunden Parameter näher am tatsächlichen Minimum befinden. Bei einer stärkeren Rauschamplitude weicht der Verlauf immer stärker ab. Die Wahl der Anzahl der Suchdurchläufe $\mathsf{N_O}$, der Anzahl der Werte NP in den beiden Suchräumen und der Wert der Rauschamplitude $\Delta\mathsf{P}$ beeinflussen also das Ergebnis der Suche nach dem Minimum.

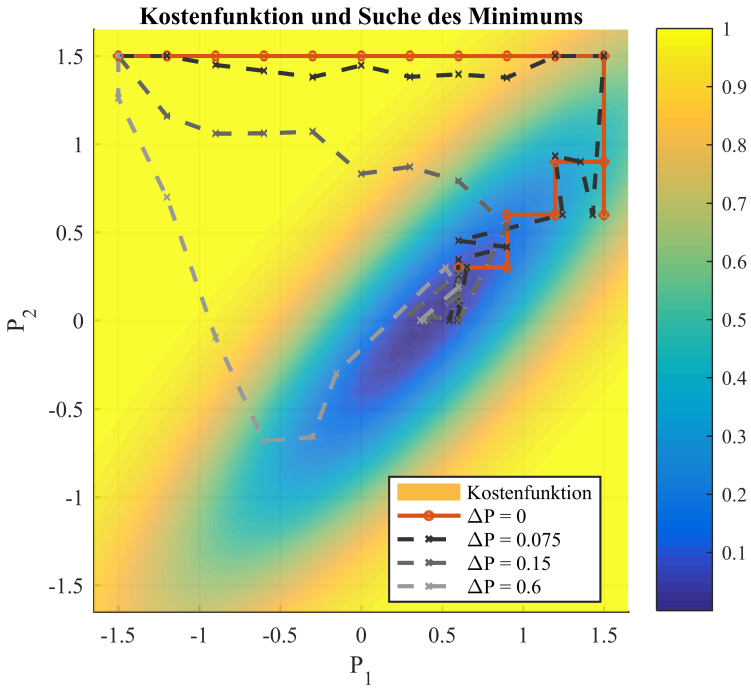


Abbildung A.1: Kostenfunktion und Suche des Minimums mit den gefundenen besten Werten für die Parameter P_1 und P_2 mit und ohne Rauschen ($c = 3{,}5$, $NP = 11$, $N_O = 10$)

Um zu zeigen, welche Vorteile sich bei Verwendung einer Rauschamplitude $\Delta P > 0$ ergeben, und wie diese zu wählen ist, wurde folgende Untersuchung durchgeführt:
In Abhängigkeit von der Anzahl der Werte NP in den jeweiligen Suchräumen wurde untersucht, welches Minimum gefunden wurde. Da im Fall $\Delta P > 0$ der Verlauf zur Suche des Minimums für eine feste Rauschamplitude sehr unterschiedlich ist, wird dieser folgendermaßen statistisch ausgewertet. Für $\Delta P > 0$

wird für jede vorgegebene Anzahl von Werten NP in den beiden Suchräumen die Suche nach dem Minimum mittels der Liniensuche 250-mal wiederholt, um den Einfluss des Rauschens zu untersuchen. Dazu wurde folgendes untersucht:

- Median der gefundenen Minima:
  Der Median ist robuster als der Mittelwert gegenüber Ausreißern und liefert einen guten Eindruck davon, welches Minimum typischerweise gefunden wird.
- Maximaler und minimaler Wert der gefundenen Minima:
  Der maximale Wert gibt an, welches Minimum im schlechtesten Fall gefunden wurde. Umgekehrt zeigt der minimale Wert an, welches Minimum im besten Fall gefunden wurde.

Zur Untersuchung der Abhängigkeit zwischen der Anzahl der Werte NP und des gefunden Minimums wurden folgende Fälle betrachtet:

1. Stark abhängige Parameter mit $c = 3,75$:
   a) Kleine Rauschamplitude $\Delta\mathsf{P} = 0,1$ mit $\mathsf{N_O} = \begin{pmatrix} 2 & 4 & 8 & 16 \end{pmatrix}$
      (siehe Abbildung A.2)
   b) Mittlere Rauschamplitude $\Delta\mathsf{P} = 0,45$ mit $\mathsf{N_O} = \begin{pmatrix} 2 & 4 & 8 & 16 \end{pmatrix}$
      (siehe Abbildung A.3)
   c) Große Rauschamplitude $\Delta\mathsf{P} = 0,8$ mit $\mathsf{N_O} = \begin{pmatrix} 2 & 4 & 8 & 16 \end{pmatrix}$
      (siehe Abbildung A.4)
2. Schwach abhängige Parameter mit $c = 1,5$:
   a) Kleine Rauschamplitude $\Delta\mathsf{P} = 0,1$ mit $\mathsf{N_O} = \begin{pmatrix} 2 & 4 & 8 & 16 \end{pmatrix}$
      (siehe Abbildung A.5)
   b) Mittlere Rauschamplitude $\Delta\mathsf{P} = 0,45$ mit $\mathsf{N_O} = \begin{pmatrix} 2 & 4 & 8 & 16 \end{pmatrix}$
      (siehe Abbildung A.6)
   c) Große Rauschamplitude $\Delta\mathsf{P} = 0,8$ mit $\mathsf{N_O} = \begin{pmatrix} 2 & 4 & 8 & 16 \end{pmatrix}$
      (siehe Abbildung A.7)

Alle zu den einzelnen Fällen genannten Abbildungen enthalten jeweils den Verlauf des gefundenen Minimums in Abhängigkeit von der Anzahl NP der Werte in den Suchräumen. In einer Abbildung ist dabei der Verlauf für die unterschiedlichen, gegebenen Werte für die Anzahl der Suchdurchläufe dargestellt. Zum Vergleich ist der Fall ohne Verrauschen der Suchparameter eingezeichnet ($\Delta\mathsf{P} = 0$). Im Folgenden wird zunächst der Fall mit stark abhängigen ($c = 3,75$) und anschließend der mit schwach abhängigen Parametern ($c = 1,5$) diskutiert. Der Fall mit schwach abhängigen Parametern dient dabei zur Überprüfung, ob hier ein Verrauschen der Suchparameter kontraproduktiv ist.

Im Fall stark abhängiger Parameter (Fälle 1a, 1b und 1c) wird deutlich, dass durch das Verrauschen der Suchparameter in allen Fällen das tatsächliche Minimum besser erreicht wird (siehe Verlauf des Medians). Auch in den jeweiligen schlechtesten Fällen (siehe Verlauf des Maximums) wird das tatsächliche Minimum besser erreicht als im rauschfreien Fall. Zudem wird klar, dass auch bei einer kleineren Anzahl $\mathsf{N_O}$ von Suchdurchläufen das tatsächliche Minimum besser erreicht wird. Ein Vergleich der Rauschamplituden $\Delta\mathsf{P}$ ergibt, dass bei einer zu kleinen Wahl tendenziell mehr Werte NP in den Suchräumen ausgewählt werden müssen. Wird die Rauschamplitude zu groß gewählt, wirkt sich das tendenziell wieder kontraproduktiv aus. Das Zwischenfazit lautet hier, dass im Falle stark abhängiger Parameter das Verrauschen der Suchparameter eine effizientere Möglichkeit zur Suche des Minimums bietet. Es werden hier weniger Suchdurchläufe $\mathsf{N_O}$ und eine geringere Anzahl von Werten NP in den Suchräumen benötigt. Eine gute Wahl für die Rauschamplitude ist in diesem Beispiel $\Delta\mathsf{P} = 0,4$.

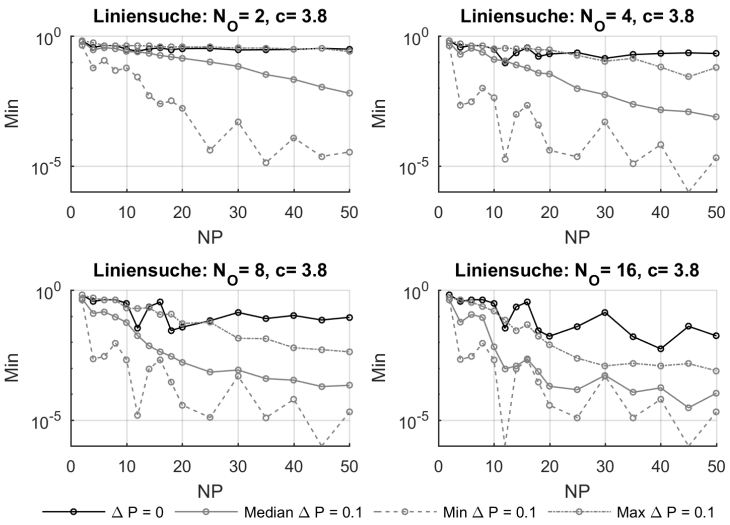


Abbildung A.2: Untersuchung der Liniensuche mit Rauschamplitude $\Delta P = 0,1$

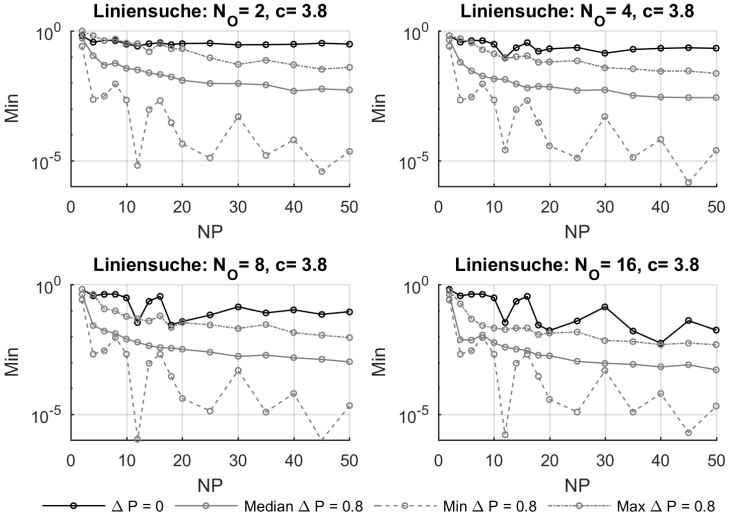


Abbildung A.4: Untersuchung der Liniensuche mit Rauschamplitude $\Delta P = 0,8$

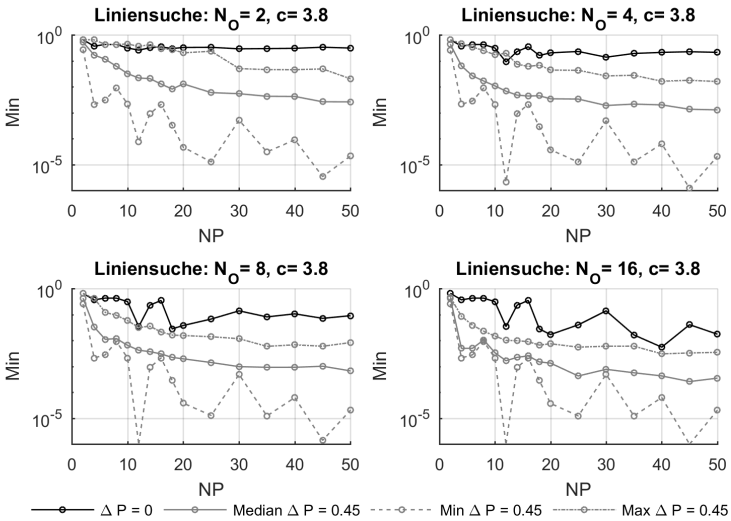


Abbildung A.3: Untersuchung der Liniensuche mit Rauschamplitude $\Delta P = 0,45$

Im Fall schwach abhängiger Parametern wird deutlich (Fälle 2a, 2b und 2c), dass erst für eine höhere Anzahl von Suchdurchläufen N_O das Verrauschen der Suchparameter im schlechtesten Fall nicht kontraproduktiv ist (siehe Verlauf des Maximums). Es wird jedoch deutlich, dass im Falle kleinerer Werte NP in den Suchräumen typischerweise (siehe Verlauf des Medians) kein Unterschied zwischen dem Fall mit und ohne Rauschen besteht. Für große Werte NP besteht eine große Wahrscheinlichkeit, dass durch Verrauschen der Suchparameter das tatsächliche Minimum besser erreicht wird als ohne Rauschen. Jedoch wird dieser Effekt durch eine zu große Rauschamplitude ΔP wieder aufgehoben.
Für einen Suchraum mit kontinuierlichen Werten zwischen 0 und 1 wird daher eine optimale Rauschamplitude von $\Delta P = 0,075$ empfohlen. Mit diesem Wert werden in diesem Beispiel sowohl bei stark abhängigen als auch bei schwach abhängigen Parametern gute Ergebnisse erzielt. Für andere Anwendungen wird

davon ausgegangen, dass es sich ähnlich verhält. Die Rauschamplitude skaliert dabei linear mit den Grenzen des Suchraums. Für ganzzahlige Werte der Rauschamplitude, wird empfohlen, eine Rauschamplitude von $\Delta P = 0,75$ zu verwenden, unabhängig von der Anzahl der Werte im Suchraum.

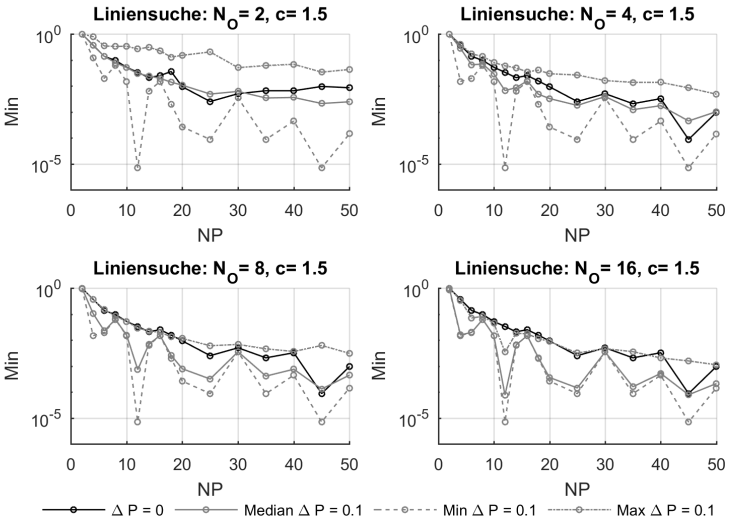


Abbildung A.5: Untersuchung der Liniensuche mit Rauschamplitude $\Delta P = 0,1$

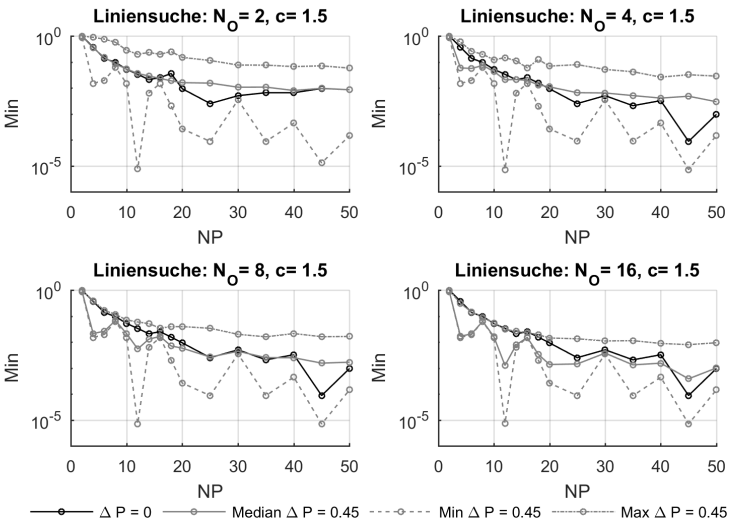


Abbildung A.6: Untersuchung der Liniensuche mit Rauschamplitude $\Delta P = 0,45$

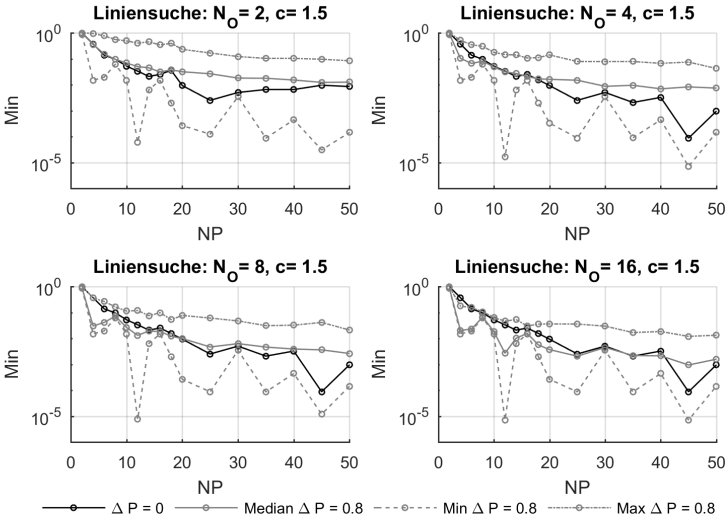


Abbildung A.7: Untersuchung der Liniensuche mit Rauschamplitude $\Delta P = 0,8$

Zusammenfassend lassen die zuvor beschriebenen Untersuchungen folgende Schlüsse zu:

- Im Allgemeinen ist unbekannt, wie stark die Abhängigkeit von Parametern ist. Jedoch zeigt sich, dass im Fall schwach abhängiger Parameter im Mittel durch Verrauschen das Minimum der Kostenfunktion besser erreicht wird. Im Fall stark abhängiger Parameter ist die Tendenz zu diesem Verhalten noch stärker ausgeprägt.
- Es muss eine geringere Anzahl von Suchdurchläufen N_O durchgeführt werden, wodurch der Aufwand der Modellerstellung sinkt.
- Es wird eine geringere Anzahl NP von Werten in den Suchräumen benötigt. Zudem hängt die Liniensuche weniger stark von den gewählten Werten in den Suchräumen ab. Dies führt weiter zu einem gesenkten Aufwand der Modellerstellung
- Für kontinuierliche Parameter wird die Verwendung einer Rauschamplitude von $0,075$ empfohlen, sofern es sich um einen Suchraum mit Werten zwischen 0 und 1 handelt. Für andere Grenzen des Suchraums wird die Rauschamplitude linear skaliert.
  Für ganzzahlige Parameter wird die Verwendung einer einheitlichen Rauschamplitude von $0,75$ empfohlen.

A.1.2 Laguerre-Filter zur Modellierung eines linearen PT2-Systems

In diesem Abschnitt wird gezeigt, wie der Pol und die Ordnung von Laguerre-Filtern durch Anwendung der Liniensuche optimiert werden können. Ziel ist es, Modellgleichungen aufzustellen, so dass gezielt ein bestimmtes gedämpftes Verhalten eingestellt werden kann. Dazu sollen die Schwingungsfrequenz und die Abklingzeit vorgegeben werden können. Dazu wird der harmonische Oszillator als Beispiel für ein PT2-System betrachtet. Dieser ist gegeben durch

$$m \cdot \ddot{y}(t) + d \cdot \dot{y}(t) + c \cdot y(t) = F(t). \tag{A.3}$$

Dabei sind m die Masse, d die Dämpfung und c die Federkonstante. $F(t)$ ist eine zeitabhängige äußere Kraft, die auf den harmonischen Oszillator einwirkt, so dass sich die Auslenkung $y(t)$ ändert. Als signalflussorientiertes Modell betrachtet, stellen $F(t)$ das Eingangssignal und $y(t)$ das Ausgangssignal dar. Gleichung A.3 kann folgendermaßen umgeformt werden

$$\begin{aligned} \ddot{y}(t) + \gamma \cdot \dot{y}(t) + \omega_0^2 \cdot y(t) &= u(t) \\ \text{Eigenfrequenz: } \omega_0^2 &= \frac{c^2}{m^2} \\ \text{Abklingkonstante: } \gamma &= \frac{d}{2m}. \end{aligned} \tag{A.4}$$

Wird der harmonische Oszillator mit einem Sprung mit Wert u_0 angeregt, ergibt sich mit den beschriebenen Parametern der umgeformten Gleichung folgendes Verhalten

$$\begin{aligned} y(t) &= A_0 \cdot \exp(-t \cdot \gamma) \cdot \sin(\omega_\mathrm{D} \cdot t + \phi_0) + \omega_0^2 \cdot u_0 \\ \text{mit: } \omega_\mathrm{D}^2 &= \omega_0^2 - \gamma^2. \end{aligned} \tag{A.5}$$

Dabei wird deutlich, dass die resultierende Schwingung je nach Wert der Abklingkonstante eine veränderte Kreisfrequenz ω_D hat. Zusätzlich bestimmt der Wert der Abklingkonstante, wie stark die Schwingung gedämpft wird. Nachdem die angeregte Schwingung abgeklungen ist, wird das neue Niveau $\frac{u_0}{\omega_0^2}$ erreicht. Soll nun das Niveau des Ausgangssignals gleich dem des Eingangssignals sein, muss die Sprungantwort noch mit der Eigenfrequenz multipliziert werden.
Für die Untersuchung der Modellierung mit Laguerre-Filtern sollen die Abklingzeit τ und die Frequenz $\omega_\mathrm{D} = 2\pi \cdot f_\mathrm{D}$ der resultierenden Schwingung vorgegeben werden und die entsprechende Modellgleichung in einer Simulation verwendet werden, um so die Daten zu gewinnen. Dazu wird die Masse $m = 1$

gewählt. Die Abklingkonstante γ ist umgekehrt proportional zur Abklingzeit $\tau = \frac{1}{\gamma}$. Obige Modellgleichung kann folgendermaßen umgeformt werden (unter Verwendung von Gleichung A.4)

$$\ddot{y}(t) + \frac{1}{\tau} \cdot \dot{y}(t) + \underbrace{\left((2\pi f_{\mathrm{D}})^2 + \frac{1}{\tau^2} \right)}_{\tilde{\omega}^2} \cdot y(t) = \tilde{\omega}^2 \cdot u(t). \tag{A.6}$$

Nun wird darauf eingegangen, wie ein PT2-System durch lineare Überlagerung von Laguerre-Antwortfunktionen approximiert werden kann. Dazu müssen die beiden Hyperparameter Ordnung η und Pol α bestimmt werden, um eine geeignete Approximation zu finden. Bei fest vorgegebenen Hyperparametern können die Parameter der linearen Überlagerung durch eine Least-Squares-Optimierung gefunden werden. Zur Untersuchung der Liniensuche wurden zunächst Simulationsdaten erzeugt. Dazu wurden eine Abklingzeit von $\tau = 0,2[\mathrm{s}]$ und eine Frequenz von $f_{\mathrm{D}} = 0,3[Hz]$ gewählt. Weiter wurden 250 Sprungantworten mit einer Dauer von $8[\mathrm{s}]$ aufgezeichnet. Die verschiedenen Niveaus der Sprungantworten wurden dabei durch gleichmäßig verteilte Zufallswerte zwischen 0 und 1 erzeugt. Von den so erzeugten Simulationsdaten wurden die ersten 80 % zur Parameteroptimierung und die letzten 20% zur Validierung verwendet. Um die Liniensuche weiter zu untersuchen und graphisch darzustellen, wie durch die Liniensuche der Pol und die Ordnung optimiert wurden, wurde die Kostenfunktion der beiden Hyperparameter im gesamten zweidimensionalen Raum bestimmt. Dazu wurden alle Ordnungen zwischen 1 und 41 verwendet. Für die Zeitkonstante wurden das Intervall zwischen $0,01[\mathrm{s}]$ und $1[\mathrm{s}]$ in 21 gleichmäßige Intervalle geteilt. Damit der Validierungsfehler in diesem Beispiel mit zunehmender Ordnung wieder leicht ansteigt, wurden die Ausgangswerte y mit einer kleinen Amplitude von $\sigma_r = 0,001$ durch eine gaußsches Rauschen verrauscht.

Folgende Suchräume wurden schließlich bei der Liniensuche verwendet

$$\begin{aligned} \mathrm{SP}_{\eta} &= \begin{pmatrix} 3 & 5 & \dots & 39 \end{pmatrix} \\ \mathrm{SP}_{T_\mathrm{K}} &= \mathrm{linspace}(0{,}01\ , 1, 11). \end{aligned} \tag{A.7}$$

In Abbildung A.8 ist zu sehen, wie das Minimum der Kostenfunktion durch die Liniensuche ermittelt wird. Dieses wird dabei schon nach 21 Iterationen gefunden (siehe Abbildung 5.3)

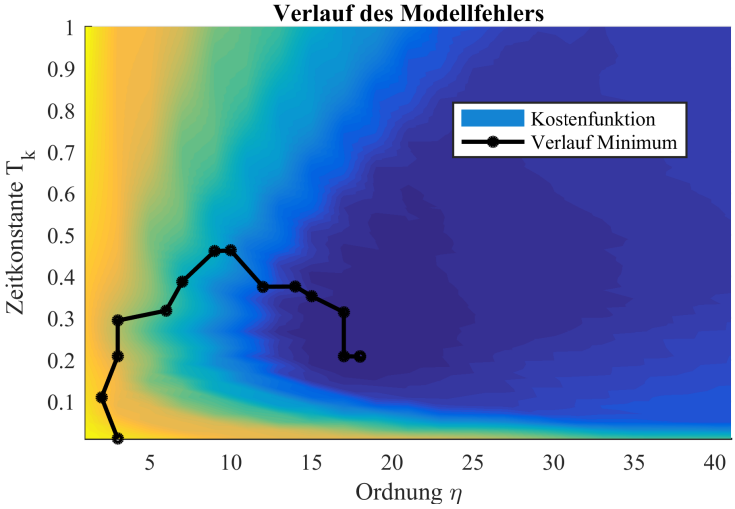


Abbildung A.8: Kostenfunktion und Verlauf des Minimums der Liniensuche

A.2 Algorithmen zur Erstellung von LM-Netzen

In diesem Kapitel werden zunächst wichtige Eigenschaften der linearen Optimierung lokaler Modellnetze beschrieben, die ein Teil der Partitionierungsverfahren sind. Die lineare Optimierung wird dann durchgeführt, wenn die Zentren bzw. die Aktivierungsfunktionen bekannt sind. Anschließend wird das in dieser Arbeit verwendete inkrementelle Clustering-Verfahren näher beschrieben.

A.2.1 Lineare Optimierung von LM-Netzen

Bei der linearen Optimierung lokaler Modellnetze kann zwischen der lokalen und der globalen Optimierung unterscheiden werden. Um Missverständnisse zu vermeiden, wird erwähnt, dass damit nicht die globale Optimierung gemeint ist, bei der durch gradientenfreie Verfahren das globale Minimum der Kostenfunktion ermittelt wird. Im Folgenden wird auf die Unterschiede und die jeweiligen Vorteile eingegangen.

Globale Optimierung

Bei der globalen Optimierung werden die Parameter $\mathbf{w}$ der lokalen Modelle des lokalen Modellnetzes gleichzeitig optimiert. Das heißt, dass die Zentren dabei nicht berücksichtigt werden, sondern bekannt sein müssen. Die globale Verlustfunktion J_{global} der N Messwerte ist gegeben durch

$$J_{\text{global}} = \sum_{i=1}^{\mathsf{N}} \left(y(i) - \hat{y}(i)\right)^2 = \left(\mathbf{y} - \hat{\mathbf{y}}\right)^T \left(\mathbf{y} - \hat{\mathbf{y}}\right). \tag{A.8}$$

Dabei ist $\mathbf{y}$ der Vektor der gemessenen Ausgänge und $\hat{\mathbf{y}}$ der Vektor der Modellausgänge. Der Modellausgang des lokalen Modellnetzes kann geschrieben werden als

$$\hat{\mathbf{y}} = \mathbf{X}_g \mathbf{w}. \tag{A.9}$$

Dabei ist $\mathbf{X}_g$ die Regressionsmatrix, die die Messwerte $\mathbf{X}$ der lokalen Modelle enthält. Die Messwerte werden zusätzlich mit Gültigkeitsfunktionen Φ_i gewichtet, die in die Gewichtungsmatrizen $\mathbf{Q}_j$ eingehen. Die Regressionsmatrix $\mathbf{X}_g$ ist dann geben durch

$$\mathbf{X}_g = \begin{pmatrix} \mathbf{Q}_1\mathbf{X} & \mathbf{Q}_2\mathbf{X} & \dots & \mathbf{Q}_\mathsf{M}\mathbf{X} \end{pmatrix}, \tag{A.10}$$

mit den Messwerten $\mathbf{X}$ der lokalen Modelle

$$\mathbf{X} = \begin{pmatrix} 1 & x_1(1) & x_2(1) & \dots & x_\mathsf{k}(1) \\ 1 & x_1(2) & x_2(2) & \dots & x_\mathsf{k}(2) \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_1(\mathsf{N}) & x_2(\mathsf{N} & \dots & x_\mathsf{k}(\mathsf{N}) \end{pmatrix} \tag{A.11}$$

und den Gewichtungsmatrizen

$$\mathbf{Q}_i = \operatorname{diag}\begin{pmatrix} \Phi_i(\mathbf{Z}(1,:)) & \Phi_i(\mathbf{Z}(2,:)) & \dots & \Phi_i(\mathbf{Z}(\mathsf{N},:)) \end{pmatrix}, \tag{A.12}$$

bei denen die Messwerte $\mathbf{Z}$ des Gültigkeitsraums verwendet werden (siehe Kapitel 4). Um das Minimum der Verlustfunktion in Gleichung (A.8) zu bestimmen, wird die Regressionsmatrix verwendet und eine globale Least-Squares-Schätzung durchgeführt. Die Parameter $\mathbf{w}$ sind dann geben durch [ZI08]

$$\mathbf{w} = \left(\mathbf{X}_g^T \mathbf{X}_g\right)^{-1} \mathbf{X}_g^T \mathbf{y}. \tag{A.13}$$

Diese Lösung wird gefunden, indem der Gradient

$$g_i = \frac{\partial J_{\text{global}}(\mathbf{w})}{\partial w_i} \tag{A.14}$$

für jeden Parameter w_i berechnet und betrachtet wird, wann dieser verschwindet [Nel01].

Lokale Optimierung

Bei der lokalen Optimierung werden die Modellparameter der lokalen Modelle separat bestimmt. Dazu wird die gewichtete Verlustfunktion $J_{k,\text{lokal}}$ des k-ten lokalen Modells betrachtet, um daraus die Parameter $\mathbf{w}_k$ des k-ten lokalen Modells zu bestimmen. Die gewichtete Verlustfunktion ist gegeben durch

$$J_{k,\text{lokal}} = \sum_{i=1}^{\mathsf{N}} \Phi_k\left(\mathbf{Z}(i,:)\right)\left(y(i) - \hat{y}_k(i)\right) = \left(\mathbf{y} - \hat{\mathbf{y}}_k\right)^T \mathbf{Q}_k \left(\mathbf{y} - \hat{\mathbf{y}}_k\right). \tag{A.15}$$

Dabei wird die Gewichtungsmatrix $\mathbf{Q}$ von oben verwendet. Der Modellausgang $\hat{\mathbf{y}}_k$ des k-ten Modells ist gegeben durch

$$\hat{\mathbf{y}}_k = \mathbf{X}\mathbf{w}_k. \tag{A.16}$$

Mit Hilfe dieser Gleichung kann durch Bildung des Gradienten und anschließender Suche, bei welcher Wahl der $\mathbf{w}_k$ dieser null ist, die Rechenvorschrift für die lokale Least-Squares-Schätzung abgeleitet werden [Nel01; Har14]. Die lokale Schätzung der Modellparameter hat die Eigenschaft, dass es zu einer implizierten Regularisierung kommt. Diese Regularisierung bedeutet, dass die Modellparameter benachbarter lokaler Modelle nicht beliebig stark voneinander abweichen können. Dadurch wird das Modell zwar unflexibler, jedoch robuster gegenüber Überanpassung (siehe Kapitel 4). Wie stark dieser Effekt ist, hängt

vom Grad der Überlappung der lokalen Modelle ab [Har14]. Die lokalen Schätzung bietet außerdem numerische Vorteile, da die Parameter jedes einzelnen lokalen Modells nacheinander bestimmt werden. In [MJ95] wird gezeigt, dass der Rechenaufwand nur linear mit der Anzahl der lokalen Modelle ansteigt, während er bei der globalen Schätzung kubisch ansteigt.
Da in dieser Arbeit zwischen dem **x**-Regressor und dem **z**-Regressor unterschieden werden kann, wird auf die lokale Optimierung für diesen allgemeineren Fall näher eingegangen. Dazu werden die Daten des Raums der Aktivierungsfunktionen zu der Matrix $\mathbf{Z}$ zusammengefasst, wobei die Messzeiten $1,2,\ldots,\mathsf{N}$ verwendet werden

$$\mathbf{Z} = \begin{pmatrix} z_1(1) & z_2(1) & \ldots & z_{\mathsf{pz}}(1) \\ z_1(2) & z_2(2) & \ldots & z_{\mathsf{pz}}(2) \\ \vdots & \vdots & & \vdots \\ z_1(\mathsf{N}) & z_2(\mathsf{N}) & \ldots & z_{\mathsf{pz}}(\mathsf{N}) \end{pmatrix}. \tag{A.17}$$

Das heißt in jeder Zeile steht der **z**-Regressor zu den jeweiligen Messzeiten. Diese werden dann verwendet, um die lokale Gewichtungsmatrix $\mathbf{Q}_k$ zu bestimmen, die gegeben ist durch

$$\mathbf{Q}_k = \operatorname{diag}\Big(\Phi_k(\mathbf{Z}(1,:)) \quad \Phi_k(\mathbf{Z}(2,:)) \quad \ldots \quad \Phi_k(\mathbf{Z}(\mathsf{N},:))\Big). \tag{A.18}$$

Weiter wird die Regressionsmatrix $\mathbf{X}$ benötigt, wobei ebenfalls die Daten des $\mathbf{x}$-Regressors zu den Messzeiten $1,2,\ldots,\mathsf{N}$ verwendet werden

$$\mathbf{X} = \begin{pmatrix} 1 & x_1(1) & x_2(1) & \ldots & x_{\mathsf{px}}(1) \\ 1 & x_1(2) & x_2(2) & \ldots & x_{\mathsf{px}}(2) \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_1(\mathsf{N}) & x_2(\mathsf{N}) & \ldots & x_{\mathsf{px}}(\mathsf{N}) \end{pmatrix}. \tag{A.19}$$

Die Parameter des k-ten lokalen Modells sind dann gegeben durch

$$\mathbf{w}_k = \left(\mathbf{X}^T \mathbf{Q}_k \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{Q}_k \mathbf{y}. \tag{A.20}$$

Somit können die Parameter der lokalen Modelle nacheinander bestimmt werden.

A.2.2 Inkrementelles Clustering-Verfahren

In diesem Abschnitt wird das inkrementelle Clustering-Verfahren beschrieben, das zur Erstellung lokaler Modellnetze in dieser Arbeit verwendet wird. Die Idee des angepassten Clustering-Verfahrens besteht aus einer Kombination eines einfaches Clustering-Verfahrens, das von [Pao89; NMG93] entwickelt wurde, und inkrementellen Clustering-Verfahren, wie zum Beispiel aus [Mur94]. Bei dem einfachen Clustering-Verfahren werden die Zentren der Aktivierungsfunktionen anhand der Verteilung der Daten gesetzt. Dabei wird eine Vorgabe eines kleinsten Abstands r_{min} verwendet, um Zentren zu erzeugen, wobei die euklidische Norm verwendet wird. Zur Erzeugung der Zentren werden die Datenpunkte nacheinander verwendet. Ist ein Datenpunkt von allen bisherigen Zentren weiter entfernt als der gegebene kleinste Abstand, wird dieser als ein neues Zentrum hinzugefügt. Im anderen Fall wird dieser Datenpunkt dem ihm am nächsten

gelegenen Zentrum hinzugefügt. Der Vorteil dieses Verfahrens ist, dass auf einfache Weise eine Partitionierung erzielt wird. Nachteil ist, dass es bei höheren Dimensionen schwer ist, einen geeigneten kleinsten Abstand vorzugeben. Zudem orientiert sich das Setzen der Zentren nicht am Ausgangsverhalten, und es werden Zentren an Stellen gesetzt, an denen viele Daten vorhanden sind. Dies sind dabei nicht unbedingt die Stellen, an denen ein stark nichtlineares Verhalten vorhanden ist.

Deshalb wurden Ideen von [Mur94; JK06] aufgegriffen, um durch eine Erweiterung des einfachen Clustering-Verfahrens diesen Nachteil zu beseitigen. Bei den genannten Verfahren werden pro Iterationsschritt an den Stellen im Eingangsraum Zentren hinzufügt, an denen noch Verbesserungsbedarf besteht. Dabei werden die Elemente der Kovarianz-Matrix bestimmt. Besonders bei höherdimensionalen Eingangsräumen ist die Ermittlung der Kovarianz-Matrix numerisch sehr anfällig. Dieser Schritt entfällt bei dem in dieser Arbeit verwendeten Clustering-Verfahren, und es werden Zentren mit einheitlicher Standardabweichung und mit diagonaler Kovarianz-Matrix verwendet. Auf diese Weise lassen sich Seiteneffekte vermeiden, die durch die Normierung entstehen[1]. Das hier verwendete inkrementelle Clustering-Verfahren besteht daher aus folgender Kombination des einfachen Clustering-Verfahrens und Teilen der inkrementellen Clustering-Verfahren:

Pro Iterationsschritt werden die Eingangsdaten nach der Größe des Modellfehlers sortiert. Dabei werden zuerst die Eingangsdaten berücksichtigt, die zu einem hohen Modellfehler führen. Um die Anfälligkeit für Ausreißer und Rauschen zu minimieren, kann ein kleiner prozentualer Anteil der nach ihrem Modellfehler sortierten Eingangsdaten unberücksichtigt bleiben. Durch die inkrementelle Funktionsweise ist es zudem möglich, die Anzahl der Modellparameter zu optimieren. Dazu wird die Modellgüte unter Verwendung der Validierungsdaten verwendet, und es wird am Ende das lokale Modellnetz ausgegeben, das zum

[1] Seiteneffekte entstehen bei stark unterschiedlichen Standardabweichungen, führen zu unerwartetem Verhalten und sind ausführlich in [Mur94; Nel01] beschrieben.

kleinsten Validierungsfehler geführt hat. Für das hier verwendete Clustering-Verfahren werden die Zentren nicht normierter Gaußfunktionen bestimmt, die gegeben sind durch

$$\mu_k(\mathbf{z}) = \exp\left(\frac{1}{2\cdot\sigma^2}\sum_{j=1}^{\mathsf{pz}}(z_j - c_{i,j})^2\right). \tag{A.21}$$

Die Dimension des $\mathbf{z}$-Regressors wird mit pz bezeichnet. Zudem wird die einheitliche Standardabweichung σ verwendet, worauf später genauer eingegangen wird. Aus den nicht normierten Gaußfunktionen werden anschließend die normierten Gaußfunktionen berechnet, die dann zur Berechnung der Gewichtungsmatrizen $\mathbf{Q}_k$ verwendet werden können. Die normierten Gaußfunktionen sind gegeben durch

$$\Phi_k(\mathbf{z}) = \frac{\mu_k(\mathbf{z})}{\sum_{j=1}^{\mathsf{M}}\mu_j(\mathbf{z})}. \tag{A.22}$$

Das inkrementelle Clustering-Verfahren, das zur Bestimmung der Zentren der nicht normierten Gaußfunktionen verwendet wird, zeichnet sich durch folgende vier Eigenschaften aus:

1. Die Partitionierung kann numerisch effizient durchgeführt werden, auch wenn die Dimension des $\mathbf{z}$-Regressors größer ist und viele Abtastzeitpunkte bei den Simulationsdaten verwendet werden.
2. Durch das inkrementelle Vorgehen beim Erstellen der lokalen Modelle kann der Verlauf des nichtlinearen Verhaltens berücksichtigt werden. Das bedeutet, dass lokale Modelle im Eingangsraum dort neu entstehen, wo die Modellgüte noch verbessert werden muss. Bei diesem Clustering-Verfahren werden die Zentren nicht normierter Gaußfunktionen erstellt, und durch anschließende Normierung der erstellten nicht normierten Gaußfunktionen entstehen die Aktivierungsfunktionen, die den Raum partitionieren. Durch die Normierung ist die Erstellung der Aktivierungs-

funktionen gekoppelt. Durch das inkrementelle Vorgehen kann diese nichtlineare Kopplung der Aktivierungsfunktionen zum Teil mitberücksichtigt werden.

3. Es kann mehr als nur ein lokales Modell pro Iteration hinzugefügt werden, um den Aufwand bei der Modellerstellung zu reduzieren. Gerade bei höheren Dimensionen kann es vorteilhaft sein, pro Schritt mehrere lokale Modelle hinzuzufügen, da hier häufig durch das Hinzufügen eines Modells die Modellgüte nur geringfügig verbessert wird.
4. Durch die Verwendung von Gaußfunktionen mit einer einheitlichen Standardabweichung können Seiteneffekte vermieden werden. Die Auswirkung von Seiteneffekten ist ausführlich in [Mur94] beschrieben. Eine weitere Begründung ist, dass es bei höheren Dimensionen sehr aufwändig wird, eine Vorschrift zur Berechnung der Elemente der Kovarianz-Matrix zu finden. Dies gilt auch dann, wenn nur Elemente auf der Diagonalen zugelassen werden. In Abbildung A.9 ist zu sehen, dass durch die Verwendung einer einheitlichen Standardabweichung ebenso eine gute Partitionierung möglich ist.

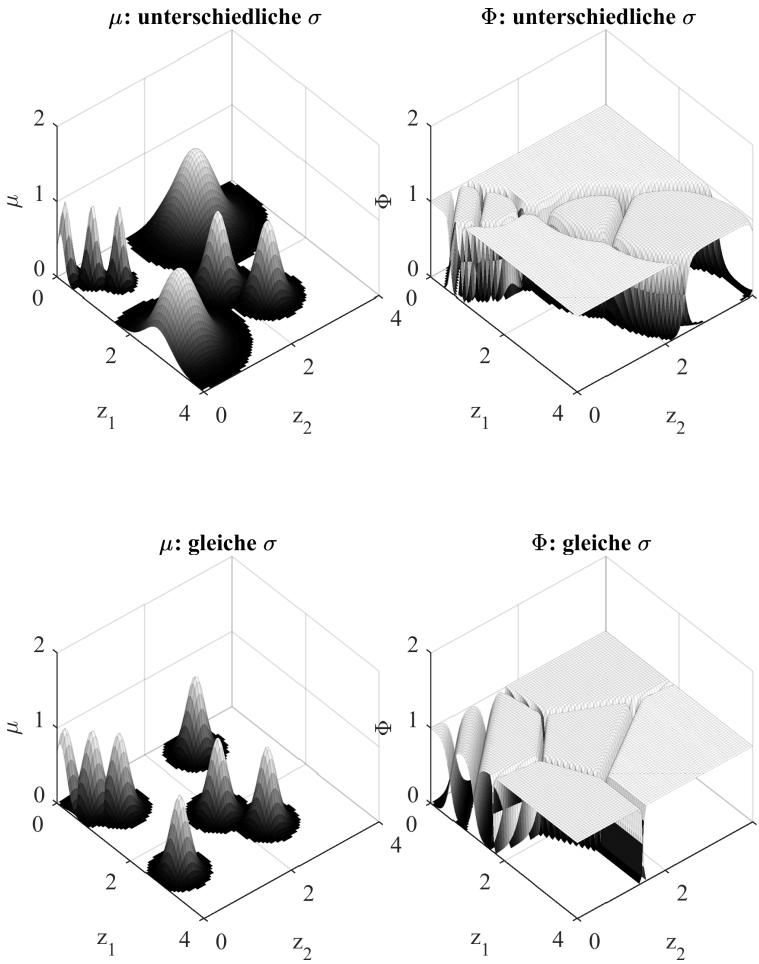


Abbildung A.9: Auswirkung einer einheitlichen Standardabweichung bei Gaußfunktionen

Im Folgenden wird auf den Ablauf des inkrementellen Clustering-Verfahrens näher eingegangen. Dabei startet das Verfahren mit einem lokal-linearen Modell,

wobei die zugehörigen Modellparameter durch die Berechnung der Pseudo-Inversen bestimmt werden können. Anschließend wird der Modellausgang $\hat{y}$ des erstellten linearen Modells verwendet, um die Modellabweichungen $\varepsilon = \text{abs}(\hat{y} - y)$ zu bestimmen.
Für die weiteren Schritte des inkrementellen Clustering-Verfahrens werden noch weitere Angaben benötigt:

- Kleinstmöglicher Abstand r_{min} zwischen zwei Zentren und einheitliche Standardabweichung σ
- Maximale Anzahl der Inkremente $\text{Ink}_{\text{max}} = \text{L}$
- Maximale Anzahl der Modellparameter $n_{\text{w,max}}$
- Anzahl NewM der Zentren, die pro Inkrement hinzugefügt werden
- Globaler Validierungsfehler ξ, der zum Finden des besten lokalen Modellnetzes verwendet wird.

Anschließend werden wie folgt weitere lokale Modelle hinzugefügt:

1. Die absoluten Modellabweichungen $\varepsilon = \text{abs}(\hat{y} - y)$ werden nach der Größe der Modellabweichungen sortiert. Dabei entsteht eine Liste **zidx**, die die Modellabweichungen mit den Daten des **z**-Regressors verknüpft. Somit wird in der Liste der Datenpunkt des **z**-Regressors mit dem größten Modellfehler verknüpft. Um robuster gegen Rauschen oder Ausreißer zu sein, wird das erste Prozent dieser Liste nicht berücksichtigt.
2. Diese Liste wird nun folgendermaßen für das Hinzufügen neuer Zentren verwendet:
   a) Verwendung des ersten zulässigen Datenpunkts des **z**-Regressors als potentielles neues Zentrum **v**
   b) Berechnung des kleinsten Abstands d_{min} von den vorhandenen Zentren der nicht normierten Gaußfunktionen und dem potentiellen Zentrum **v**

c) Durchführung einer Fallunterscheidung (siehe [Pao89; NMG93]):
   i. Ist der kleinste Abstand d_{min} zu den bisher ermittelten Zentren größer als der vorgegebene Schwellwert r_{min} des Abstands, dann wird das potentielle Zentrum als neues Zentrum zur Liste der Zentren hinzugefügt, wobei sich die Anzahl der Zentren von M um eins auf $\mathsf{M}+1$ erhöht.
   Zudem wird der Index n dieses Zentrums mit eins initialisiert. Dieser Index wird für den anderen Fall der Fallunterscheidung benötigt.
   ii. Im anderen Fall wird das potentielle Zentrum dem ihm am nächsten liegenden Zentrum hinzugefügt. Dabei wird nach und nach der ein angepasster gewichteter Mittelwert gebildet, so dass nach einigen Iterationen ein stabiles Zentrum entsteht. Dazu wird der oben eingeführte Index n benötigt, der angibt, wie viele Zentren bereits hinzugefügt worden sind. Dieser gewichtete Mittelwert ist gegeben durch

$$c_{\text{neu}} = \frac{n \cdot c_{\text{alt}} + z}{n+1}. \tag{A.23}$$

d) Das Hinzufügen von Zentren endet, wenn die maximal erlaubte Anzahl NewM erreicht ist, die pro Iterationsschritt hinzugefügt werden darf. Bei zu groß gewähltem kleinsten Abstand r_{min} kann es jedoch sein, dass alle Datenpunkte des $\mathbf{z}$-Regressors betrachtet wurden und es trotzdem nicht gereicht hat, die gewünschte Anzahl von Zentren hinzuzufügen.
   Kann überhaupt kein Zentrum mehr hinzugefügt werden, führt dies zum Abbruch des inkrementellen Clustering-Verfahrens. Dieser Abbruch ist jedoch nicht in Abbildung A.10 zu sehen.

3. Anschließend werden die Gaußfunktionen mit den neu hinzugefügten Zentren berechnet und normiert. Dann werden die Modellparameter der

M lokalen Modelle durch eine lokale Schätzung berechnet (siehe Gleichung A.20). Diese werden dazu verwendet, den Modellausgang $\hat{y}$ und die Modellabweichungen $\varepsilon = \text{abs}(\hat{y} - y)$ neu zu bestimmen. Dabei wird auch der RMSE der Validierungsdaten bestimmt. Ist dieser kleiner als der bisherige globale Validierungsfehler ξ, wird das erstellte lokale Modellnetz mit seinen Parametern $\mathbf{w}$ der lokalen Modelle, seinen Zentren $\mathbf{c}_1, \mathbf{c}_1, \ldots, \mathbf{c}_M$ und der gewählten einheitlichen Skalierung als neues bestes lokales Modellnetz BLM gespeichert. Zusätzlich wird der neue Validierungsfehler gespeichert.

4. Anschließend wird mit den neu berechneten Modellabweichungen ε von vorne begonnen, wobei die Liste, die die Datenpunkte des $\mathbf{z}$-Regressors mit der Modellabweichung verknüpft, aktualisiert wird.
   Diese Schritte werden solange wiederholt, bis entweder eine untere Fehlerschranke oder die maximale Anzahl erlaubter Iterationen erreicht ist. Wie bereits erwähnt führt es auch zum Abbruch, wenn keine neuen Zentren mehr hinzugefügt werden können.

Durch den beschriebenen Ablauf werden die Zentren und Modellparameter des lokalen Modellnetzes ausgegeben, das den niedrigsten Validierungsfehler hatte. Somit findet beim inkrementellen Clustering-Verfahren eine Optimierung der Anzahl der Modellparameter statt.
In Abbildung A.10 ist ein Ablaufdiagramm des inkrementellen Clustering-Verfahrens zu sehen, in dem die zuvor beschriebenen Schritte dargestellt sind.

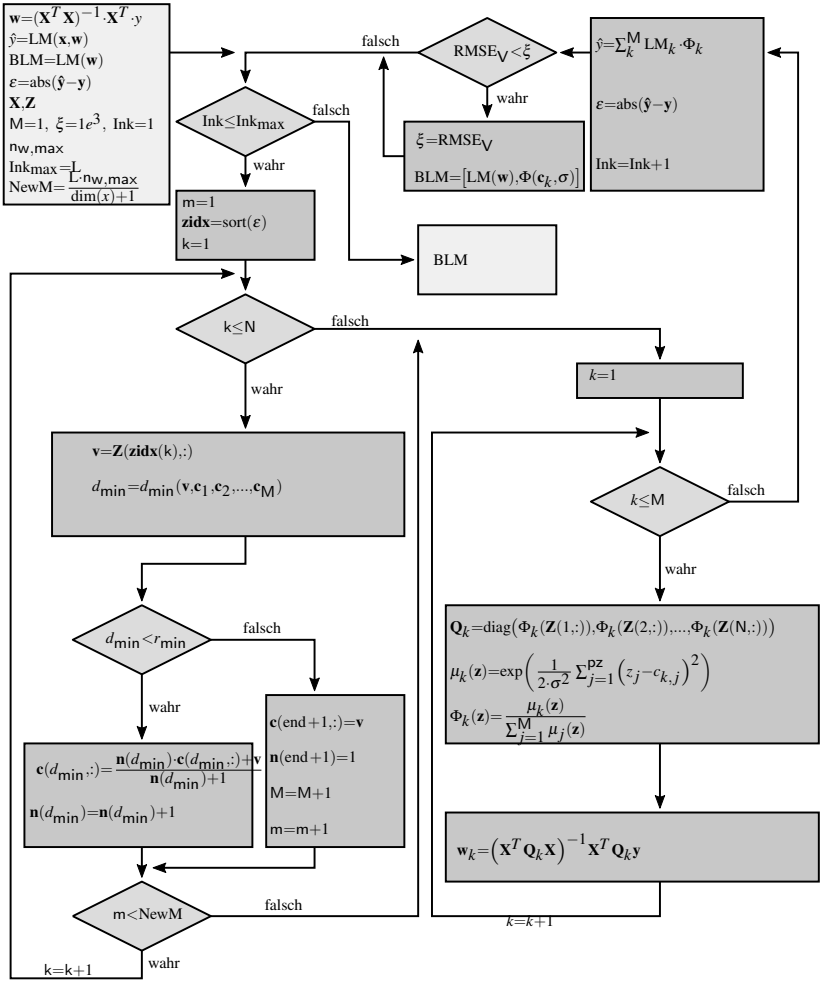


Abbildung A.10: Ablaufdiagramm des inkrementellen Clustering-Verfahrens

Zum Abschluss wird noch auf ein Beispiel eingegangen, das in einer Dimension illustriert, wie die Zentren nacheinander gesetzt werden. Dazu wurden die Testfunktionen aus Gleichung (A.24) verwendet. Dabei ist Abbildung A.11 zu erkennen wie die Zentren nacheinander gesetzt werden. Dabei wurde pro Iterationsschritt ein Zentrum hinzugefügt.
In Abbildung A.12 wird deutlich, wie die Approximation der gegebenen Kurve immer besser wird. Nach 7 Iterationen wurde abgebrochen, da ein vorgegebener minimaler Fehler unterschritten wurde.

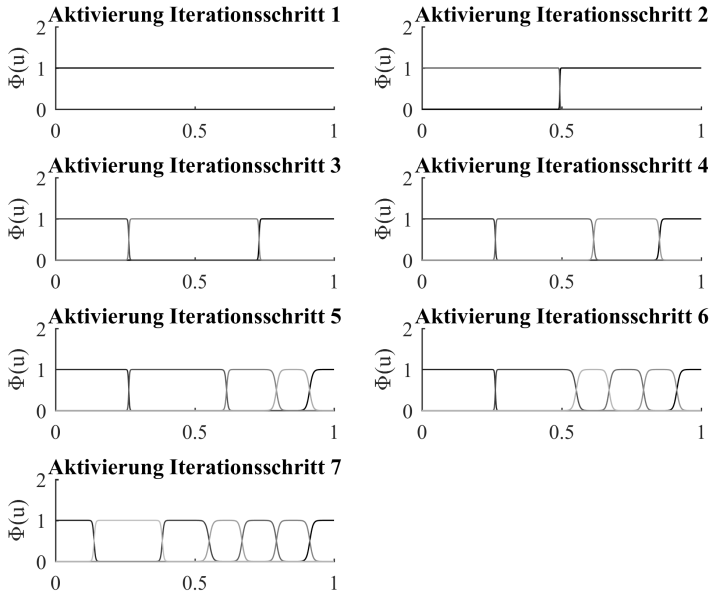


Abbildung A.11: Gültigkeitsfunktionen zu jedem Iterationssschritt

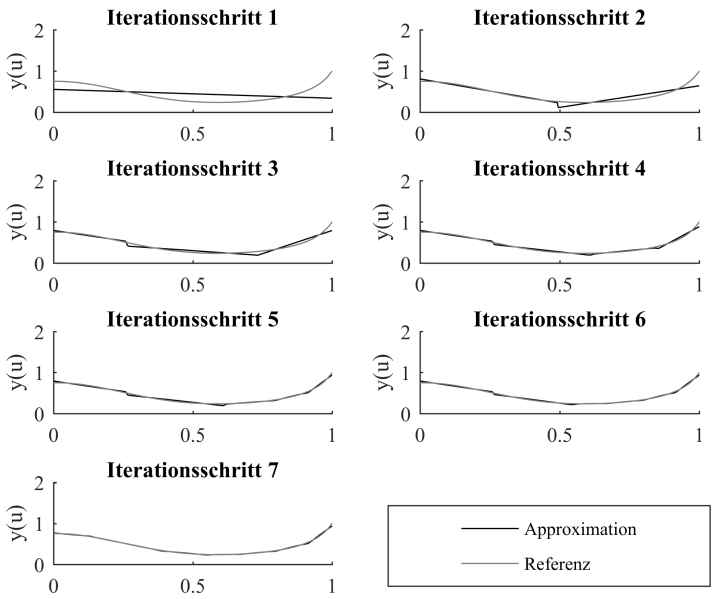


Abbildung A.12: Gewichtete Summe zu jedem Iterationssschritt

A.2.3 Vergleich des inkrementellen Clustering-Verfahrens mit inkrementellen Baumkonstruktionsverfahren

In diesem Abschnitt wird das zuvor beschriebene Clustering-Verfahren verwendet, um eine Testfunktion aus [Har14] zu approximieren. Diese Testfunktion wurde dort verwendet, um inkrementelle Baumkonstruktionsverfahren zu vergleichen. Im Folgenden wird gezeigt, dass das hier vorgestellte Clustering-Verfahren vergleichbare Werte erzielt. Die Testfunktion ist für die Dimension p des Eingangsraums gegeben durch

$$y(k) = \frac{1}{1 + 10 \cdot \sum_{j=1}^{\mathsf{p}} \frac{1}{\mathsf{p}} (1 - u_j(k))} + \frac{29}{44} \exp\left(-\frac{1}{2} \sum_{i=1}^{\mathsf{p}} (4 \cdot u_i(k))^2 \right). \quad \text{(A.24)}$$

Die Funktion in [Har14] wurde bis zur Dimension $\mathsf{p} = 5$ verwendet, und es wurden jeweils $5^5 = 3125$ raumfüllende Datenpunkte zur Modellerstellung erzeugt. Als Maß für die Modellgüte wird der normierte RMSE (NRMSE) verwendet, der gegeben ist durch

$$\begin{aligned} J &= \sqrt{\frac{\sum_{k=1}^{\mathsf{N}} (y(k) - \hat{y}(k))^2}{\sum_{k=1}^{\mathsf{N}} (y(k) - \bar{y})^2}} \\ \text{mit } \bar{y} &= \frac{1}{\mathsf{N}} \sum_{k_1}^{\mathsf{N}} y(k). \end{aligned} \quad \text{(A.25)}$$

Bei diesem Fehlermaß wird mit der Standardabweichung des Prozessausgangs normiert. Es wird an dieser Stelle betont, dass dieses Maß nur in diesem Beispiel verwendet wird, um die Verfahren miteinander vergleichen zu können. Da beim inkrementellen Clustering-Verfahren nicht fest vorgegeben ist, wie der kleinste Abstand r_{min} und die Standardabweichung σ gewählt werden sollen, wird für diese beiden Parameter eine Strukturoptimierung benötigt. Deshalb wurde die aus Kapitel 5 beschriebene Liniensuche verwendet:

- Suchraum für r_{min}
- Suchraum für σ.

Außerdem wurde eine feste Anzahl von Zentren vorgegeben, die pro Iterationsschritt hinzukamen. Zusätzlich wurde hier als Abbruchkriterium die unterste Fehlerschranke gewählt, um die Modellerstellung durch das Clustering-Verfahren mit den inkrementellen Baumkonstruktionsverfahren vergleichen zu können. Die Anzahl von Zentren, die pro Inkrementschritt hinzukommen, ist gleich der Dimension der Testfunktion, also zum Beispiel bei Dimension $\mathsf{p} = 2$ kommen 2 Zentren pro Iteration hinzu.

In Abbildung A.13 ist der Verlauf der Fehler mit steigender Anzahl von Modellparametern zu sehen. Zusätzlich sind die jeweiligen Zeiten zur Parameteroptimierung angegeben.

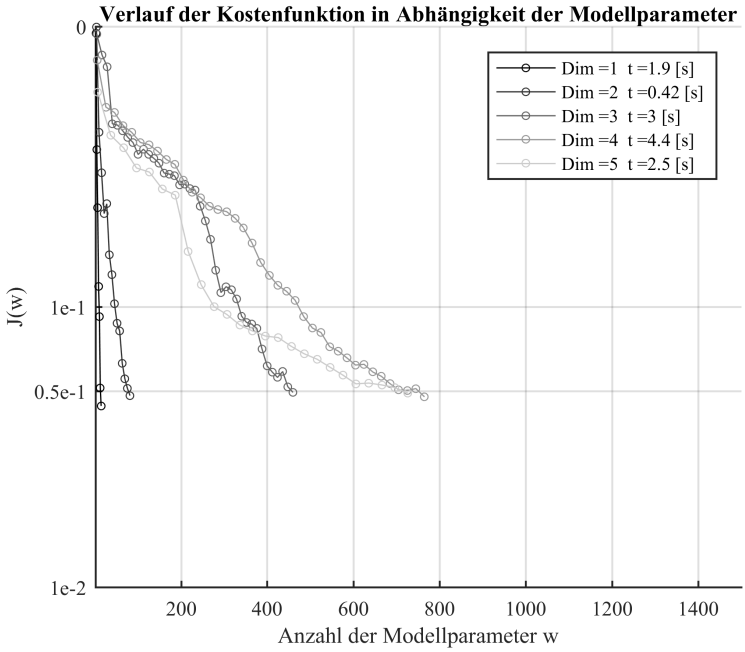


Abbildung A.13: Fehlerverlauf der Testfunktion bei verschiedenen Dimensionen des Eingangsraums

Zur Berechnung wurde ein PC mit einem Intel Pentium *I*7 M620 und einem Arbeitsspeicher mit 4gb RAM verwende, was vergleichbar ist mit den in [Har14] verwendeten Ressourcen. Bei der Anzahl der Modellparameter, die benötigt werden, um die geforderte Modellgüte zu erreichen, liegt das beschriebene Clustering-Verfahren zwischen dem HILOMOT-Verfahren und dem LOLIMOT-

Verfahren. Die Zeiten, die zur Bestimmung der LM-Netze durch das inkrementelle Clustering-Verfahren benötigt werden, steigen nicht mit der Dimension der Testfunktion. Das liegt an den stochastischen Elementen der Liniensuche und bedeutet, dass die Dauer zur Erreichung der Fehlerschranke variieren kann. Insgesamt sind die Zeiten bei diesem Beispiel jedoch mit denen der inkrementellen Verfahren vergleichbar.

A.2.4 Verlauf des inkrementellen Clustering-Verfahrens

In Abbildung A.14 ist der Verlauf des inkrementellen Clustering-Verfahrens dargestellt, wie es zur Modellierung des in Kapitel 5 nichtlinearen dynamischen Systems verwendet wurde. Dabei ist zu sehen, wie sich mit steigender Anzahl von Modellparametern die Modellgüte verbessert. Außerdem ist zu erkennen, dass Trainings- und Validierungsfehler nahezu identisch sind und der Validierungsfehler hier mit steigender Anzahl von Modellparametern nicht merklich ansteigt. Somit kann hier die maximale Anzahl von Modellparametern verwendet werden, ohne dass eine Überanpassung stattfindet. Zusätzlich ist dabei zu beachten, dass hier der Verlauf des Clustering-Verfahrens zu sehen ist, welches das gefundene Minimum der Liniensuchen ergeben hat. Das bedeutet, dass hier die besten Werte der Hyperparameter verwendet wurden.

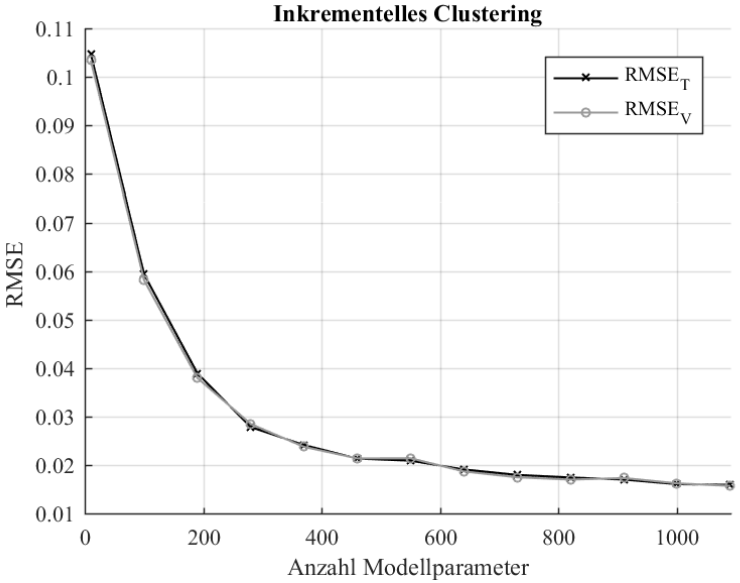


Abbildung A.14: Verlauf des inkrementellen Clustering-Verfahrens mit den besten, fest vorgegebenen Hyperparametern

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

[Abe+11] A. Abel u. a. „Hardware-in-the-Loop-Simulation mit detaillierten physikalischen Getriebemodellen“. In: *Virtual Powertrain Creation*. Hrsg. von MTZ. Springer, 2011.

[ABS02] J. Abonyi, R. Babuska und F. Szeifert. „Modified Gath-Geva fuzzy clustering for Identification of Takagi-Sugeno fuzzy models“. In: *Transaction on Systems, Man, and Cybernetics, Part B: Cybernetics*. Hrsg. von IEEE. Bd. 32. 5. 2002, S. 612–621.

[Alv08] G. Alvermann. „Virtuelle Getriebeabstimmung“. Dissertation. TU Braunschweig, 2008.

[Amm97] D. Ammon. *Modellbildung und Systementwicklung in der Fahrzeugdynamik*. B. G. Teubner Stuttgart, 1997.

[Ant05] A.C. Antoulas. „An overview of approximation methods for large-scale dynamical systems“. In: *Annual Reviews in Control* 29.2 (2005), S. 181–190.

[Ast04] P. Astrid. „Reduction of Process Simulation Models: a proper orhogonal decompostion approach“. Diss. Technische Universiteit Eindhoven, 2004.

[Bag+08] B. Bagot u. a. „Schaltqualitätsoptimierung von Automatikgetrieben“. In: *ATZ - Automobiltechnische Zeitschrift* 110.5 (2008), S. 404–411.

[BBF14] U. Baur, P Benner und L Feng. *Model order reduction for linear and nonlinear systems: a system-theoretic perspective*. Preprint MPIMD/14-07. Max Planck Institute Magdeburg, März 2014.

[BC85] S. Boyd und L. Chua. „Fading memory and the problem of approximating nonlinear operators with Volterra series“. In: *IEEE Transactions on Circuits and Systems* 32.11 (Nov. 1985), S. 1150–1161.

[BSH14] M. Bachinger, M. Stolz und M. Horn. „Fixed step clutch modeling and simulation for automotive real-time applications“. In: *American Control Conference (ACC), 2014*. Juni 2014, S. 2593–2599.

[BV96] R. Babuska und H.B. Verbruggen. „An overview of fuzzy modeling for control“. In: *Control Engineering Practice* 4.11 (1996), S. 1593–1606.

[BYA12] L. Belmon, Y. Yan und A. Abel. „Modelling and Simulation of DCT Gearshifting for Real-Time and High-Fidelity Analysis“. In: *Proceedings of the FISITA 2012 World Automotive Congress*. Hrsg. von SAE-China und FISITA. Springer Berlin Heidelberg, 2012, S. 399–411.

[Cav+12] N. Cavina u. a. „Development of a Dual Clutch Transmission Model for Real-Time Applications“. In: *IFAC Workshop on Engine and Powertrain Control, Simulation and Modeling*. Bd. 3. 1. 2012, S. 440–447.

[Cav+13] N. Cavina u. a. „Development and Implementation of Hardware in the Loop Simulation for Dual Clutch Transmission Control Units“. In: *SAE Int. J. Passeng. Cars - Electron. Electr. Syst.* 6 (Aug. 2013), S. 458–466.

[Chr+11] E. Chrisofakis u. a. „Simulation-based development of automotive control software with Modelica“. In: *Proceedings of the 8th Modelica Conference*. 2011.

[CNG13] M. Cabrera Cano, D. Neumerkel und M. Geimer. „Lokale Modellnetze zur Verkürzung der Simulationszeit des Schaltverhaltens von Automatikgetriebemodellen“. In: *Proceedings of the 23rd Workshop on Computational Integelligence*. Dortmund: KIT Scientific Publishing, Dez. 2013, S. 145–164.

[CNG14] M. Cabrera Cano, D. Neumerkel und M. Geimer. „Black-Box-Modelle zur systematischen numerischen Vereinfachung von physikalischen Automatikgetriebemodellen“. In: *Tagungsband des 4. Kongresses zu Einsatz und Validierung von Simulationsmethoden für die Antriebstechnik*. 2014.

[Dro03] S. Dronka. „Die Simulation gekoppelter Mehrkörper- und Hydraulikmodelle mit Erweiterung für Echtzeitsimulation“. Dissertation. TU Dresden, 2003.

[Dür+14] C. Dürr u. a. „Nine-Speed Automatic Transmission 9G-Tronic by Mercedes-Benz“. In: *ATZ worldwide* 116.1 (2014), S. 20–25.

[DW10] A. Dankers und D.T. Westwick. „A convex method for selecting optimal Laguerre filter banks in system modelling and identification“. In: *American Control Conference (ACC), 2010*. Juni 2010, S. 2694–2699.

[Dym] Dymola. www.dymola.com - Zuletzt abgerufen: 08.06.2016.

[Elm+04] H. Elmqvist u. a. „Realtime Simulation of Detailed Vehicle and Powertrain Dynamics“. In: *Proceedings of the SAE World Congress*. SAE International, 2004.

[EMO02] H. Elmqvist, S. E. Mattsson und H. Olsson. „New Methods for Hardware-in-the-Loop-Simulation of Stiff Modells“. In: *Proceedings of the 2nd International Modelica Conference*. März 2002, S. 59–64.

[Fis+12] R. Fischer u. a. *Das Getriebebuch*. Der Fahrzeugantrieb. Springer, 2012.

[GKL06] M. Geimer, T. Krüger und P. Linsel. „Co-Simulation, gekoppelte Simulation oder Simulatorkopplung? “ In: *O+P - Ölhydraulik und Pneumatik* 50.11-12 (2006), S. 572–576.

[Gru10] W.-D. Gruhle. „Steuerung und Regelung von Automatikgetrieben“. In: *Elektronisches Management motorischer Fahrzeugantriebe*. Hrsg. von Rolf Isermann. Vieweg+Teubner, 2010, S. 288–305.

[Güh02] C Gühmann. „Testautomatisierung und Modellbildung für die Hardware-in-the-Loop Simulation“. In: *IIR Tagung Versuch, Test und Simulation*. 2002.

[Güh03] C Gühmann. „Modellbildung und Simulation für die Funktions- und Softwareentwicklung in der Automobilelektronik“. In: *ASIM-Jahrestagung 2003, Magdeburg*. 2003.

[Hal+99] C. Halfmann u. a. „Adaptive Echtzeitmodelle für die Kraftfahrzeugdynamik“. In: *ATZ - Automobiltechnische Zeitschrift* 101.12 (1999), S. 994–1001.

[Har+12] B. Hartmann u. a. „LMNTool - Toolbox zum automatischen Trainieren lokaler Modellnetze“. In: *Proceedings of the 22nd Workshop Computational Intelligence*. Dortmund: KIT Scientific Publishing, Dez. 2012, S. 341–355.

[Har14] B. Hartmann. „Lokale Modellnetze zur Identifikation und Versuchsplanung nichtlinearer Systeme“. Dissertation. Universität Siegen, 2014.

[Hay09] S.S. Haykin. *Neural Networks and Learning Machines*. Neural networks and learning machines Bd. 10. Prentice Hall, 2009.

[HH03] C. Halfmann und H. Holzmann. *Adaptive Modelle für die Kraftfahrzeugdynamik*. Engineering online library. Springer Berlin Heidelberg, 2003.

[Hof03] S. Hofmann. „Identifikation von nichtlinearen mechatronischen Systemen auf der Basis von Volterra-Reihen“. Dissertation. TU München, 2003.

[Ise08a] R. Isermann. „Grundlagen der theoretischen Modellbildung technischer Prozesse“. In: *Mechatronische Systeme*. Springer Berlin Heidelberg, 2008, S. 47–117.

[Ise08b] R. Isermann. „Integrierte mechanisch-elektronische Systeme“. In: *Mechatronische Systeme*. Springer Berlin Heidelberg, 2008, S. 1–44.

[Ise08c] Rolf Isermann. „Mechatronic systems – Innovative products with embedded control“. In: *Control Engineering Practice* 16.1 (2008), S. 14–29.

[Jan09] K. Janschek. *Systementwurf mechatronischer Systeme: Methoden – Modelle – Konzepte*. Springer Berlin Heidelberg, 2009.

[JK03] M. Jelali und A. Kroll. *Hydraulic Servo-systems: Modelling, Identification and Control*. Advances in Industrial Control. Springer London, 2003.

[JK06] S. Jakubek und N. Keuth. „A local neuro-fuzzy network for high-dimensional models and optimization “. In: *Engineering Applications of Artificial Intelligence* 19.6 (2006). Special Section on Innovative Production Machines and Systems, S. 705–717.

[JSM00] T. Johansen, R. Shorten und R. Murray-Smith. „On the Interpretation and Identification of Dynamic Takagi-Sugeno Fuzzy Models“. In: *Transactions on Fuzzy Systems*. Hrsg. von IEEE. Bd. 8. 3. 2000, S. 297–313.

[Jud+95] A Juditsky u. a. „Nonlinear black-box models in system identification: Mathematical foundations“. In: *Automatica* 31.12 (1995). Trends in System Identification, S. 1725–1750.

[Kah13] S. Kahlbau. „Mehrkriterielle Optimierung des Schaltablaufs von Automatikgetrieben“. Dissertation. BTU Cottbus, 2013.

[KG00] F. Kessler und J. Gebert. „Testautomatisierung und Antriebsstrangmodellierung an HIL-Steuergeräteprüfständen in der BMW-Getriebeentwicklung“. In: *ATZ - Automobiltechnische Zeitschrift* 102.5 (2000), S. 312–323.

[Kir07] E. Kirchner. „Architektur, Komponenten und Baugruppen automatisch schaltender PKW-Getriebe“. In: *Leistungsübertragung in Fahrzeuggetrieben*. Hrsg. von Eckhard Kirchner. VDI-Buch. Springer Berlin Heidelberg, 2007, S. 265–422.

[Koc01] J. Koch. „Modellbildung und Simulation eines Automatikgetriebes zur Optimierung des dynamischen Schaltablaufs“. Dissertation. Universität Stuttgart, 2001.

[KR88] B.W. Kernighan und D.M. Ritchie. *The C Programming Language*. Prentice-Hall software series. Prentice Hall, 1988.

[Kuc+10] A. Kuchle u. a. *Automotive Transmissions: Fundamentals, Selection, Design and Application*. Springer, 2010.

[LG03] M Lindemann und C Gühmann. „VeLoDyn - Ein Werkzeug zur Triebstrangsimulation von Kraftfahrzeugen“. In: *1. Tagung Simulation und Test in der Funktions- und Softwareentwicklung*. 2003.

[Lju98] L. Ljung. *System Identification: Theory for the User*. Pearson Education, 1998.

[LW07] H. Lutz und W. Wendt. *Taschenbuch der Regelungstechnik*. Deutsch, 2007.

[Map] MapleSim. http://www.maplesoft.com/products/maplesim/ - zuletzt abgerufen am 08.06.2016.

[Mat] Matlab. www.mathworks.de - Zuletzt abgerufen: 08.06.2016.

[Mat13] Felix Matthies. „Beitrag zur Modellbildung von Antriebssträngen für Fahrbarkeitsuntersuchungen“. Dissertation. TU Berlin, 2013.

[MJ95] R. Murray-Smith und T.A. Johansen. „Local learning in local model networks“. In: *Fourth International Conference on Artificial Neural Networks*. Juni 1995, S. 40–46.

[MJ97] R. Murray-Smith und T. A. Johansen. „Side-Effects of Normalising Basis Functions in Local Model Networks“. In: *Multiple Model Approaches To Nonlinear Modelling And Control*. Series in Systems and Control. Taylor & Francis, 1997. Kap. 8, S. 211–229.

[Mod] Modelica. www.modelica.org - Zuletzt abrufen: 08.06.2016.

[Mur94] R. Murray-Smith. „A Local Model Network Approach to Nonlinear Modeling“. Diss. University of Strathclyde, 1994.

[Nel01] O. Nelles. *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models*. Engineering online library. Springer, 2001.

[Nel06a] O. Nelles. „Axes-oblique partitioning strategies for local model networks“. In: *2006 IEEE International Symposium on Intelligent Control*. Munich, Germany, Okt. 2006, S. 2378–2383.

[Nel06b] O. Nelles. „Neuronale Netze: Eine Übersicht“. In: *Informationsfusion in der Mess- Sensortechnik*. Universitätsverlag Karlsruhe, 2006.

[Nil09] O. Nilsson. „On Modeling and Nonlinear Model Reduction in Automotive Systems“. Diss. Lund University, 2009.

[NMG93] D. Neumerkel, R. Murray-Smith und H. Gollee. „Modelling Dynamic Processes with Clustered Time-Delay Neurons“. In: *International Joint Conference on Neural Networks*. 1993.

[OHK09] M. Otter, A. Haumer und Ch. Kral. „Objektorientierte Modellierung und Simulation von Antriebssystemen“. In: *Elektrische Antriebe – Regelung von Antriebssystemen*. Springer Berlin Heidelberg, 2009, S. 1049–1165.

[Pao89] Y.-H. Pao. „Unsupervised Learning Based on Discovery of Cluster Structure, Appendix B“. In: *Adaptive Pattern Recognition and Neural Networks*. Addison Wesley, 1989.

[Pfe08a] F. Pfeiffer. „Dynamics of Hydraulic Systems“. In: *Mechanical System Dynamics*. Hrsg. von Friedrich Pfeiffer. Bd. 40. Lecture Notes in Applied and Computational Mechanics. Springer Berlin Heidelberg, 2008, S. 187–212.

[Pfe08b] F Pfeiffer. „Power Transmission“. In: *Mechanical System Dynamics*. Hrsg. von Friedrich Pfeiffer. Bd. 40. Lecture Notes in Applied and Computational Mechanics. Springer Berlin Heidelberg, 2008, S. 213–328.

[Ram+99] S. Raman u. a. „Design and implementation of HIL simulators for powertrain control system software development“. In: *Proceedings of the American Control Conference*. Bd. 1. 1999, 709–713 vol.1.

[RB13] I. Rennert und B. Bundschuh. *Signale und Systeme: Einführung in die Systemtheorie*. Carl Hanser Verlag GmbH & Company KG, 2013.

[RCT09] Anton Rink, Emmanuel Chrisofakis und Mugur Tatar. „Automatisierter Test für Softwaremodule“. In: *ATZelektronik* 4.6 (2009), S. 36–41.

[Rei10] K. Reif. *Konventioneller Antriebsstrang und Hybridantriebe*. Vieweg und Teubner Verlag, 2010.

[Ren12] Z. Ren. „Zur Identifikation mechatronischer Stellglieder mit Reibung bei Kraftfahrzeugen“. Dissertation. Universität Kassel, 2012.

[RHW86] D. E. Rumelhart, G. E. Hinton und R.J. Williams. „Learning internal representations by error propagation“. In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition Vol.1*. MIT Press, Cambridge, 1986.

[Rie+04] A. Riel u. a. „Modellierung von Fahrzeug und Antriebsstrang im gesamten Entwicklungsprozess“. In: *ATZ - Automobiltechnische Zeitschrift* 106.6 (2004), S. 522–531.

[RKS13] Z. Ren, A. Kroll und M. Sofsky. „Zur physikalischen und datengetriebenen Modellbildung von Systemen mit Reibung: Methoden und Anwendung auf Kfz-Drosselklappen“. In: *at - Automatisierungstechnik Methoden und Anwendungen der Steuerungs-, Regelungs- und Informationstechnik* 62.3 (März 2013), S. 155–171.

[Sba97] D. Sbarbaro. „Local Laguerre Models“. In: *Multiple Model Approaches to Nonlinear Modelling And Control*. Taylor & Francis, 1997.

[SBB02] C. Schlegel, M. Bross und P. Beater. „HIL-Simulation of the Hydraulics and Mechanics of an Automatic Gearbox“. In: *Proceedings of the 2nd Internation Modelica Conference*. März 2002, S. 67–75.

[Sch08] Schilders, W. H. A. and Vorst, H. A. and Rommes, J., Hrsg. *Model Order Reduction: Theory, Research Aspects and Applications*. Springer Berlin Heidelberg, 2008.

[Sch10] D Schröder. *Intelligente Verfahren: Identifikation und Regelung nichtlinearer Systeme*. Springer Berlin Heidelberg, 2010.

[Sch13] J. Schröter. „Das erweiterte X-in-the-Loop-Framework zur durchgängigen Integration von Optimierungsverfahren in den Produktentwicklungsprozess am Beispiel der Entwicklung energieeffizienter Fahrzeuge“. Dissertation. Karlsruher Institut für Technologie, 2013.

[Sim] SimulationX. http://www.simulationx.com/de/ - zuletzt abgerufen am 08.06.2016.

[Sjö+95] J. Sjöberg u. a. „Nonlinear black-box modeling in system identification: a unified overview“. In: *Automatica* 31.12 (1995). Trends in System Identification, S. 1691–1724.

[SO05] C. Schweiger und M. Otter. „Modelica-Modellbibliothek zur Simulation der Dynamik von Schaltvorgängen bei Automatikgetrieben“. In: *Dynamik und Regelung von automatischen Getrieben, VDI-Schwingungstagung 2005*. VDI. Stuttgart: VDI Verlag GmbH, Nov. 2005, S. 105–124.

[SWP12] K. Strehmel, R. Weiner und H. Podhaisky. *Numerik gewöhnlicher Differentialgleichungen: Nichtsteife, steife und differential-algebraische Gleichungen*. SpringerLink : Bücher. Vieweg+Teubner Verlag, 2012.

[SZ13] J. Schäuffele und T. Zurawka. *Automotive Software Engineering: Grundlagen, Prozesse, Methoden und Werkzeuge*. ATZ/MTZ-Fachbuch. Vieweg+Teubner Verlag, 2013.

[Til01] M. Tiller. *Introduction to Physical Modeling with Modelica*. The Springer International Series in Engineering and Computer Science. Springer US, 2001.

[TR12] L. D. Tufa und M. Ramasamy. „System identification using orthonormal basis filters“. In: *Frontiers in Advandes Control Systems*. Hrsg. von Dr. Ginalber Luiz Serra. InTech, 2012.

[TS85] T. Takagi und M. Sugeno. „Fuzzy identification of systems and its applications to modeling and control“. In: *IEEE Transactions on Systems, Man and Cybernetics* SMC-15.1 (Jan. 1985), S. 116–132.

[TZC11] G. Tao, T. Zhang und H. Chen. „Modeling of shift hydraulic system for automatic transmission“. In: *International Conference on Consumer Electronics, Communications and Networks*. Apr. 2011, S. 474–477.

[Wah91a] B. Wahlberg. „Identification of resonant systems using Kautz filters“. In: *Proceedings of the 30th IEEE Conference on Decision and Control*. Dez. 1991, 2005–2010 vol.2.

[Wah91b] B. Wahlberg. „System identification using Laguerre models“. In: *IEEE Transactions on Automatic Control* 36.5 (Mai 1991), S. 551–562.

[Wel79] P.E. Wellstead. *Introduction to Physical System Modelling*. Mathematics in Science and Engineering series. Academic Press, 1979.

[Wie+10] M. Wieczorek u. a. „Wachsende Funktionsumfänge für Steuergeräte von Automatikgetrieben“. In: *ATZ - Automobiltechnische Zeitschrift* 11 (2010), S. 828–833.

[WS03] S. Watechagit und K. Srinivasan. „Modeling and Simulation of a Shift Hydraulic System for a Stepped Automatic Transmission“. In: *SAE Transmission & Driveline Systems Symposium 2003*. Hrsg. von SAE International. 2003.

[ZI08] R. Zimmerschied und R. Isermann. „Regularisierungsverfahren für die Identifikation mittels lokal-affiner Modelle“. In: *Automatisierungstechnik* 56.7 (2008), S. 339–349.

Karlsruher Schriftenreihe Fahrzeugsystemtechnik (ISSN 1869-6058)

Herausgeber: FAST Institut für Fahrzeugsystemtechnik

Die Bände sind unter www.ksp.kit.edu als PDF frei verfügbar oder als Druckausgabe bestellbar.

Band 1 Urs Wiesel
Hybrides Lenksystem zur Kraftstoffeinsparung im schweren Nutzfahrzeug. 2010
ISBN 978-3-86644-456-0

Band 2 Andreas Huber
Ermittlung von prozessabhängigen Lastkollektiven eines hydrostatischen Fahrantriebsstrangs am Beispiel eines Teleskopladers. 2010
ISBN 978-3-86644-564-2

Band 3 Maurice Bliesener
Optimierung der Betriebsführung mobiler Arbeitsmaschinen. Ansatz für ein Gesamtmaschinenmanagement. 2010
ISBN 978-3-86644-536-9

Band 4 Manuel Boog
Steigerung der Verfügbarkeit mobiler Arbeitsmaschinen durch Betriebslasterfassung und Fehleridentifikation an hydrostatischen Verdrängereinheiten. 2011
ISBN 978-3-86644-600-7

Band 5 Christian Kraft
Gezielte Variation und Analyse des Fahrverhaltens von Kraftfahrzeugen mittels elektrischer Linearaktuatoren im Fahrwerksbereich. 2011
ISBN 978-3-86644-607-6

Band 6 Lars Völker
Untersuchung des Kommunikationsintervalls bei der gekoppelten Simulation. 2011
ISBN 978-3-86644-611-3

Band 7 3. Fachtagung
Hybridantriebe für mobile Arbeitsmaschinen. 17. Februar 2011, Karlsruhe. 2011
ISBN 978-3-86644-599-4

Karlsruher Schriftenreihe Fahrzeugsystemtechnik (ISSN 1869-6058)

Herausgeber: FAST Institut für Fahrzeugsystemtechnik

Band 8 Vladimir Iliev
Systemansatz zur anregungsunabhängigen Charakterisierung des Schwingungskomforts eines Fahrzeugs. 2011
ISBN 978-3-86644-681-6

Band 9 Lars Lewandowitz
Markenspezifische Auswahl, Parametrierung und Gestaltung der Produktgruppe Fahrerassistenzsysteme. Ein methodisches Rahmenwerk. 2011
ISBN 978-3-86644-701-1

Band 10 Phillip Thiebes
Hybridantriebe für mobile Arbeitsmaschinen. Grundlegende Erkenntnisse und Zusammenhänge, Vorstellung einer Methodik zur Unterstützung des Entwicklungsprozesses und deren Validierung am Beispiel einer Forstmaschine. 2012
ISBN 978-3-86644-808-7

Band 11 Martin Gießler
Mechanismen der Kraftübertragung des Reifens auf Schnee und Eis. 2012
ISBN 978-3-86644-806-3

Band 12 Daniel Pies
Reifenungleichförmigkeitserregter Schwingungskomfort – Quantifizierung und Bewertung komfortrelevanter Fahrzeugschwingungen. 2012
ISBN 978-3-86644-825-4

Band 13 Daniel Weber
Untersuchung des Potenzials einer Brems-Ausweich-Assistenz. 2012
ISBN 978-3-86644-864-3

Band 14 **7. Kolloquium Mobilhydraulik. 27./28. September 2012 in Karlsruhe.** 2012
ISBN 978-3-86644-881-0

Band 15 4. Fachtagung
Hybridantriebe für mobile Arbeitsmaschinen 20. Februar 2013, Karlsruhe. 2013
ISBN 978-3-86644-970-1

Karlsruher Schriftenreihe Fahrzeugsystemtechnik (ISSN 1869-6058)

Herausgeber: FAST Institut für Fahrzeugsystemtechnik

Band 16 Hans-Joachim Unrau
Der Einfluss der Fahrbahnoberflächenkrümmung auf den Rollwiderstand, die Cornering Stiffness und die Aligning Stiffness von Pkw-Reifen. 2013
ISBN 978-3-86644-983-1

Band 17 Xi Zhang
Untersuchung und Entwicklung verschiedener Spurführungsansätze für Offroad-Fahrzeuge mit Deichselverbindung. 2013
ISBN 978-3-7315-0005-6

Band 18 Stefanie Grollius
Analyse des gekoppelten Systems Reifen-Hohlraum-Rad-Radführung im Rollzustand und Entwicklung eines Rollgeräuschmodells. 2013
ISBN 978-3-7315-0029-2

Band 19 Tobias Radke
Energieoptimale Längsführung von Kraftfahrzeugen durch Einsatz vorausschauender Fahrstrategien. 2013
ISBN 978-3-7315-0069-8

Band 20 David Gutjahr
Objektive Bewertung querdynamischer Reifeneigenschaften im Gesamtfahrzeugversuch. 2014
ISBN 978-3-7315-0153-4

Band 21 Neli Ovcharova
Methodik zur Nutzenanalyse und Optimierung sicherheitsrelevanter Fahrerassistenzsysteme. 2014
ISBN 978-3-7315-0176-3

Band 22 Marcus Geimer, Christian Pohlandt
Grundlagen mobiler Arbeitsmaschinen. 2014
ISBN 978-3-7315-0188-6

Band 23 Timo Kautzmann
Die mobile Arbeitsmaschine als komplexes System. 2014
ISBN 978-3-7315-0187-9

Karlsruher Schriftenreihe Fahrzeugsystemtechnik (ISSN 1869-6058)

Herausgeber: FAST Institut für Fahrzeugsystemtechnik

Band 24 Roman Weidemann
Analyse der mechanischen Randbedingungen zur Adaption der oszillierenden Hinterschneidtechnik an einen Mobilbagger. 2014
ISBN 978-3-7315-0193-0

Band 25 Yunfan Wei
Spurführungsregelung eines aktiv gelenkten Radpaars für Straßenbahnen. 2014
ISBN 978-3-7315-0232-6

Band 26 David Schmitz
Entwurf eines fehlertoleranten Lenkventils für Steer-by-Wire Anwendungen bei Traktoren. 2014
ISBN 978-3-7315-0264-7

Band 27 Christian Schwab
Beitrag zu einer universellen Baggerschnittstelle zur Übertragung elektrischer und hydraulischer Leistung sowie elektronischer Signale für komplexe Anbaugeräte. 2014
ISBN 978-3-7315-0281-4

Band 28 Peter Dengler
Untersuchung zum effizienten Betrieb von Hydraulikzylindern in Konstantdrucksystemen unter Verwendung einer Zwischendruckleitung. 2015
ISBN 978-3-7315-0295-1

Band 29 Manuel Bös
Untersuchung und Optimierung der Fahrkomfort- und Fahrdynamikeigenschaften von Radladern unter Berücksichtigung der prozessspezifischen Randbedingungen. 2015
ISBN 978-3-7315-0310-1

Band 30 5. Fachtagung
Hybride und energieeffiziente Antriebe für mobile Arbeitsmaschinen 25. Februar 2015, Karlsruhe. 2015
ISBN 978-3-7315-0323-1

Band 31 Michael Eckert
Energieoptimale Fahrdynamikregelung mehrmotoriger Elektrofahrzeuge. 2015
ISBN 978-3-7315-0332-3

Karlsruher Schriftenreihe Fahrzeugsystemtechnik (ISSN 1869-6058)

Herausgeber: FAST Institut für Fahrzeugsystemtechnik

Band 32 Martin Scherer
Beitrag zur Effizienzsteigerung mobiler Arbeitsmaschinen. Entwicklung einer elektrohydraulischen Bedarfsstromsteuerung mit aufgeprägtem Volumenstrom. 2015
ISBN 978-3-7315-0339-2

Band 33 Rinaldo Arnold
Automatische Abstimmung der Sekundärseite eines dreiphasigen Systems zur berührungslosen induktiven Energieübertragung. 2015
ISBN 978-3-7315-0355-2

Band 34 Johannes Gültlinger
Kraftübertragung und Fahrbahnverschleiß durch Spikereifen. 2015
ISBN 978-3-7315-0358-3

Band 35 Thorsten Dreher
Energieeffizienz von Konstantdrucksystemen mit sekundärgeregelten Antrieben beim Einsatz in mobilen Arbeitsmaschinen. 2015
ISBN 978-3-7315-0377-4

Band 36 Steffen Kölling
Konzeptionelle Untersuchung zur Neigekompensation von Stromabnehmern. 2015
ISBN 978-3-7315-0387-3

Band 37 Michael Fritz
Entwicklungswerkzeuge für die Fahrzeugklimatisierung von Nutzfahrzeugen. 2015
ISBN 978-3-7315-0384-2

Band 38 Ralf Oberfell
Stochastische Simulation von Energieflüssen im Nutzfahrzeug Ein einsatzorientiertes Bewertungs- und Optimierungsverfahren. 2015
ISBN 978-3-7315-0403-0

Band 39 Christoph Sturm
Bewertung der Energieeffizienz von Antriebssystemen mobiler Arbeitsmaschinen am Beispiel Bagger. 2015
ISBN 978-3-7315-0404-7

Karlsruher Schriftenreihe Fahrzeugsystemtechnik (ISSN 1869-6058)

Herausgeber: FAST Institut für Fahrzeugsystemtechnik

Band 40 Florian Netter
Komplexitätsadaption integrierter Gesamtfahrzeugsimulationen. 2016
ISBN 978-3-7315-0414-6

Band 41 Markus Springmann
Auslegung eines asynchronen Langstatorlinearmotors mit großem Luftspalt als Straßenbahnantrieb. 2015
ISBN 978-3-7315-0418-4

Band 42 Alexander Basler
Eine modulare Funktionsarchitektur zur Umsetzung einer gesamtheitlichen Betriebsstrategie für Elektrofahrzeuge. 2015
ISBN 978-3-7315-0421-4

Band 43 Hans-Georg Wahl
Optimale Regelung eines prädiktiven Energiemanagements von Hybridfahrzeugen. 2015
ISBN 978-3-7315-0422-1

Band 44 Jennifer Heck
Zur Simulation des Rad-Schiene-Verschleißes bei Straßenbahnen. 2016
ISBN 978-3-7315-0443-6

Band 45 Moritz Vaillant
Design Space Exploration zur multikriteriellen Optimierung elektrischer Sportwagenantriebsstränge: Variation von Topologie und Komponenteneigenschaften zur Steigerung von Fahrleistungen und Tank-to-Wheel Wirkungsgrad. 2016
ISBN 978-3-7315-0452-8

Band 46 Philip Nagel
Entwicklung einer Betriebsstrategie zur Energierückgewinnung in hybriden Mehrverbrauchersystemen. 2016
ISBN 978-3-7315-0479-5

Band 47 Matthias Pfriem
Analyse der Realnutzung von Elektrofahrzeugen in kommerziellen Flotten zur Definition einer bedarfsgerechten Fahrzeugauslegung. 2016
ISBN 978-3-7315-0489-4

Karlsruher Schriftenreihe Fahrzeugsystemtechnik (ISSN 1869-6058)

Herausgeber: FAST Institut für Fahrzeugsystemtechnik

Band 48 Mohanad El-Haji
Ontologie-basierte Definition von Anforderungen an Validierungswerkzeuge in der Fahrzeugtechnik. 2016
ISBN 978-3-7315-0496-2

Band 49 **9. Kolloquium Mobilhydraulik**
22./23. September 2016 in Karlsruhe. 2016
ISBN 978-3-7315-0573-0

Band 50 6. Fachtagung
Hybride und energieeffiziente Antriebe für mobile Arbeitsmaschinen 15. Februar 2017, Karlsruhe. 2017
ISBN 978-3-7315-0601-0

Band 51 Fabian Schirmaier
Experimentelle Untersuchung und Simulation des Umformverhaltens nähgewirkter unidirektionaler Kohlenstofffasergelege. 2017
ISBN 978-3-7315-0620-1

Band 52 Mathias Cabrera Cano
Neuronale Netze mit externen Laguerre-Filtern zur automatischen numerischen Vereinfachung von Getriebemodellen. 2017
ISBN 978-3-7315-0621-8

Karlsruher Schriftenreihe
Fahrzeugsystemtechnik

FAST
Institut für Fahrzeugsystemtechnik

Die durchgängige Verwendung von Getriebemodellen in der simulationsgestützten Entwicklung mechatronischer Systeme im PKW gewinnt zunehmend an Bedeutung. Dabei sollen auch in einer späteren Phase der Entwicklung valide Getriebemodelle der Fachabteilung zur Verfügung stehen, wenn weitere Funktionen oder das Getriebesteuergerät durch Simulationen abgesichert werden. Erschwert wird eine durchgängige Verwendung durch die numerischen Eigenschaften von detaillierten Getriebemodellen.
In dieser Arbeit wird daher eine Methode zur automatischen numerischen Vereinfachung von detaillierten Getriebemodellen vorgestellt. Es werden neuronale Netze mit externen Laguerre-Filter verwendet, die eine gute Approximation der Schaltdynamik ermöglichen. In dem gezeigten Beispiel dieser Arbeit ist durch Verwendung des numerisch vereinfachten Modells auch eine Echtzeitsimulation zur Absicherung des Getriebesteuergeräts möglich.

ISSN 1869-6058
ISBN 978-3-7315-0621-8

ISBN 978-3-7315-0621-8
9 783731 506218 >